UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and the Environment

Data Management in Engineering Design

Jonathan D. Owen

Thesis for the degree of Doctor of Engineering

June 2015

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND THE ENVIRONMENT
School of Engineering

Doctor of Engineering

DATA MANAGEMENT IN ENGINEERING DESIGN

by Jonathan D. Owen

Engineering design involves the production of large volumes of data. These data are a sophisticated mix of high performance computational and experimental results, and must be managed, shared and distributed across worldwide networks. Given limited storage and networking bandwidth, but rapidly growing rates of data production, effective data management is becoming increasingly critical. Within the context of Airbus, a leading aerospace engineering company, this thesis bridges the gap between academia and industry in the management of engineering data. It explores the high performance computing (HPC) environment used in aerospace engineering design, about which little was previously known, and applies the findings to the specific problem of file system cleaning.

The properties of Airbus HPC file systems show many similarities with other environments, such as workstations and academic or public HPC file systems, but there are also some notably unique characteristics. In this research study it was found that Airbus file system volumes exhibit a greater disk usage by a smaller proportion of files than any other case, and a single file type accounts for 65% of the disk space but less than 1% of the files. The characteristics and retention requirements of this file type formed the basis of a new cleaning tool we have researched and deployed within Airbus that is cognizant of these properties, and yielded disk space savings of 21.1 TB (15.2%) and 37.5 TB (28.2%) over two cleaning studies, and may be able to extend the life of existing storage systems by up to 5.5 years. It was also noted that the financial value of the savings already made exceed the cost of this entire research programme.

Furthermore, log files contain information about these key files, and further analysis reveals that direct associations can be made to infer valuable additional metadata about such files. These additional metadata were shown to be available for a significant proportion of the data, and could be used to improve the effectiveness and efficiency of future data management methods even further.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Jonathan Owen, declare that the thesis entitled "Data Management in Engineering Design", and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University;
- Where any part of this thesis has been previously submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- Where I have consulted the published work of others, this is always clearly attributed;
- Where I have quoted the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- Where the thesis is based on work done by myself jointly with others, I have made clear what was done by others and what I have contributed myself;
- Parts of this work have been published by the author, as listed in Section 1.5.

Signed: .....

Date: .....

Acknowledgements

I would like to thank Airbus for their help in making this research possible, for their funding, and for their many insights into the realities of industrial high performance computing and aerospace engineering design. Particular thanks to the team with which I had the pleasure of spending my placement years, for their strong support and excellent humour throughout my time at Airbus.

Many thanks also to my friends and family, for supporting my throughout my studies and for helping me to keep things in perspective during the many challenges. I count myself very fortunate to have such a wonderful family and such amazing friends.

Finally, special thanks to my supervisor, Prof. Simon J. Cox, without whom I never would have finished. Thank you for your endless support, enduring wisdom, and for inexplicably managing to always have the best answer to every question, and the most helpful solution to every problem. Most of all, thank you for continuing to motivate me through the difficult times, and especially during the final push towards completion.

Chapter 1

Introduction

The engineering design process involves the production of large volumes of data, for example through computational fluid dynamics, stress analysis and laboratory experiments. These data must be stored and managed across multinational infrastructures and organisations. Little is known about the meaning, purpose or value of much of this data, yet intuitively these are precisely the factors that should be considered when making data management decisions. With ever more rapid growth in data production, there is a real and pressing need for new data management techniques.

The research in this thesis is focused on the valuable metadata that exists within engineering file systems, and how it can be used. Emphasis is placed on the development of effective data management methods and tools; on delivering practical benefits to Airbus (the sponsor), especially in purging unwanted data from existing file systems; and on the consideration of engineering design in the wider context of high performance computing (HPC). This chapter states the specific objectives for this research programme, and explains the structure of the remainder of the thesis.

1.1 Focus of Thesis

"The research in this thesis is focused on the valuable metadata that exists within engineering file systems, and how it can be used."

1.2 Objectives

The specific objectives of this thesis may be stated as follows:

1. Examine large scale HPC data from industrial engineering design and compare and contrast with other data from HPC environments.
2. Identify metadata trends and characteristics that may be used to improve data management practices, particularly data cleaning.
3. Research practical data cleaning methodologies on real engineering file systems at Airbus, maximising savings in disk space whilst retaining all required data.
4. Identify and extract additional metadata from engineering file systems, using novel sources to learn more about the file content, purpose and value.
5. Make recommendations regarding future data storage and management practices to Airbus (the sponsor) and to other practitioners in the field.

1.3 Structure of Thesis

The remainder of this thesis is structured as follows:

Chapter 2: Background - This chapter describes the industrial engineering context in which the research in this thesis was completed, and reviews related work from the field of data management.

Chapter 3: File System Metadata - A study into file system metadata from HPC systems at Airbus, which compares and contrasts with several other environments and identifies trends and characteristics to support improved data management practices.

Chapter 4: Practical Data Cleaning - A report on two highly successful data cleaning exercises carried out at Airbus, detailing the development of the methodology, the substantial disk space savings achieved, and the sustainability of the method.

Chapter 5: Additional Metadata - A study into the extraction of additional metadata from secondary sources such as file names and log files, the availability of such metadata and the usefulness of this information for data management.

Chapter 6: Discussion - A discussion of some of the topics covered thus far, the applicability of the work to other fields, and the ways in which data management methods could be improved.

Chapter 7: Conclusions - A summary of the overall conclusions, and an explanation of how the work conducted in this thesis meets the stated objectives.

1.4 Research Results, Novelty and Sponsor Value

As an engineering doctorate (EngD) research programme, the work was undertaken whilst working closely with Airbus, the sponsoring organisation. From an academic perspective, the novel aspects were the exploration and comparison of a real industrial engineering, considering the characteristics of data and metadata and comparing these with other environments (chapter 3); and the extraction of aerodynamic and other technical metadata from log files, coupled with the association of these metadata with larger solution files. This approach provides an effective way to better describe significant volumes of data (chapter 5).

In addition, it was very important that the research delivered practical results and benefits to Airbus. Significant improvements were implemented in the management of computational fluid dynamics (CFD) data, particularly in the targeted cleaning of unwanted intermediate results, which led to substantial cost and space savings (chapter 4). Recommendations were also made regarding the capture of additional metadata, which may further improve data management in the future (chapter 5).

Perhaps the most valuable output of this research for Airbus is the internal Python program (pDisk) that was written to extract and analyse file system metadata, and to indicate which files could potentially be deleted to save disk space. This tool may be used as often as necessary to purge unwanted files without affecting ongoing project work, and may be able to extend the life of existing storage systems by up to five years. Furthermore, it has been realised that the financial value of the disk space savings made have *already* exceeded the total cost of this entire research programme!

1.5 Publications

The following work in this thesis has been published:

- Jonathan D. Owen and Simon J. Cox. File System Metadata in Engineering Design. Technical Report D15017455, Airbus and University of Southampton, 2015 [62]. *(Chapter 3: File System Metadata).*
- Jonathan D. Owen. Data Cleaning in Engineering Design. Presentation to peers and stakeholders at the July 2013 Airbus UK EngD Workshop. Awarded Best Presentation. *(Chapter 4: Practical Data Cleaning).*
- Jonathan D. Owen and Simon J. Cox. Practical Data Cleaning. Technical Report ME1448107, Airbus and University of Southampton, 2014 [61]. *(Chapter 4: Practical Data Cleaning).*
- Jonathan D. Owen and Simon J. Cox. Additional Metadata. Technical Report ME1448106, Airbus and University of Southampton, 2014 [60]. *(Chapter 5: Additional Metadata).*

Chapter 2

Background

This thesis considers a number of aspects of data management, primarily within the context of high performance computing (HPC) for aerodynamic engineering design at Airbus. Data management, HPC, and engineering design are all broad fields, encompassing many disciplines. This chapter provides a brief summary of the aspects of each of these fields that relate to the studies, findings and discussions of the remainder of the thesis. It also describes some of the background context and some of the prior work that led to the commencement of this research programme.

2.1 Data Management

The management of data is a broad field, encompassing many disciplines across many vastly different environments. Broadly speaking, data management is an administrative process through which data are gathered, stored, protected, catalogued and shared, throughout a natural data lifecycle after which the data are archived or deleted. The details of the many data management processes vary in order to meet the needs of end-users and data requirements. These data requirements may include important specifications such as those detailing the security, accessibility, and retention of the data. This section provides a brief overview of the aspects of data management that are most relevant to the discussions in this thesis.

2.1.1 Data Lifecycle

There are many ways to describe the various stages through which data moves throughout its lifetime. Data lifecycles provide a *"structure for considering the many operations that will need to be performed on a data record throughout its life"* [13]. Figure 2.1 shows a simple and generic example of a data lifecycle. For a given context, it is possible to map specific activities to the various stages in this lifecycle. For example, in the context of the engineering design process (explained in detail in section 2.3), creating a new computer aided design (CAD) model would be a data *creation* step. The aerodynamic properties of this model are then *analysed*, thereby *creating* new results files, and the model and the output results are then *stored*. After repeating these steps for a variety of designs, a user will *retrieve* the results for many models and make a design decision. Once results are no longer the subject of ongoing analysis, they can be *archived*. Finally, if and when files are no longer required at all, they can be *deleted*.

The model in figure 2.1 decsribes the basic common characteristics. Ultimately, files will always be created, analysed and stored; will generally need to be searched and retrieved; must be preserved for some length of time; and will then be archived and potentially disposed of if and when they cease to be relevant or required. However, the particular choice of data lifecycle model and the resulting processes by which data are managed will greatly depend on the needs of the users interacting with the data. For example, in the academic community, there is a strong need to share data between many organisations using different software packages. This presents *"unique challenges for preservation, including variety and complexity of file formats, as well as sheer volume, which continues to increase rapidly"* [19]. This means that it is also necessary to preserve the software responsible for data analysis, as well as the more commonly expected information regarding provenance, structure and description. This results in correspondingly increased complexity in the *storage* of data sets, which must now reference the tools, software versions, algorithms, etc., that are necessary to make effective use of the data.

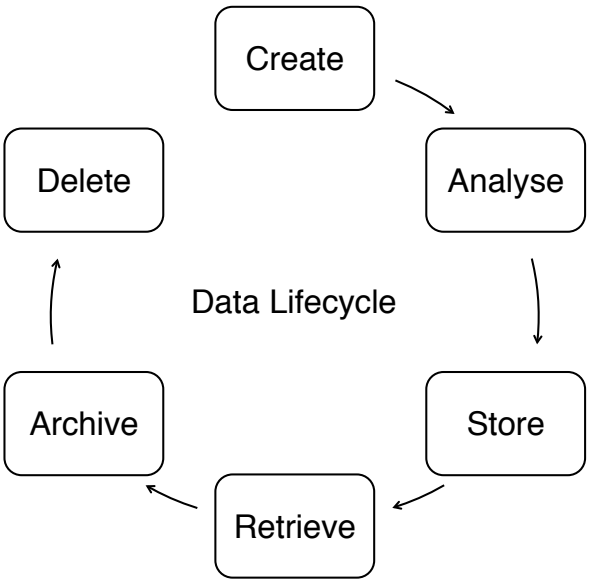

FIGURE 2.1: A generic data lifecycle. Not all data will be handled the same way, but they will broadly be subjected to this type of lifecycle. One notable exception would be particularly critical files that may be archived indefinitely rather than being deleted.

In a given industrial organisation, there is likely to be much less variety in the software tools tools associated with the production and analysis of a data set. However, it is still important to take note of the data provenance, particularly in terms of software versions, which may lead to substantially different data analysis as algorithms are improved or replaced within a software package over time. Some similar considerations to those in academia are thus necessary, despite the differences in context, environment and purpose.

Amidst a huge volume of seemingly similar data it can be very difficult to identify the correct data needed to inform decision-making operations. In industry, where an organisation may perform a similar type of analysis on the same product under a wide variety of conditions, data search and *retrieval* can thus be a key issue. As a result of the rapid growth of data generation in recent years, engineers reportedly spend *"50 percent or more of their time looking for the information they need to do their jobs"* [71]. With this in mind, it is hardly surprising that effective data management can lead to significantly increased productivity and corresponding cost savings. Enterprise data management aims address such issues by linking data management, process management and context management, and by working collaboratively across various disciplines throughout the lifecycle of the products with which data are related [71].

Also related to data search and retrieval are Electronic Document Management Systems (EDMS/EDS) and Product Data Management (PDM) systems. EDMS typically use a database to store all information about users, rights, accessibility, as well as information *about* the data. The data are stored separately, with users performing search and retrieval through the database, which can perform rights and access management before linking users to the appropriate data. PDM systems may take this concept further by integrating with applications, providing previews, and allowing finer-granularity version tracking. For example, in CAD environments this may mean linking changes to the specific parts that were modified [15]. Such systems are helpful in large organisations, although complexity is increased rapidly when multi-disciplinary collaboration is required, as the needs to different user groups may vary considerably.

For large-scale scientific data management in general, a capability maturity model has also been proposed, with a view to aiding *"organisations in evaluating and planning improvements to their [scientific data management] practices"* [20]. The proposed model considers five levels of maturity, from a reliance on individual performance and intuition through to an optimized, managed and measured data management system. It also identifies many key processes for effective data management, and breaks down high level areas such as *"data description and representation"* into more specific practices that should be considered. This is particularly helpful for industrial organisations, where data can easily be viewed as a by-product of the main activities, rather than as a valuable asset with serious needs for sharing and preservation.

2.1.2 Users and Administrators

Depending on the parent organisation, the type of user responsible for particular data management processes may change throughout the data lifecycle. Expanding on the previous example, the creation, analysis and storage steps may be undertaken by application users working on CAD designs or computational fluid dynamics (CFD) solvers. Data search and retrieval may be executed by the same type of application user, or possibly by a data auditor looking to understand the justification behind certain design decisions. Lastly, data may be archived or deleted by administrators or management personnel, or this responsibility may be devolved to the original application users. In addition, the actual storage systems will be managed by a group of IT technicians, although IT systems and their management are often outsourced and provided as a service.

In this thesis, we consider a context where a typical application user has an engineering role and an assumed responsibility for managing their own data. An administrator will provide some guidance and information regarding best practice, but the user must act on this information in managing their data. The administrator also has a view of the system as a whole, and is responsible for capacity planning and ensuring that sufficient space is available on existing storage systems for users to carry out their work. This is

usually done by either ensuring that users carry out their own data management, and reminding particular users when quotas or limits are reached; or by organising larger and more systematic cleaning exercises. The interactions between these application users and administrators, while generally beyond the scope of this thesis, are very interesting, as there can be a delicate balance between collaboration and confrontation.

Users often have a good view of what data is worth preserving, and what could be removed. For example, data generated as part of staff training may be of no long-term value whatsoever, but to a third party may appear identical to highly valuable data. Another user behaviour worth considering is the copying of data to ensure that a "clean" original is always available, which can quickly become very wasteful when applied to large data sets. Copied data could potentially be tagged as such, in order to simplify redundant data disposal and to assess the usage of storage resources.

2.1.3 Metadata

Metadata can be simply described as "data about data" or "information about information". It can be described in more detail as "structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource" [57]. This is a very broad definition, and there are many ways to describe data. In addition to general descriptive terms, there may also be many context-dependent fields that could be used for particular environments. As such, metadata plays a vital role in many computing applications, including databases and Internet search engines [87].

A simple example of metadata is that of songs in a music collection. Each track can be tagged with various fields to help classify and describe the song, such as the artist, album and year. These fields can be left blank if not applicable, and are not required in order to play the track. However, they may be very helpful to a user in trying to find a specific song in a large collection. Media library applications such as iTunes [11] and Windows Media Player [52] make extensive use of the available metadata in music files to provide search and sorting functionality for end-users.

Another example is the metadata used in digital photography. Digital cameras store extensive information about the settings used to create an image, such as the shutter speed, aperture size and ISO. This is commonly achieved using the exchangeable image file format (Exif), which combines existing file formats with specific metadata tags [75]. They may also apply information about the make and model of the camera and lens used, as well as information about the owner of the camera. Using GPS, it is also possible to store the location from which the image was taken. None of these metadata fields are required to view the image, but the additional information may be used to enhance search and retrieval from a larger collection, as implemented in photography software such as Aperture [9] and Lightroom [1].

There are several ways in which metadata can be categorised. A simple view is that metadata can be either "technical metadata" or "business metadata" [37], which essentially divides the information into that which is useful to technicians and that which is useful to a typical end user. An alternative view illustrates "descriptive metadata" for discovery and identification, "structural metadata" for creating compound objects, and "administrative metadata" for managing a particular resource [57].

Metadata is also used in commercial and industrial environments. In some cases there is a substantial gap between technical metadata and business metadata. Technical metadata is important for design, development and maintenance of data resources, while business metadata is useful to the end-users in understanding the business context of data [37]. In contexts that are intrinsically very technical, such as engineering, it may be somewhat more difficult to distinguish between technical and business metadata. However, it is worth noting that different perspectives call for different information, as this has a strong impact on the metadata requirements.

As mentioned previously, it is very important to record the provenance of data. This could include metadata such as the author, the project for which the data was generated, and details of the context or circumstances for which the data set is valid. It could also include details such as the software and version used for creation, the particular configuration or settings applied in this process, and in the case of HPC data, details of the hardware on which the parent HPC job was run. Fields describing this type of provenance metadata are very useful for identifying data, thereby enhancing search and retrieval, but can also aid any future analysis by ensuring conditions are suitably similar for data sets to be comparable.

Technical metadata may also specify requirements for security, storage, and backup procedures. In the simplest sense, the size of a file indicates how much storage space is required if this file is to be moved; and in the case of particularly large single files, whether the destination file system is able to support files of this size. However, more detailed administrative metadata may support more complex requirements. For example, some data may be particularly sensitive and restricted to access by a particular set of users. In the aerospace industry, any certification data must be preserved for the entire service life of the aircraft, thus storage requirements and backup procedures are often much stricter for this type of data than for design data. This again relates to data provenance, and shows how data management can be significantly affected by purpose and context.

Overall, there are many types of metadata, and different information can be useful for different purposes. Some fields may be specific to one particular application, while others may be useful for many purposes. One of the key challenges in metadata management is the identification of useful metadata, as it can be difficult to anticipate the needs of future processes and applications, and therefore to supply the required information.

2.1.4 File Systems

File systems are responsible for the persistent storage of data, and enable basic functions such as search, create, read, update and delete (sometimes referred to as SCRUD) [48]. These functions cover the most basic needs of the end-users, and more complex operations can be built on top, either directly in the operating system or as part of applications or separate utilities. Files are often stored in a hierarchical structure, using directories or folders to organise files and subdirectories. Many file systems also implement journaling, where changes are recorded before being committed, enabling *"fast file system recovery after a crash"* [66], thereby avoiding data loss in the event of a failure.

Although not necessarily the best option for every circumstance, general purpose file systems are applicable to a wide variety of environments, from desktops to mobile devices, and removable media to web servers. Storage can also be accessed over a network using a distributed file system protocol such as Network File System (NFS). However, more specialised file systems are often more appropriate in high-performance environments, where they must be able to deal with vastly increased throughput, concurrency and scalability. In HPC systems, these requirements are often met through the utilisation of parallel and distributed cluster file systems such as Lustre [43], Ceph [85] and GFS [31].

File systems typically store some basic metadata for each file, including the file size, last access time, last modification time, owner ID, and so on. These metadata are stored in the inode, but notably the name of a file is instead stored in the directory that contains it [14]. Many file systems, including NTFS (Microsoft Windows), HFS+ (OSX) and ext2, ext3, ext4 (Linux), also provide some support for the storage of additional or extended attributes, although these are not supported by the NFS protocol [7]. Extended attributes are user or application defined, and can be used to store information such as the name of an author, or the size or resolution of an image. This type of metadata can either be stored in file forks, which provide storage for *"additional content in separate data areas alongside the file content"* [69], or as additional file attributes in the inode, depending on the file system.

It has been argued that *"users needs more effective ways of organising and searching their data"* [74]; that the *hierarchical namespace was invented decades ago and is no longer adequate in managing files today"* [45]; and that the *"simple hierarchical model has outlasted its usefulness"* [70]. To this effect, there has been a great deal of work in alternative ways to organise files, and to improve data access, search and retrieval. Semantic file systems were proposed to provide *flexible associative access to the system's contents by automatically extracting attributes from files with file type specific transducers"* [32], and organise files by properties such as author, date, subject, etc. [56]. However, this approach still relies on extracting useful information from files, and users can be understandably reluctant to relinquish control of the organisation of their data.

2.1.5 Search Applications

As already discussed, one of the key challenges in data management is the search for desired data. Most operatings systems, provide a search feature to help locate specific files based on given criteria, including Windows (Windows Search) [54] and OSX (Spotlight) [8]. These search tools index files based on their names; by their content, for certain file types where this is possible, sometimes supporting custom metadata import plugins for extending functionality to proprietary formats [8]; and by any other recognised forms of metadata, such as Exif metadata for photographs, and tags for audio files.

Search tools such as Windows Search and Spotlight can be greatly beneficial to users searching for data. Indexing files allows for significant performance improvements compared to non-indexed, brute-force searches for data. Building additional features on top of a hierarchical file system provides familiarity to users, who may chose to access their data in whatever way best suits their needs. The extended use of a hierarchical system also means that there are no issues with backwards compatibility, and legacy applications can continue to function as normal.

In addition to the content and attribute based functionality implemented by many search applications, data search can be improved by considering the context in which a file was created, including *"other concurrently accessed files, the user's current task, ... any actions or data that the user associates with the file's use."* [74]. In a world of increasingly mobile computation, this can also correspond to physical location in the real world [34]. Context-enhanced search has been shown to notably improve search effectiveness for only modest increases in query and indexing times [74].

It has been recognised that *"the efficiency of content-based search heavily depends on files that contain explicitly understandable contents"* [35]. In desktop environments, where documents and other files often adhere to key file types that can be easily analysed, this causes no major problems. On the other hand, this is a major limitation for the organisation of proprietary data formats and large binary objects that cannot be easily analysed or understood. In the industrial engineering context considered in this thesis, such files are commonplace, thus data search is a particular challenge and any sources of descriptive metadata are particularly valuable. To this effect, consideration of metadata produced by applications has been strongly recommended [29].

Furthermore, it has also been observed that *understandably, most users are unwilling to perform the difficult and time-consuming task of manually assigning attributes to their files"* [74]. It follows that there are issues with the effective application of semantic file systems or advanced content-based and attribute-based search systems to large volumes of existing data. This thesis considers content and context analysis in a specific industrial engineering environment in chapter 5, with a view to improving search effectiveness through improved data description.

Many engineering environments make use of product data management (PDM) systems. However, these existing systems are often specific to individual departments or contexts, and greatly limit multi-disciplinary collaboration due to the lack of data sharing. With this in mind, it has been argued that *"fields such as computer-aided engineering and design... are predestined for applying the semantic file system"* [26]. Such environments are also well-suited to the capture of additional metadata upon data creation, and it has been noted that *"it is useful to know how... [data] is generated"* in order to properly interpret scientific results [40]. The integration of such systems in a collaborative and multi-discplinary manner may lead to substantially improved productivity in the future, but is unlikely to deal with any of the challenges associated with managing large volumes of *existing* data, hence the focus of this thesis on the latter.

2.1.6 Tiered Storage

There are many different types of persistent storage, including tape drives, disk drives, and solid state drives, each with their own cost, performance and security characteristics. In addition, the inclusion of RAID provides many possible configurations for trading off redundancy and performance in arrays of these drives. Furthermore, there are many options for the external backup of data. It follows that there are many possible ways to store data physically, and that each has a corresponding cost and performance. The best choice of storage depends very much on the environment and on the particular data in question, and may change throughout the lifecycle of the data.

Tiered storage allows multiple types of storage to be transparently amalgamated into a larger storage system. This combines the benefits of each storage type while minimising the cost. For example, frequently accessed data may be stored on a fast but expensive solid-state drive, while less frequently accessed data could be moved to a slower but cheaper disk drive. This concept has been recently implemented by Apple in the Fusion Drive [10], but also has commercial and industrial applications. The automated management of data between multiple storage ideas typically relies on data access patterns and requirements [44]. However, it could potentially be based on other aspects of the data lifecycle and made more cognizant of the nature and characteristics of the data being managed. This would enable the implementation of more sophisticated data management policies.

2.1.7 Object-Based Storage

Permanent storage devices used similar block-based interfaces for many years, but more recently started to become something of a performance bottleneck. Object-based storage combines the benefits of file access and block-access. Like blocks, storage objects can be accessed directly on a storage device, thereby offering the excellent performance of block-based access. However, they can also be accessed through a more file-like interface, for easy accessibility from multiple platforms [49]. This combination of high performance and easy accessibility is extremely valuable, particularly in HPC environments. As such, object storage devices (OSD) are used in many high performance environments, and are implemented in the Lustre [43] and Panasas [63] parallel file systems, both of which are very common in HPC environments [79].

Object-based storage systems offload storage management operations from the file system to object storage devices, allowing the storage device to take care of internal space management. This separates user components such as presenting logical data structures from the storage components that map the data to physical storage. Moreover, the physical storage does matter to the OSD, which simply defines an alternative interface between the file system and storage [49]. With this in mind, object-based storage could be linked with tiered storage to combine high-performance and very large scale longer-term storage, with automated movement of data between storage media as necessary. An alterative solution is the automated management of data replication [84], which can improve performance and help to protect against the inevitable failures experienced by large storage systems.

The architecture of object-based storage systems provides significantly improved scalability. Data stored on OSDs, and can be accessed directly by clients, providing excellent performance. Metadata, attributes, and queries are handled separately by object managers, which grant capabilities to clients, ensuring vastly improved security when compared with storage area networks (SAN) [49]. Decoupling the metadata and data paths, and allowing direct access to storage devices eliminates the performance bottlenecks and provides the scalability required for high performance environments. Moreover, this performance can be combined with a scalable, semantic namespace to provide the type of semantic navigation, virtual directories, and search capabilities offered by semantic file systems [40].

2.2 High Performance Computing

Parallel and distributed computing has been the source of many breakthroughs, especially in scientific and engineering computing where compute power is a common limitation for research and development. Advances in parallel in distributed algorithms, coupled with the increased availability and affordability of HPC solutions, have greatly increased HPC usage in many environments.

2.2.1 Clusters, Commoditisation and Services

A common way to implement high performance computing is through compute clusters, where many individual compute nodes are linked together. Each node may contain several compute cores, and the system is interconnected using a high speed interconnect such as InfiniBand. Compute power can be increased by adding compute nodes, or larger facilities can be developed by linking multiple clusters. Work is shared across many-core compute clusters using parallel and distributed algorithms, often based on technologies such as Message Passing Interface (MPI).

Supercomputers have traditionally been highly specialised, often relying on many bespoke components. However, in recent years there has been a shift towards using commodity components where possible, such as CPUs, RAM and other key parts. Careful design is still needed to balance the huge power and thermal requirements, but off-the-shelf components have greatly improved the affordability of HPC systems. The modularisation of clusters has also enabled much easier expansion of HPC systems, again increasing the usage of HPC systems.

Another key factor that has increased the availability of HPC systems is the service-based model. Instead of purchasing the HPC hardware, companies can now lease complete systems, essentially outsourcing the configuration and management. There are many business cases where this is a very appealing proposition, due to the fixed costs, reduced maintenance, and increased business flexibility. The service model also helps organisations to focus on their value-added activities and get on with work, rather than requiring significant expertise in HPC systems and maintenance.

2.2.2 The Cloud

Cloud-based computing is an HPC model where compute power can be acquired as needed from large cloud computing providers such as Windows Azure [51] and Amazon Elastic Compute Cloud (Amazon EC2) [4]. A particular advantage of this approach is that it eliminates the need for companies to purchase and maintain expensive high performance compute clusters. Although many high performance computing systems

can be rented as long-term solutions, cloud-based computing enables instant scaling based on the workload and the number of cores required. This responsiveness to change can be extremely beneficial where the workload is highly variable.

In terms of data storage, cloud storage can be purchased on a similar *pro rata* basis to cloud computing. This model is greatly appealing to many industries, including engineering design. A cloud computing and storage provider will be able to focus their attention on delivering maximum performance for end-users, and to balance the huge investments in resources and infrastructure between many customers. Outsourcing storage and high performance computing to the cloud can thus provide clear benefits simply through economies of scale, and consumers need only pay for the services that they actually need.

However, there are a number of challenges that inhibit the adoption of cloud computing over alternative high performance computing methods. One of the key issues relates to network performance and accessibility. In the context of continuous globalisation in many industries, the centralisation of storage and computing through a cloud-based provider requires a significant networking infrastructure to allow data to flow to and from the cloud. Many organisations have a substantial internal networking infrastructure, but relatively small external networking capabilities, since the majority of network traffic is internal. Moving resources to the cloud would alter this underlying assumption and thus require the enhancement of existing external networking capabilities.

The relocation of computing and storage to the cloud presents further issues in transnational and global organisations due to the often highly distributed nature of existing resources. Consolidating high performance computing in the cloud inherently requires centralisation, which will induce varying networking latencies and bandwidth limitations for different sites simply due to the geographic separation. The possible locations for centralisation will be limited by the availability of cloud vendors, and the performance costs of data transfer under these conditions may be prohibitive.

Perhaps more significant are the security and legal implications of cloud computing for the engineering industry. The transfer of data across external networks and the sharing of confidential and potentially sensitive data with external parties are serious concerns. While the economies of scale from cloud vendors are greatly appealing, the security and privacy of company data is paramount. Unless significant efforts are made to maintain confidentiality and security at every stage of data computation and transfer, cloud-based solutions will not be viable for many organisations.

The challenges facing cloud-based solutions for the engineering industry may be overcome in time. However, it is clear that the current technical, security and legal implications are critical factors that inhibit the cloud as a viable high performance computing and storage solution for engineering.

2.3 Industrial Engineering Design

Industrial engineering design is a very broad field, incorporating a large number of companies and organisations, each of which may possess certain unique qualities and present different challenges. The work described throughout this thesis was undertaken within one specific department in one particular leading aerospace engineering company: Airbus. It follows that some of the challenges faced in this environment may not be applicable elsewhere. However, the work was carried out in recognition of the wider context and in an abstract manner wherever possible.

2.3.1 Engineering Design

The nature of the work undertaken on any system will determine the nature of the data that are created. Engineering design is a very methodical process, with much of the work focusing on refining a particular product or design. This will involve incremental changes to a base design, followed by some analysis of the outcome, which will guide further changes to the base design, and so on. A company will be working on a number of active products, each of which will be made up of many smaller components. The design process will be applied to both components and assemblies, and subjected to a variety of constraints relating to financial limitations, marketing, and the existence and availability of suitable manufacturing techniques.

Multidisciplinary optimisation techniques are required in order to reach the best overall designs. There are typically multiple objectives, for example to minimise drag, maximise lift and minimise weight, so the design process requires multiple types of analysis, such as aerodynamic analysis, mass estimation and structural analysis. Each of these types of analysis will affect not only the relative merit of each design, but also feed back into the constraints. For example, an excellent design from an aerodynamic perspective may be structurally weak, or a structurally strong design may be too heavy for purpose. The multidisciplinary nature of engineering design thus significantly adds to the overall complexity, and design optimisation is thus a very active field of research [30, 38].

In addition, there may be many dependencies between the analyses of different components, for example where the same designs are used in multiple assemblies. This is a very likely scenario for a large scale organisation working on multiple products, as parts will be reused where possible in order to improve the manufacturing efficiency. However, using the same parts in different assemblies will still require some analysis in order to determine the applicability of the design to a new set of constraints, and computation will certainly be required on final configurations for legal and certification reasons.

2.3.2 Computational Fluid Dynamics

The research presented in this thesis was undertaken within the context of aerodynamics. Computational analysis in aerodynamics mostly comprises the computational fluid dynamics (CFD) analysis of engineering components. This means examining how fluid flows over a particular object. In the aerospace industry this fluid is most commonly air [6], but the same basic process applies to water in the maritime industry, and indeed to any situation where the flow of fluid over a solid body is of importance [18]. As such, the research described in this thesis has implications for a number of environments across a very broad industry.

After creating a new geometry file, commonly by making some changes to an existing file in response to previous analysis, a mesh file is created to represent the geometry of the object in the flow field. Meshes can vary greatly in fidelity, depending on the purpose and requirements of the analysis, and in some cases multiple meshes may be generated to meet multiple requirements or to check for any variation in the results. The mesh file will then be used to generate a 3D flow solution for the given flow conditions, such as Mach number (M), Reynolds number (Re), and the angle of incidence to the direction of fluid flow (α). The flow solution can then be post-processed to extract the key forces, such as lift and drag, or to produce a visualisation of the flow field. It is also common to execute jobs for a wide range of flow conditions on a single geometry, in order to plot a polar curve of the final results.

In addition to being executed for each new geometry, it may be desirable to perform the analysis for different mesh fidelities. The different levels of detail correspond to different stages in the design process, and result in the production of different volumes of data. For example, at earlier stages in the design process, a low fidelity mesh may be adequate to assess the approximate relative merit of a large number of designs, and the analysis can be run again later at a higher fidelity when only a few designs remain. The final designs can then be run at a very high fidelity in order to ensure safety, eliminate any potential manufacturing issues, and to satisfy the many rigorous certification regulations for the industry.

The files generated vary greatly in size throughout the different stages of the workflow. For example, a 30 MB geometry file may lead to a 200 MB mesh file, each resulting flow solution might be around 2 GB, a visualisation file could be about 100 MB, but the lift and drag forces can be reduced to individual numbers. In addition, it is common for a large number of very small log files to be created at each stage of the process. These log files record important configuration information regarding computational parameters, progress and status information for key stages in the job, and file paths to the relevant input and output files. Files relating to a particular job are typically stored in the same directory structure, but this is not guaranteed.

The overall job completion time can also vary greatly, from a matter of hours for simple cases, to several weeks for more complex jobs. There is often an end-user desire for jobs to be completed overnight, as this allows engineering analysis and decision to be made during working hours while computational work is executed at night. The ability of an HPC system to meet this desire depends on the size of the job, but is also heavily impacted by scheduling. Engineering HPC systems often operate at more than 100% capacity, meaning that there are more jobs that can be executed at any one time. Scheduling these jobs in a fair and timely fashion is a complex problem beyond the scope of this thesis, but it is clear that scheduling has a strong impact on the total time to job completion.

2.3.3 Resources and Data

The key resources used in the engineering design process are people and HPC facilities. Ultimately, people are responsible for making design decisions, implementing changes to designs, and orchestrating the computational analysis of the designs. In contrast, HPC is simply a tool to speed up the otherwise prohibitively complex calculations involved in aerodynamic analysis. However, the use of HPC in engineering allows these calculations to be carried out relatively quickly, thus greatly improving the ability of engineers to analyse and visualise complex problems.

Any data created by an employee is especially valuable, since it should reflect the result of careful deliberation and prudent decision making. Accurately recreating such data can be very difficult, due to the human element to the decision making, and also very costly in terms of the human time involved in deliberating and reproducing the data. In a transient environment where users may change roles or leave the company entirely, these issues become drastically more complex. It is thus desirable to retain all user-created data indefinitely, wherever possible.

Data created by HPC systems is valuable because of the cost of providing HPC services, and the computational resources that are required for data creation. However, if the original inputs are still available, then intermediate HPC results can potentially be regenerated from the original input files. Moreover, if the original output files are also available, then the regenerated results can be validated against the original outputs. However, there are a number of important considerations regarding HPC data regeneration, including the numerical accuracy; the impact of distributed algorithms and the computing architecture; and the versioning of the relevant software packages. Despite these difficulties, the retention and potential regeneration of intermediate HPC data may be useful topics to examine in more depth.

2.3.4 Networking and Infrastructure

A large engineering organisation may have several compute clusters, internally connected using a high speed interconnect such as InfiniBand, and externally connected using a site backbone network such as 10 Gigabit Ethernet (10GigE). This backbone may also support many departmental offices, each connecting tens or hundreds of workstations over a more local network such as Gigabit Ethernet.

HPC nodes have some disk space for the temporary storage of input, intermediate and output data, usually based on a high performance parallel file system architecture such as Lustre, GPFS, or a similarly parallel or distributed file system. However, these storage systems are very expensive, thus simpler Network File System (NFS) volumes may be connected to the HPC systems via the site backbone network. This allows data to be offloaded from the high performance storage systems upon completion, since performance is less relevant for longer-term storage, resulting in a much more cost-effective solution.

Although some companies may have a single site architecture, many are geographically distributed across multiple sites nationally, or even transnationally. In these cases, the HPC systems may also be distributed amongst the discrete sites. Companies will typically employ dedicated links between sites in order to share data, often using high speed fibre optic networks in order to provide the maximum possible bandwidth, with the internal architecture of each site otherwise similar to that already described.

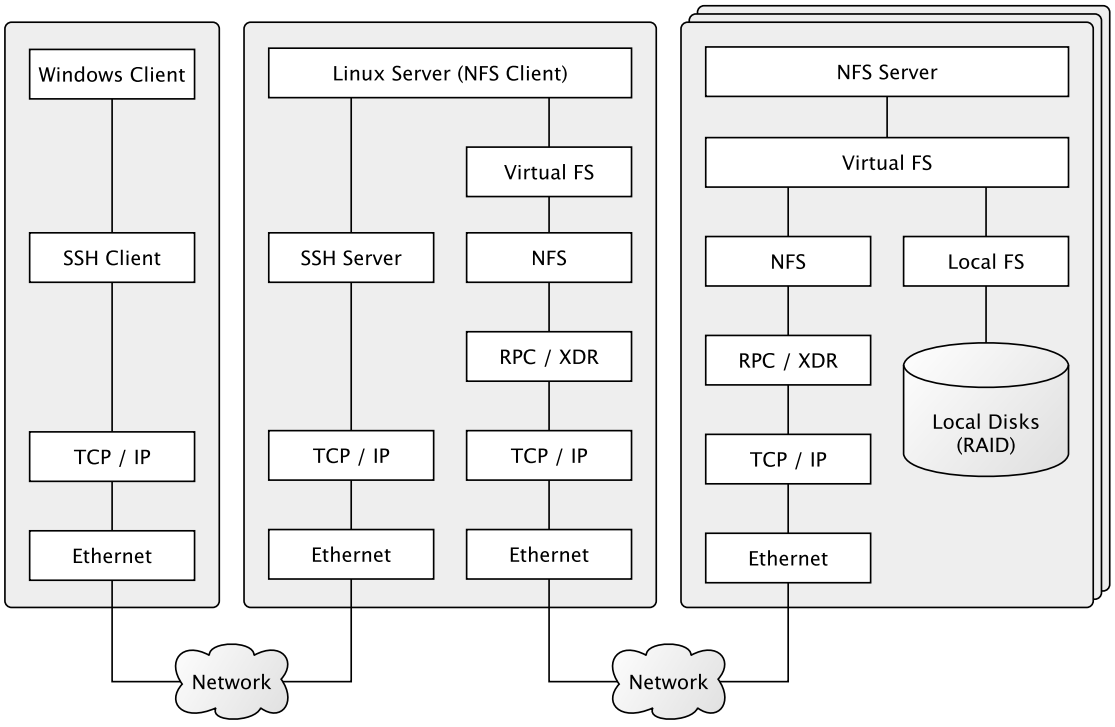

FIGURE 2.2: An architecture diagram of the storage systems considered in this thesis.

Figure 2.2 shows an architecture diagram of the storage systems considered in this thesis, and the way in which data is made accessible. Users connect to a Linux-based server from their Windows-based desktop computers using SSH. The Linux server acts as an NFS client, mounting all of the many NFS volumes into a logical hierarchy. The NFS volumes contain many large disk drives, and use RAID to protect against hardware failures. In addition, there is a tape-based backup service for all volumes except those marked as "no-backup".

2.3.5 Usage and Data Cleaning

In principle, the no-backup volumes exist as a temporary storage area for data to be automatically copied into after generation by HPC jobs. Important data should then be moved onto a volume that is backed up as soon as possible. However, when sufficient storage space is not available on backed-up volumes, the tendency is for users to start working purely on the no-backup volumes. This is an interesting human behaviour that occurs in response to a lack of appropriate storage space, and an example of how users tend to adapt to make use of whatever resources are available. Unfortunately this makes it difficult to determine which files have been properly moved to backed-up volumes and which have not. The management of this data thus becomes much more complicated, as there is no way to explicitly determine which files can be removed. As mentioned previously, this problem is greatly exacerbated when data owners change roles or leave the company.

In desktop computing, the operating system often provides a utility for cleaning temporary or commonly unwanted files. For example, the Windows Disk Cleanup Utility [53] helps a user to clear disk space by deleting all files in the `%temp%` area, removing all temporary internet files, emptying the recycle bin, and so on. This approach relies on cleaning standard locations where temporary or unwanted files are commonly stored. However, in the industrial HPC context considered in this thesis, it cannot be assumed that all removable files will be stored in this way.

One way to manage this type of scratch storage in HPC environments is to simply purge all old files older than a predefined threshold [58, 81]. However, this approach risks losing valuable data if it has not yet been safely moved to longer-term storage. This is greatly exacerbated when there is no space available on suitable longer-term storage systems. A common alternative is to rely on a quota-based system, but this simply leaves any data retention issues with the application users, and does not address problems with a lack of available storage space on suitable systems. The management of scratch volumes in an industrial context, and the timely deletion or retention of data to ensure sufficient storage availability are thus key issues.

2.3.6 Transnational Proof-of-Concept

Previous work in transnational data management showed how commodity technologies can be used to enable data search and retrieval, and how existing business applications can be seamlessly integrated across a heterogeneous infrastructure [46]. This proof of concept was developed to support a scenario requiring the collaboration of engineers between two geographically separated sites, and continuing service in the event of a failure at one of these locations.

A diagram of the architecture tiers conceived for the proof of concept is shown in figure 2.3. The isolation of the user interface, the services and the actual data allow for modularity and flexibility in the deployment environment and the available resources. For example, the illustrated services could be used from a variety of user interfaces, which could be standalone, web-based, or integrated with other business applications.

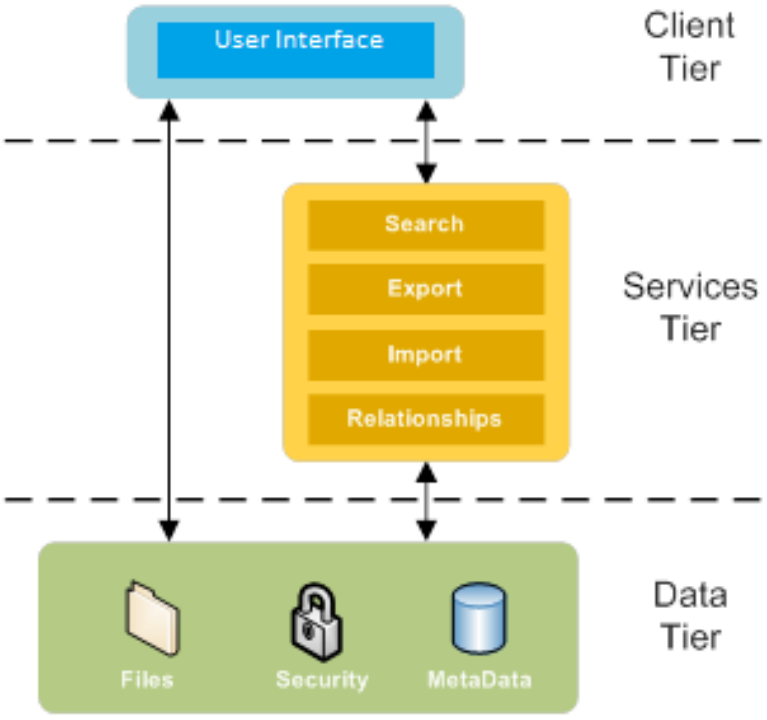

FIGURE 2.3: Proof of Concept Architecture Tiers (Adapted from [46])

The physical infrastructure envisaged during the proof of concept is shown in figure 2.4. This highlights the back-end communication between file servers, as well as the interface integration with an existing business application, and the implementation of a common web-based user interface.

The research described in this thesis began in response to this proof of concept. At the time, the network performance and data management were proving to be significant bottlenecks to the performance of the system as a whole. The initial objective was to improve the overall performance of the system through improved data transfer and intelligent data caching. However, the network problems were soon eliminated through minor upgrades to the infrastructure and small improvements to the way in which files were transferred at the start of a new HPC job. The research thus shifted accordingly to consider the improvements that could be made through better data management. This also made it possible to focus on a single site, rather than the entire global supercomputing infrastructure, thereby eliminating any security or data access concerns that may otherwise have been posed.

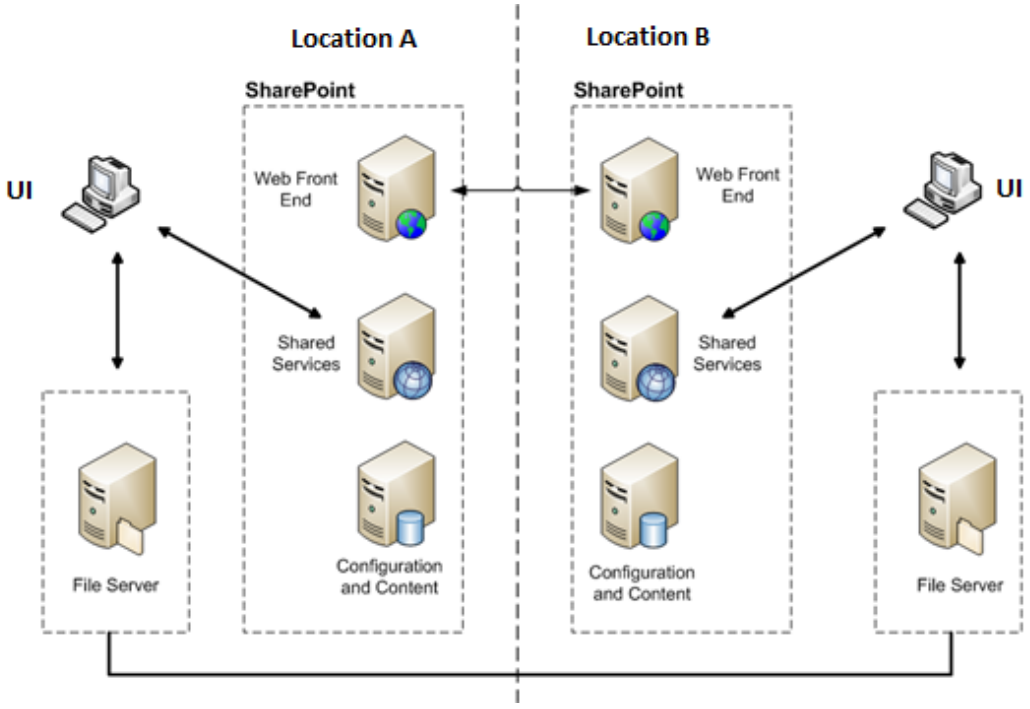

FIGURE 2.4: Proof of Concept Physical Infrastructure (Adapted from [46])

2.3.7 Key Challenges

As discussed throughout this chapter, there are many key challenges currently facing the management of data in the context of industrial engineering design. These challenges span several of the data management lifecycle steps described in section 2.1.1. The rate at which data is produced in by HPC facilities running CFD applications means that it can be difficult to ensure that there sufficient space to *store* new data in the longer-term once it has been created and analysed. Mitigating this problem without adding excessive storage capacity means facing the challenge of *deleting* data that is no longer required, in order to make way for new data. It is thus necessary to balance the long-term need to retain some data for certification purposes against the short-term storage capacity needs.

A related issue pertains to the search and *retrieval* of specific data, for example when revisiting earlier analysis or performing an audit to understand the justification behind design decisions. Given the volume of data already stored, it can be difficult to differentiate between data sets and identify the purpose and provenance of specific files. Much of this desired information is known by application users, but is not capture and stored with the actual data. This greatly complicates data search and retrieval when the data creators are not available, or when they move on to new roles within or outside the organisation. There is thus a challenge in capturing sufficient metadata to enable meaningful data search within engineering data, whilst minimising the required human effort in describing each file and collection. Again, this is a particularly prevalent issue for existing data, where the original data owners may already be unavailable.

2.3.8 Limitations

One of the conditions placed on this research programme was that any additional load placed on the Airbus file systems should be kept to a minimum, in order to avoid any disruption to the on-going engineering activities. The suggested approach was to record snapshots of the file system properties so that the majority of research and analysis could be conduced off-line. It was also requested that the research utility developed to explore and analyse these file systems be written in Python, and snapshot data be stored using PyTables, so that all code could be easily reviewed before execution.

These measures were justifiably established in order to protect critical corporate systems. However, they did mean precluding the use of live metadata registries and alternative database systems like NoSQL. Although such technologies may have opened up some interesting avenues for the research, there were no issues in meeting the research objectives with those that were available. It is also important to note that the tools developed were only research prototypes focused on data exploration, and that production implementations may still be able to use more advanced technologies to improve performance.

Chapter 3

File System Metadata

Significant metadata is stored within the index nodes of file systems, recording key file attributes such as file size; last accessed, modified and changed times; and ownership and permission information. This metadata provides valuable insight into the nature of the data that is stored on such file systems, and is fundamental in maximising the performance of file systems and data management tools.

Previous studies have examined file system metadata from a variety of environments, such as desktops computers [2, 24, 50], national and academic high performance computing [21, 83], and file and web servers [68, 72, 77]. There are some differences in metadata from different contexts, particularly regarding file types and access patterns, but there is significant commonality in many aspects. Most notably it has been observed that most files are very small, and disks are generally filled by a small number of very large files.

This chapter presents the findings of a study into the file system metadata of file systems supporting high performance computing facilities at Airbus, a leading aerospace engineering company. The objectives for this study were to compare and contrast the engineering industry with other HPC environments, and to identify any trends and characteristics that may be used to improve data management practices. The methodology used was very similar to that of prior studies, but no snapshot-based research into the engineering design environment had previously been published.

It was found that there are many similarities between the Airbus data and the other environments analysed, but there were also some notably unique characteristics. In particular, the Airbus data exhibited substantially more pronounced clustering in the file size distribution; a key file type was identified that accounted for 64.2% of the bytes in just 0.99% of the files; and it was observed that data retention policies require files to be kept for much longer than many other disciplines.

3.1 Introduction

The engineering design process involves the production of large volumes of data, for example through computational fluid dynamics, stress analysis and laboratory experiments. These data must be stored, shared and distributed across multinational infrastructures and organisations. Advances in parallel and distributed algorithms, improved processing performance, and the growing commoditisation and increased accessibility of high performance computing (HPC) have contributed to a steady rise in data production. In contrast, the storage systems and networking infrastructure on which the data must be stored and shared may often benefit from only relatively modest advancements. As a result, storage and networking may often become bottlenecks in the overall efficiency of high performance computing systems.

In corporate engineering environments, it is clear that efficient data management is becoming increasingly critical. The effectiveness of any data management techniques depends greatly on the requirements of the parent organisation and workflow, and on the nature of the stored data. In this chapter we present the results of a study into the characteristics of file system metadata at Airbus [3], a leading aerospace engineering company. The Airbus environment examined here shares many similarities with the aerospace industry in particular, but also the wider engineering sector, including the maritime and motorsport industries where computational fluid dynamics (CFD) analysis is used extensively. Previous work has already recognised that significant variation exists in file system metadata from different environments [67], so we compare and contrast results with external data from a desktop environment [2] and a variety of other HPC systems [21].

This research was motivated by a need to make better use of existing storage resources. The key goals were to explore how the context and environment are reflected in the file system metadata, and to highlight any characteristics that could be used to improve data management practices, especially in the cleaning of unwanted or redundant data. We also aimed to find traits and features that could be used to describe the content, using only the basic file system attributes. This additional goal is the subject of a separate study into additional sources of metadata (see chapter 5), but some key observations can be made from the characteristics analysed in this chapter .

3.2 Related Work

Previous work has examined file systems from a number of different environments, using two key methodologies. Snapshot based studies record file properties at a particular point in time, allowing the analysis of index node (inode) metadata characteristics, file organisation and historic access patterns from a fixed perspective. This approach has been used in the past to study desktop computers [2, 24, 33, 50] and high performance computing (HPC) file systems [21, 83]. In contrast, trace-based studies consider requests for data access, typically providing more insights into the file system workload and performance. This methodology has also been used to study a variety of environments, including desktops and servers [12, 67, 68, 72, 77], and commercial and engineering sources [5, 41]. It has been noted that although there has been much research into the characteristics of research and desktop environments, there has been much less into large-scale corporate and industrial systems [41]. The latter has been the focus of this study, but we draw comparisons with other environments as appropriate.

The actual methodology used in file system benchmarking has also been studied, noting that the tremendous diversity in systems, optimisations and workloads necessitate multiple benchmarks [80], which can be hard to compare between different studies. There has also been work into modelling file characteristics, particularly the file size distribution [25, 28, 55], although it seems clear that each file type will have a distinct file size profile [55]. The organisation of files is also of interest [33], and despite the prevalence of hierarchical file systems, it has been noted that this may not be the best solution [70].

3.3 Industrial Context

In contrast with many of the past studies into academic and desktop computers, we examine industrial engineering data at a major aerospace engineering company. The new results presented here were extracted from a single site at Airbus, a leading aircraft manufacturer, specifically from a group of file systems supporting high performance computing (HPC) for aerodynamics. The HPC system itself has ranked highly on the TOP500 list for the past several years [79]. In addition to the aerodynamic work considered for this particular study, it is also used extensively for many other aspects of the engineering design process. The file system volumes scanned are all UNIX NFS volumes with fast network access to the main HPC systems, operating independently of the internal HPC file system from which data is copied back upon job completion. As a result of constant data production by the HPC systems, these volumes rapidly become full, whereupon data must be archived or deleted, as appropriate. The snapshot analysis was initially carried out specifically to support the development of new methods for this type of data management.

Two main groups of volumes were scanned. In both cases, additional capacity has been added continuously over the past decade in order to meet the ever-growing storage requirements. The project volumes exist to store valuable project-specific data, and are subjected to rigorous backup procedures. The projects referred to are high-level aircraft programs like the A320 and A380 aircraft, and the volumes examined contain large sets of pertinent aerodynamic data. The second group of file systems comprises a number of volumes that are not backed up, and are made available as temporary storage for disk-intensive work until more permanent storage can be provided on a project volume.

The workflow responsible for the production of the data stored on these volumes is the aerodynamic analysis of engineering components and assemblies using computational fluid dynamics (CFD). Much of the data will thus have been generated computationally by CFD applications such as elsA [59] and Tau [23]. CFD is widely used across many sectors of the engineering industry, and file system metadata from other engineering environments using HPC to perform CFD are likely to exhibit similar characteristics to those shown in the results section of this chapter. Further details of the aerodynamic analysis workflow and of the network and infrastructure of the industrial file systems studied can be found in chapter 2.

3.4 Methodology

This section describes the methodology applied through the data collection process, the analysis of the results data, the processing of external data sources for comparison, and the presentation of the combined results. It also outlines the limitations of this methodology, particularly in terms of applicability to other environments.

3.4.1 Data Extraction

We wrote a simple Python program (pDisk) to recursively travel through a given directory tree from a specified path using the Python `os.walk()` method. This program records all of the metadata associated with each file and directory within the tree. Restrictions were added to ensure that symbol links are ignored, and that mount points are not crossed, thereby avoiding and problems in repeating records. The absolute files paths and the Inode metadata for each file and directory, including file sizes and timestamps, are saved as a PyTables snapshot of that directory tree for a fixed point in time. There are undoubtedly many other ways to accomplish these simple tasks. However, this program was intended to be a more comprehensive internal tool for research and data administration, allowing the implementation of more complex functionality and integration with existing tools and services.

Each snapshot is a PyTables database containing a table for files and a separate table for directories, supporting basic query functionality, which is then used extensively in analysing the data. Unencrypted file paths were saved and examined as part of the analysis, but all sensitive information has been encrypted or excluded from the presented results. Custom queries can be made to the database, but the program also provides a standard set of queries for producing summary histogram data (used to produce the results of this study), and a summary of disk usage per user across all volumes. The only input required for this automated analysis is a list of regular expressions defining known file types. This list was produced manually using a combination of the raw data and detailed knowledge of the workflow and applications responsible for data production. It could be replaced with a different set of regular expressions if the program were to be applied to a different environment or workflow.

The program was executed on a Linux server acting as an NFS client, upon which all of the relevant NFS volumes were mounted. It was run once per volume for a range of volumes that serve the HPC facilities for a single UK site at Airbus, focusing on those responsible for the storage of aerodynamic data. Querying NFS volumes remotely in this way is not as fast as running the same program on a local file system, but direct access to the file systems was not available. Since the purpose was to simply capture data to be analysed off-line, this was not a problem, but performance could certainly be improved in the future by executing the scans on directly on the NFS server side.

The scans were completed over a one-week period in April 2013 at a variety of times throughout the day, typically taking about two hours to finish, although some of the larger volumes required much longer. The scan completion time was also recorded in order to compensate for any short-term time-related effects that may have been incurred. All time-related observations refer to the difference between the recorded timestamps and the scan completion time.

Aerodynamic analysis is crucial to the engineering design process, and although there are other fields that also make use of the HPC facilities, these non-aerodynamic systems were not made accessible for data research. As such, the data presented here is a representative sample of the aerodynamics systems, but only a proportion of the bigger picture.

3.4.2 External Data Sources

In addition to the newly presented Airbus file system metadata, we draw direct comparisons to two existing snapshot studies. A previous study into the file system metadata of Microsoft desktop computers [2] published a full set of raw data. This set was filtered to exclude all but the most recent year of snapshots (2004), which were analysed in the same way as the Airbus data. Furthermore, a study into HPC file system metadata characteristics at a variety of HPC sites released summary histogram data [21], which we have included where possible alongside the Airbus and Microsoft results. The file systems examined were from the Arctic Region Supercomputing Center (ARSC) [81]; the Los Alamos National Laboratory (LANL) [42]; the National Energy Research Scientific Computing Center (NERSC) [58]; the Parallel Data Laboratory (PDL) at Carnegie Mellon University [16]; the Pacific Northwest National Laboratory (PNNL) [82]; and the Pittsburgh Supercomputing Center (PSC) [17]. These HPC centres provide supercomputing facilities for a variety of scientific applications, unlike the Airbus systems examined, which focus purely on aerodynamic analysis. The external data sets have been included for comparison and contrast, and to help clarify the applicability of any conclusions drawn to environments besides industrial engineering design. All external data sets were obtained from the Storage Networking Industry Association (SNIA) IOTTA repository [76].

A summary of the file systems analysed is shown in Table 3.1, including the Airbus, Microsoft and the other HPC sites outlined previously. The columns indicate the stated usage, total number of files, and total number of bytes considered in each sample. The external data sets were grouped together by usage category (i.e. scratch, projects) for each discrete site where multiple snapshots were available. Although the Airbus data are generally subjected to unusually long retention requirements, the volumes scanned are active volumes rather than archive storage, hence the comparisons have been made with scratch and project data sets rather than with archival systems.

TABLE 3.1: Summary of analysed data sets, including Airbus, Microsoft, and a variety of HPC file system snapshots from the SNIA IOTTA repository [76].

Sample Name	Usage	Files (M)	Bytes (TB)
Airbus-Projects	Projects	9.3	47.8
Airbus-NoBackup	Scratch	17.0	84.7
Microsoft	Desktops	1145.0	211.8
ARSC-Projects	Projects	6.2	32.2
LANL-Scratch	Scratch	7.4	64.3
NERSC-Projects	Projects	20.5	107.5
PDL-Scratch	Scratch	19.4	5.6
PNNL-Scratch	Scratch	2.2	22.5
PSC-Scratch	Scratch	2.5	36.0

3.4.3 Data Analysis and Presentation

The analysis of file systems supporting this type of industrial high performance computing may be of interest to a variety of audiences, including but not limited to:

- Developers of file systems, especially those targeting industry and high performance computing.
- Software developers of system utilities, such as backup and cleaning tools.
- Managers responsible for storage capacity planning.

One of the key goals of this work was to compare and contrast the new results with those from existing studies. Efforts were made to carry out the extraction and analysis in a similar manner to the previous work in order to best highlight the similarities and differences between the data sets.

The data are generally represented by a mixture of histograms and cumulative distribution functions (CDFs), plotted as line graphs to maximise comparability and allow the concurrent visualisation of multiple data sets. Histograms are plotted with each bin represented as a single point (x, y), where x is the midpoint of the bin and y is the size of the bin. Since much of the data spans a large range of non-negative values, many of the graphs were plotted with a logarithmic horizontal axis, using an additional zero abscissa where necessary. Furthermore, in order to improve comprehension of the time-based graphs, the ticks on the x-axis correspond to powers-of-two (2 days, 4 days, 8 days, etc.) but the grid lines and axis labels correspond to more human-meaningful time values (1 week, 1 month, 1 year, etc.).

The Airbus data sets are boldly shown in black, since these sets and their absolute and relative characteristics are the primary focus of this thesis. The Microsoft trends are similarly bold, but the remaining HPC results are much lighter, individually distinguishable by markers. This approach emphasises the similarities and differences between the Airbus distributions and the others in general, for which detailed individual analysis has already been completed.

3.4.4 Limitations

The Airbus data presented in this study are from storage volumes serving HPC facilities in a large engineering organisation. The workloads responsible for data generation in this context are highly repetitive, and the data characteristics are somewhat domain specific. As previously explained (see section 3.3 for the industrial context), computational fluid dynamics (CFD) analysis (the dominant workload examined here) is widely used in many engineering sectors. Although general observations may be applicable to other environments, the specific conclusions drawn about these data sets may only be applicable to similar engineering environments.

Another key limitation is the elapsed time between studies. Metadata characteristics are constantly changing [2], and it has been observed that static workloads can be considered unrealistic after only a few years [80]. It follows that direct comparisons between different studies must be treated with caution. However, although several years have elapsed between this study and those used for comparison, there are still some meaningful observations that can be made.

3.5 Results and Discussion

This section considers the results obtained from the scans of Airbus file systems, and where possible explains the underlying reasons behind the trends and distributions shown. It also provides comparisons to other data sources, in order to consider the new data in the wider context of existing file system data studies.

3.5.1 File Size

Figure 3.1 shows the histograms and CDFs for the distribution of file count by file size. As observed by many other file system studies, it is very clear that most files are small. All of the samples except for LANL-Scratch and PSC-Scratch show at least 95% of files to be smaller than 8 MB, and all cases show at least 99% to be smaller than 128 MB. The Microsoft and PDL-Scratch samples generally exhibit the highest proportion of very small files, with 90% being smaller than 128 KB, while the Airbus samples appear to be fairly typical of HPC data.

The histogram in particular shows that the Microsoft sample is very smooth, but all of the HPC results demonstrate significant clustering around certain file sizes. Although the peaks vary in location between samples, this is evidence of how the workload for each HPC environment affects the file system metadata distributions. In cases where the same workflow is run repeatedly over an extended period of time, the characteristics of the files generated by this workflow will become progressively more pronounced, as the same sorts of files will typically be produced for each run.

For example, in the Airbus case we know that the data was generated computationally by CFD applications running an aerodynamic analysis workflow. It follows that the peaks correspond to different types of files associated with the CFD workflow. These include the input geometry; the mesh representing the geometry in the flow field; the 3D flow solution; the output summary files from post-processing, as well as any chosen visualisations; any scripts used to execute the various stages of the workflow; and large numbers of log files that are left behind after each stage. The peaks for other HPC samples presumably correspond to similar characteristics of their respective workflows and environments.

In contrast, the Microsoft sample does not exhibit these peaks. This is partly due to the variety of potential uses and workloads for desktop computers, but also due to the sheer size of the sample. These factors mask any such variation that might be present in smaller subsets, thus hiding any peaks and yielding the smooth distribution shown.

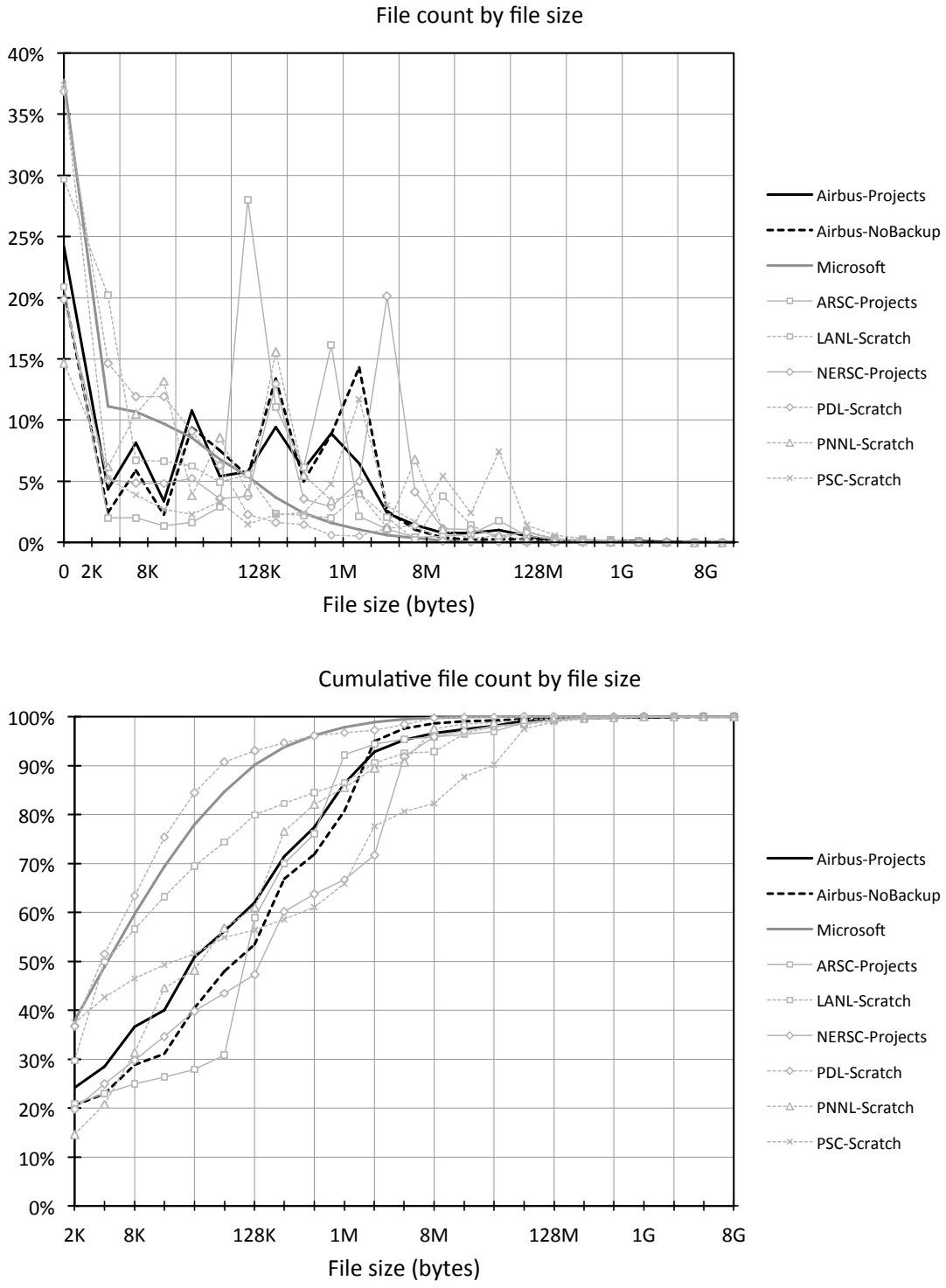

FIGURE 3.1: Histograms (top) and CDFs (bottom) of file count by file size.

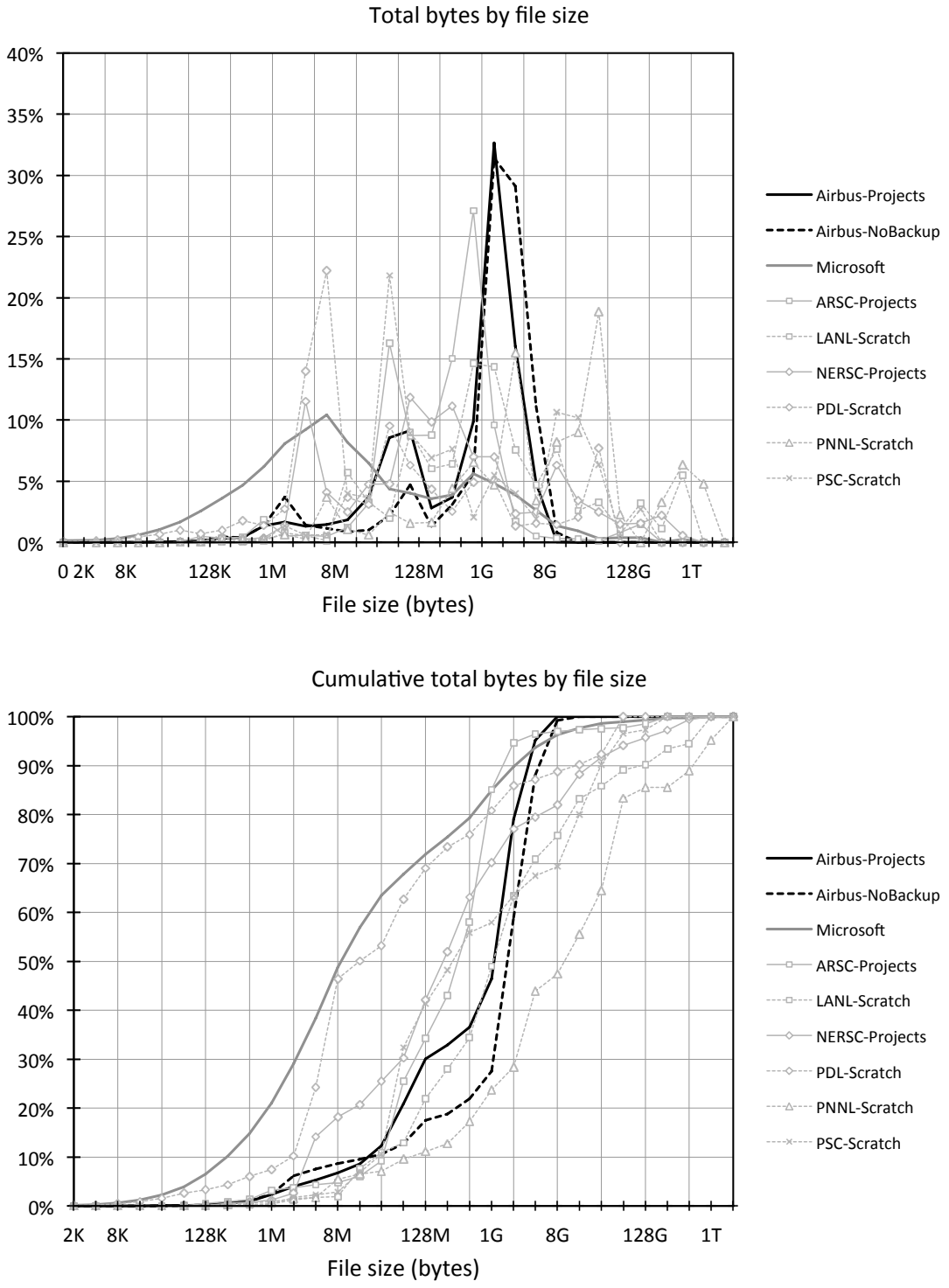

FIGURE 3.2: Histograms (top) and CDFs (bottom) of total bytes by file size.

The clustering of files around certain file sizes is also very clear in figure 3.2, which shows the distribution of total bytes by file size. However, the prominent peaks are different to those in figure 3.1, and correspond to file sizes responsible for the greatest usage of disk space, rather than just commonly occurring file sizes. The Airbus samples exhibit the greatest proportion of total bytes appearing in the narrowest range of file sizes, with the Airbus-NoBackup sample showing nearly 75% of all bytes contained in files between 1 GB and 8 GB. Although the range of Airbus file sizes falls within the bounds of the other HPC results, this is a particularly strong cluster around a very small range of files sizes. This indicates that the CFD workload responsible for data generation typically produces output files of similar sizes. The actual files in this case correspond to the 3D solution files generated through CFD analysis, as explained in chapter 2.

Compared with some of the other HPC results, the large files in the Airbus samples are relatively small. In terms of the occupied disk space, the PNNL-Scratch in particular shows a significant proportion of exceptionally large files, with 35% of the space occupied by files larger than 32 GB. In contrast, the disk space in the Microsoft sample is mostly occupied by relatively small files, with over 70% of the space accounted for by files smaller than 128 MB. The latter will be most affected by the elapsed time between studies, but even accounting for evolution with time, it is clear that there is a wide range of trends corresponding to different environments and workloads. This is clearly evident in figure 3.2, as the disk space is strongly affected by very large files.

3.5.2 Timestamps

The extracted metadata also included three timestamps: “atime”, “mtime” and “ctime”. Unfortunately, the handling of these timestamps is dependent on the operating system, thus the results must be treated with caution. It is also possible for attributes to be overridden manually, typically for application-specific purposes. The results presented here are given relative to the data snapshot scan time. Since files can still be accessed after the scan execution, small negative values are included in the first bin (0 - 2 days). For easier comprehension over a wide range of values, the tick marks on the x-axis correspond to powers-of-two, while the grid lines and axis labels are placed at more human-meaningful locations.

The “atime” attribute refers to the time a file was last accessed, so this is typically the most recent of the three timestamps. However, it is common practice, and certainly the case for the Airbus volumes, to use the “relatime” mount option in order to improve performance. This option means that “atime” is only updated if more than a set time, usually one day, has elapsed since the last recorded access time, and avoids each read operations having to be accompanied by a write operation to update the “atime” value. As such, the resolution of all of the time-based results is limited to two days, continuing in powers-of-two thereafter.

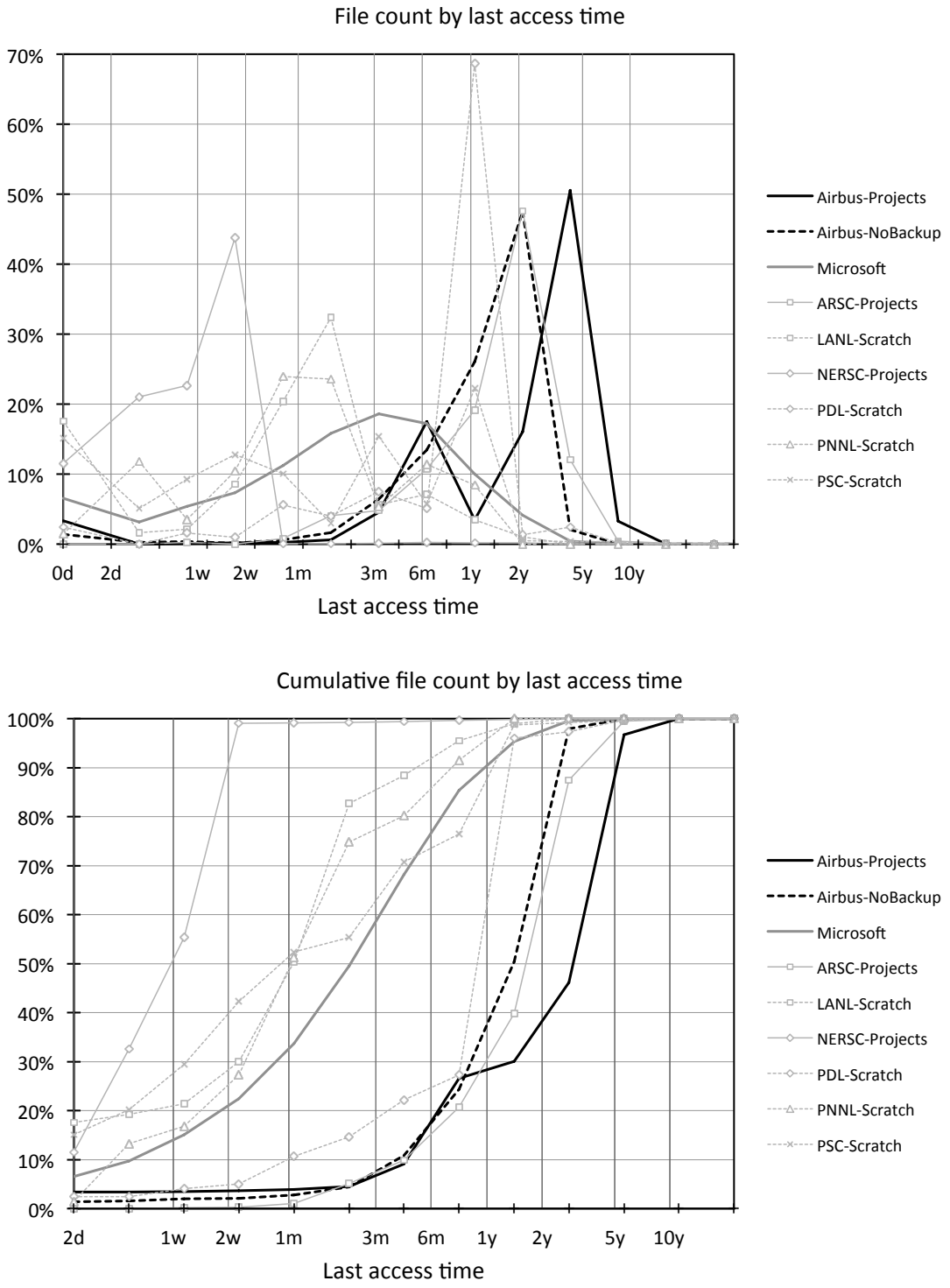

FIGURE 3.3: Histograms (top) and CDFs (bottom) of file count by access time. Note that in order to improve comprehension over the large range of time values, the x-ticks correspond to powers-of-two (2 days, 4 days, 8 days, etc.) but the grid lines and axis labels correspond to more human-meaningful times (1 week, 1 month, 1 year, etc.).

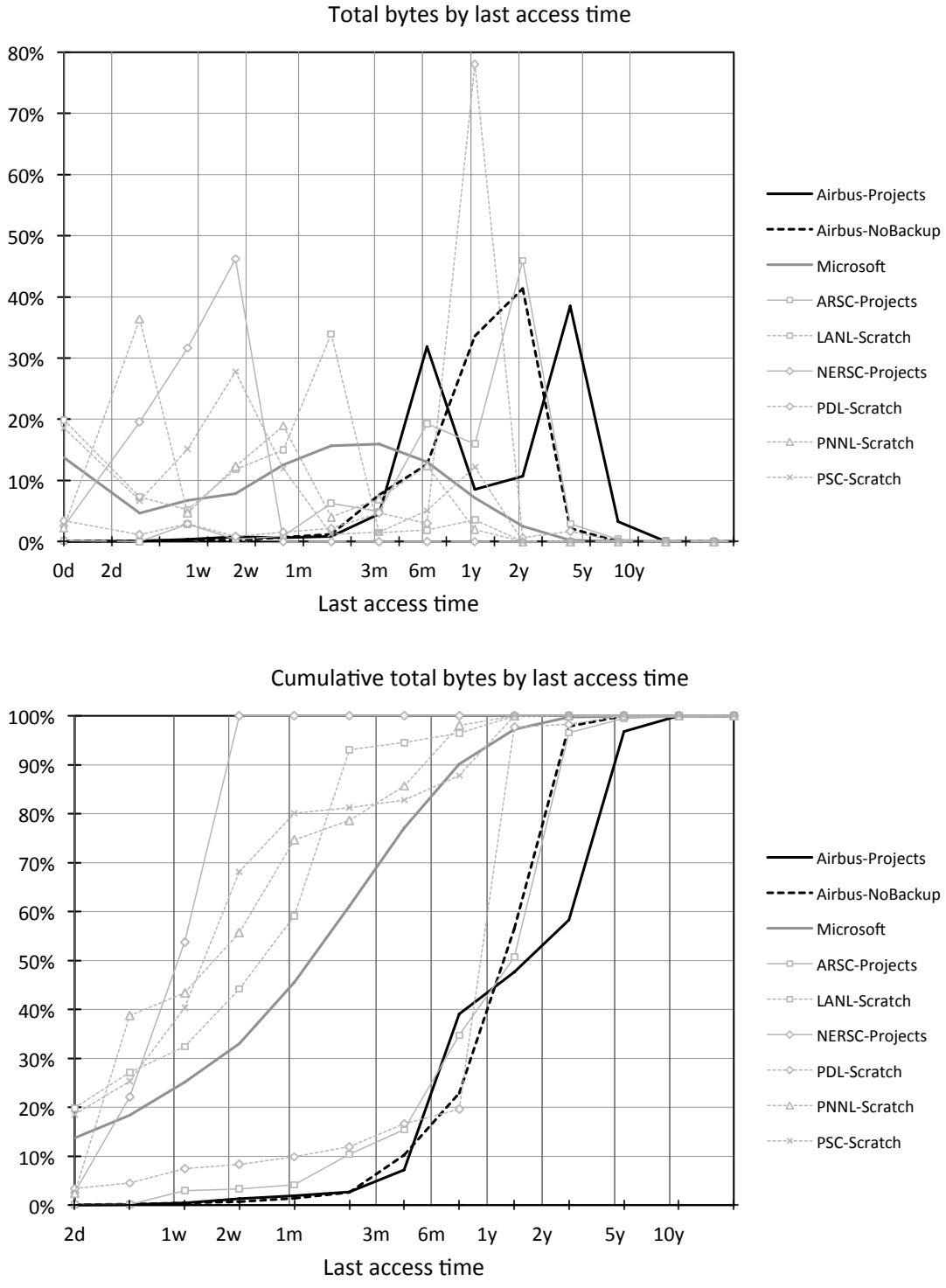

FIGURE 3.4: Histograms (top) and CDFs (bottom) of total bytes by access time. See the caption of figure 3.3 for an explanation of the x-axis ticks and grid lines.

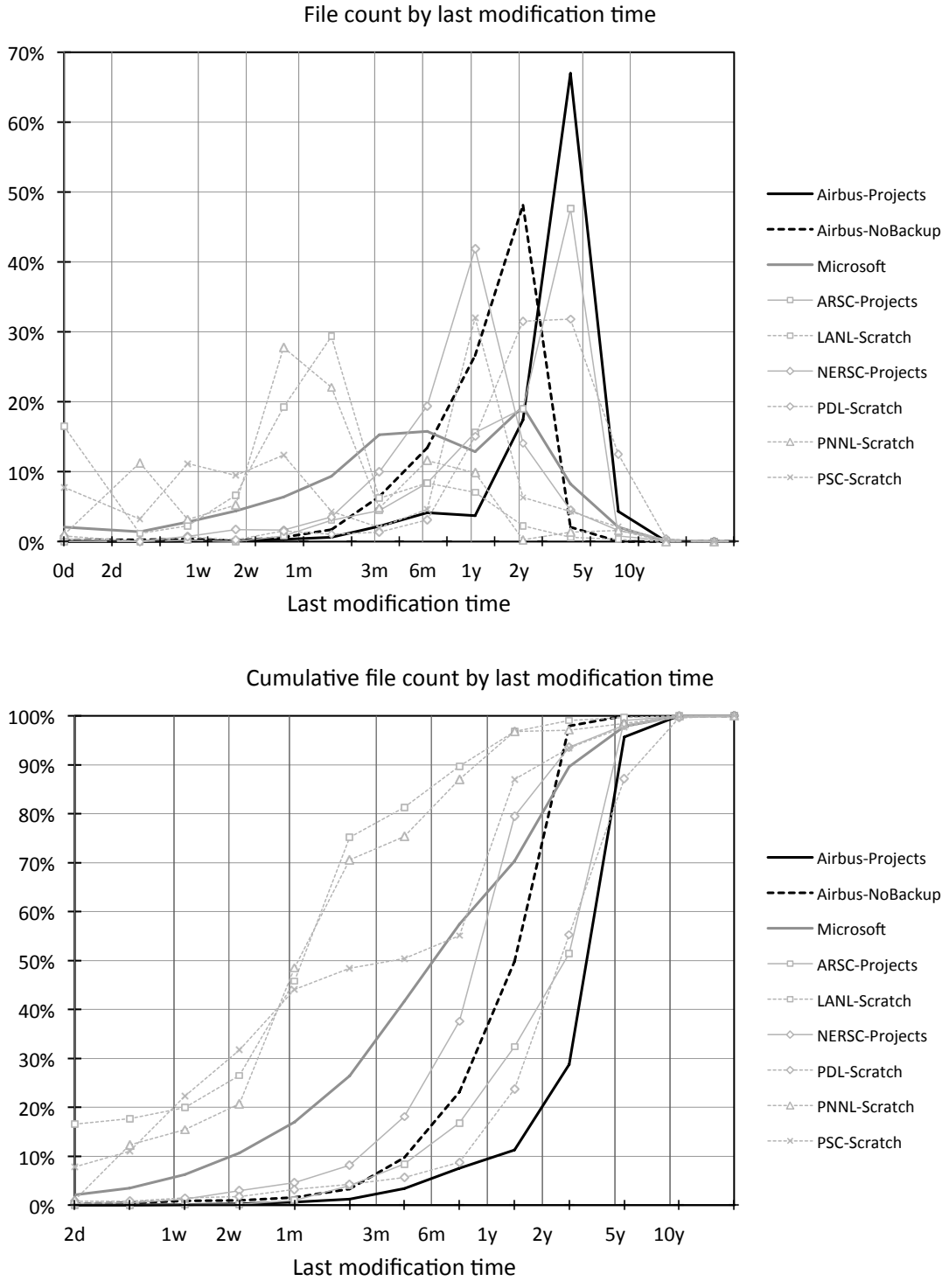

FIGURE 3.5: Histograms (top) and CDFs (bottom) of file count by modification time. See the caption of figure 3.3 for an explanation of the x-axis ticks and grid lines

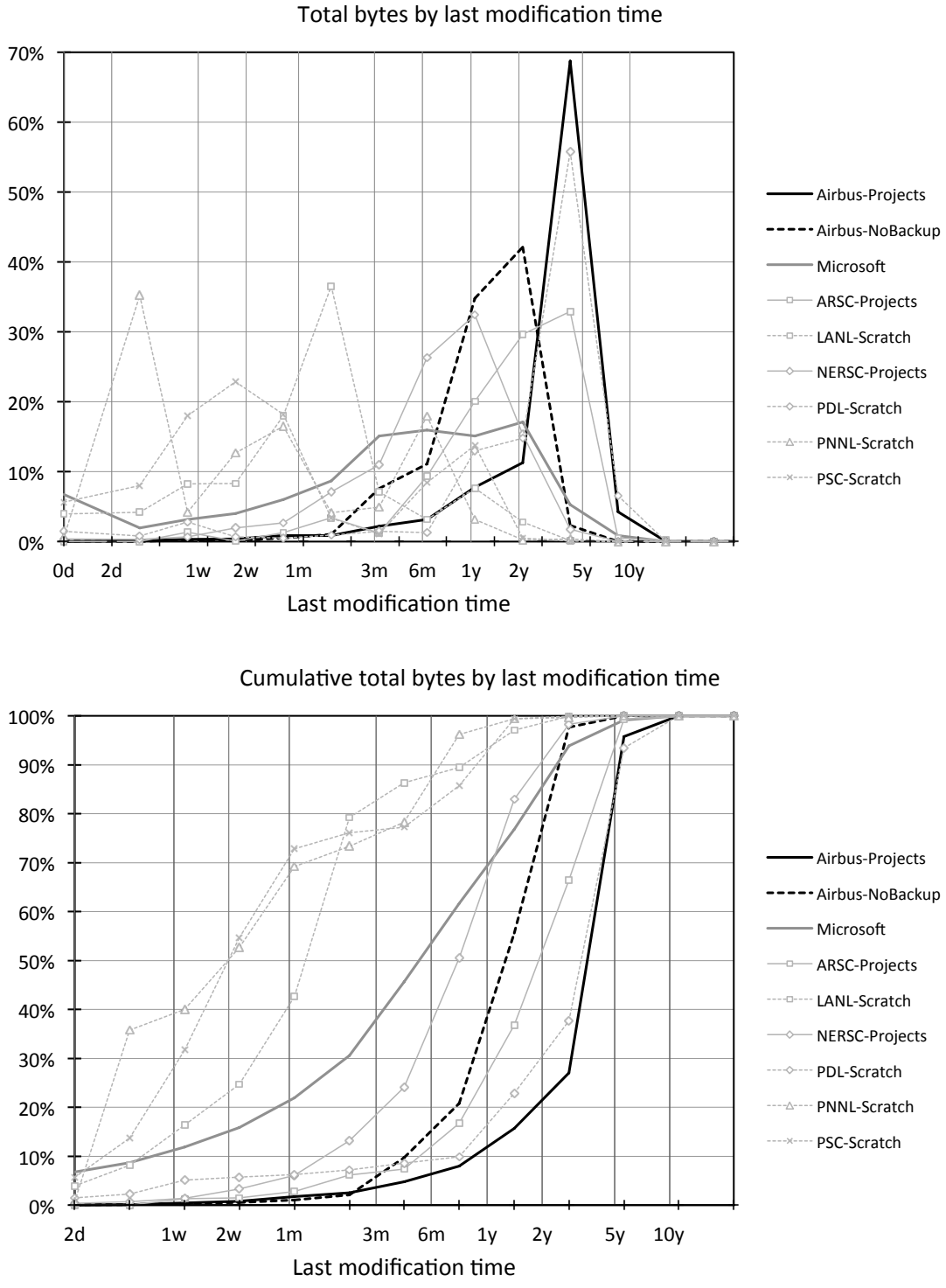

FIGURE 3.6: Histograms (top) and CDFs (bottom) of total bytes by modification time
See the caption of figure 3.3 for an explanation of the x-axis ticks and grid lines.

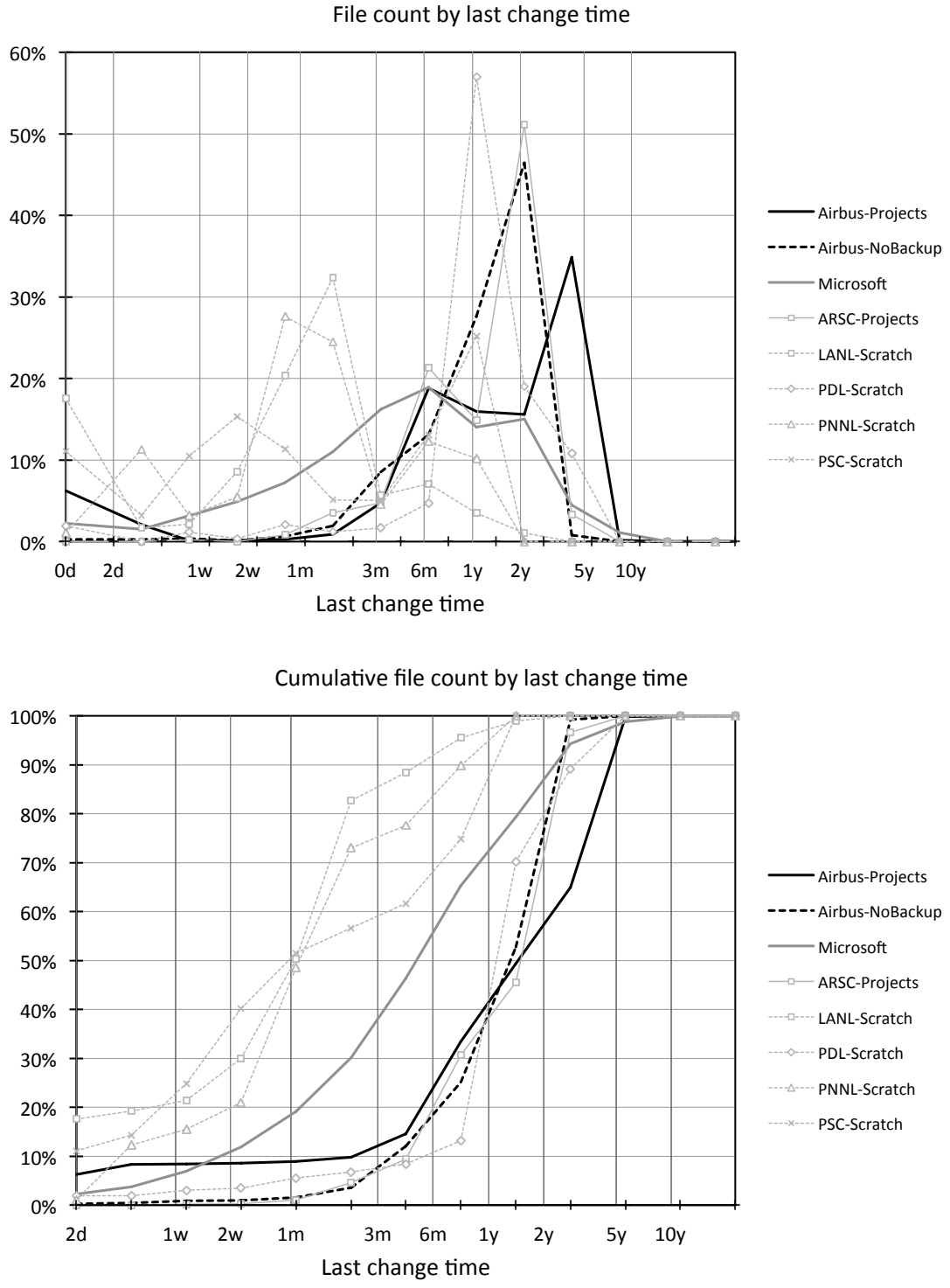

FIGURE 3.7: Histograms (top) and CDFs (bottom) of file count by change time See the caption of figure 3.3 for an explanation of the x-axis ticks and grid lines.

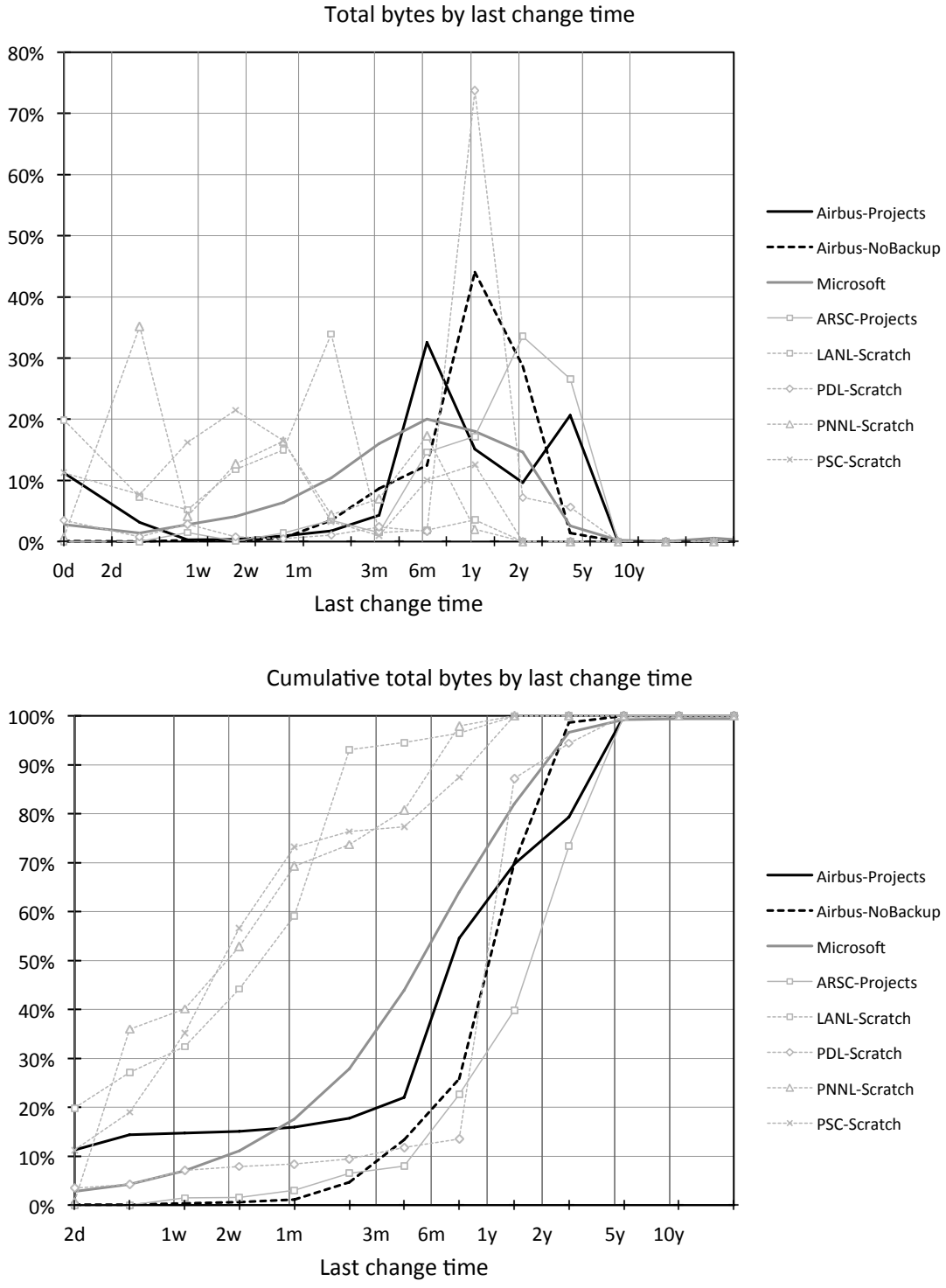

FIGURE 3.8: Histograms (top) and CDFs (bottom) of total bytes by change time See the caption of figure 3.3 for an explanation of the x-axis ticks and grid lines.

Figures 3.3 and 3.4 show the distributions of file count and total bytes by access time. Perhaps most striking is the exceptionally long time for which Airbus files remain unaccessed, far greater than most of the other data sets. Aircraft design data are subject to particularly long retention needs, and the complexity of aviation certification can make it difficult to separate data that must be archived indefinitely and data that could be deleted. For these reasons, the lack of long-term access is not particularly surprising, but it is interesting to note the difference between these results and those from other HPC environments.

Another observation that can be made of the results is that the Airbus No-Backup data generally appears to have been accessed more recently than the Airbus-Projects data. Since the Airbus-NoBackup volumes exist only as short-term storage, this is again an expected result. Interestingly, this is less evident in the distribution total bytes by access time (right), but in both cases it is clear that even short-term storage appears to contain quite old data. This indicates that it may be appropriate to provide users with updated practical guidance for the use of these file systems.

Similarly, figures 3.5 and 3.6 show the distributions of file count and total bytes by last modification time, "mtime". The Airbus results here are more similar to the Microsoft and other HPC trends, although the Airbus-Projects trend in particular remains at the edge of the range. This would indicate that although read access at Airbus differs fairly substantially from the other environments, write access is rather more similar.

This view is further reinforced by figures 3.7 and 3.8, which show the distributions of file count and total bytes by "ctime". In UNIX systems, this corresponds to the last time the metadata for a file was changed, but in Microsoft Windows systems, this field is instead used to store the creation time. As such, these figures must be treated with particular caution, especially when comparing with the Microsoft set.

The Airbus results are more similar to other data sets here than in the "atime" or the "mtime" distributions. Again, this suggests that although the access trend is quite different at Airbus, data generation and modification are much more similar to other contexts. However, the trends appear to fall into two different groups, particularly if the Microsoft set is excluded: short-term usage where many of the files and bytes are less than a month old; and long term usage where relatively few files and bytes are younger than six months. Interestingly, the former consists only of scratch systems, although the PDL-Scratch and Airbus-NoBackup sets fall into the latter category. This shows that although scratch systems often contain primarily newer data, the true usage varies greatly with the context.

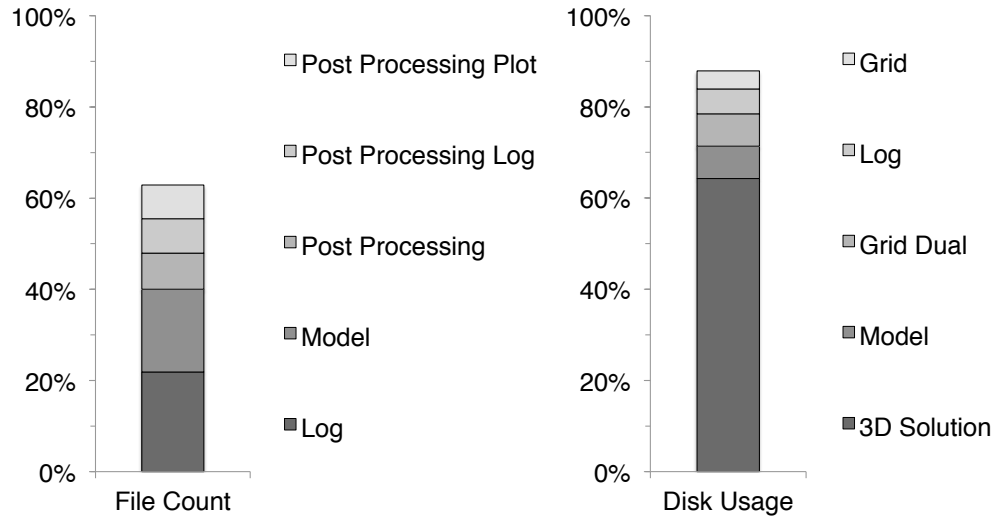

FIGURE 3.9: Stacked bar graphs of file count (left) and total bytes (right) of commonly occurring file types.

3.5.3 File Type

In many file systems, such as the desktop file systems considered in the Microsoft data set [2], file types can be easily identified by a simple extension to the file name, such as ".docx", ".xlsx" and ".jpg". Although the use of such suffixes is very widely used, it is not typically enforced by the file system. Analysis of full file names in the Airbus sample revealed that the suffix convention was not particularly prevalent, but it was possible to identify many key file types using more sophisticated regular expressions. The complexity of the regular expressions used, and the sensitivity of the information included, has been masked with descriptive type names, but figure 3.9 shows some of the most commonly occurring file types by file count and total bytes.

Compared with the Microsoft data (not shown), the figure shows that a substantially higher proportion of the files can be accounted for in the top few file types. In total, the top five file types account for 63% of the files, compared with just 37% in the Microsoft sample [2]. These file types are mostly very small files with clear reasons for long-term retention: log files are required for traceability and audit; model files are kept for the human labour involved in their creation; and post-processing files are high in value and typically required significant compute time for creation.

Figure 3.9 also shows that the top five file types by disk usage (total bytes), account for 87.9% of the overall disk space, and that a single type (namely the 3D flow solution files) account for 64.2%. These 3D solution files only account for 0.99% of the actual files, which again demonstrates the overwhelming dominance of these files in terms of the total bytes. Interestingly, these results are similar to those demonstrated previously, where simulation files accounted for an average of 1.22% of the files and 59.9% of the

bytes [33]. Moreover, many of these files are also quite old (not shown), and could perhaps be considered for compression, archival or even deletion.

The characteristics of the 3D solution files can also be observed in the file size distributions shown previously. These solution files are actually the cause of the peak in the distribution of total bytes by file size (figure 3.2). In the same way, the distribution of file count distribution in figure 3.1 highlights how files of this size account for only a tiny proportion of all files.

Perhaps most significantly, the 3D solution files are only an intermediate result in the CFD workflow. As the precise results describing the flow of fluid over a given geometry in 3D space, they are understandable large, but these files are created either to visualise the flow or to resolve the lift, drag and moment coefficients for the forces acting on the object. In either case, the solution file will be subjected to some type of post-processing in order to obtain the desired final output, thereby generating many of the other file types identified in figure 3.9. Once the necessary post-processing has been completed, the retention policy for the parent solution file can be reconsidered. There will undoubtedly be cases where permanent retention for certification and accountability will be required. However, there will also be many cases where the design moves in a different direction, and the solution file could be archived, compressed or deleted.

3.5.4 Directories

Figures 3.10 and 3.11 show the distributions of directory count by the number of immediately contained files (i.e. excluding those in subdirectories), and the total immediately contained bytes. Unfortunately, directory data was not available for the other HPC sites, which included subdirectories in the equivalent results, and are thus incomparable. The exclusion of subdirectories in this way allows the examination of the storage of groupings of files at a finer granularity than would be possible. This study also considers the distributions of files, total bytes, and directories by namespace depth (see subsection 3.5.5), providing a further way to view the characteristics of directories. Another important characteristic that must be understood before interpreting these results is the impact of excluding subdirectories from the results. In particular, the zero bin means that there were no files directly contained, and although such a directory may indeed be empty of files and bytes, it could still contain any number of subdirectories.

The Microsoft data set shows a smooth trend of decay, and a higher proportion of directories containing no files. In contrast, the Airbus set shows a distinct peak at about 32 files, before decaying similarly. As discussed previously, the size of the Microsoft data set and the general-purpose nature of desktop computers masks any specific workload or usage characteristics. However, the clear peak in the Airbus set corresponds to the typical grouping of CFD results, again showing an effect of the workload on the data.

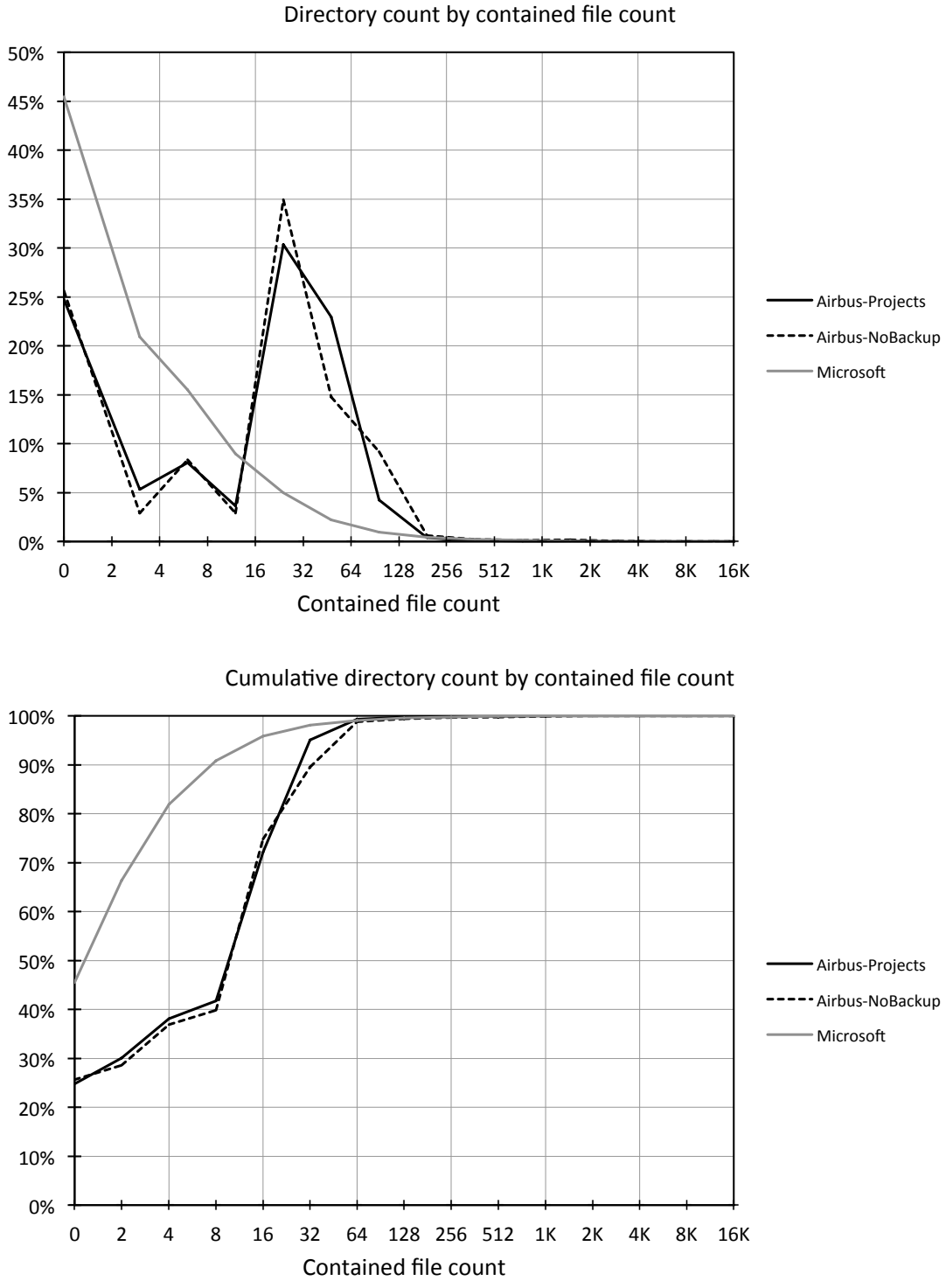

FIGURE 3.10: Histograms (top) and CDFs (bottom) of directory count by contained files.

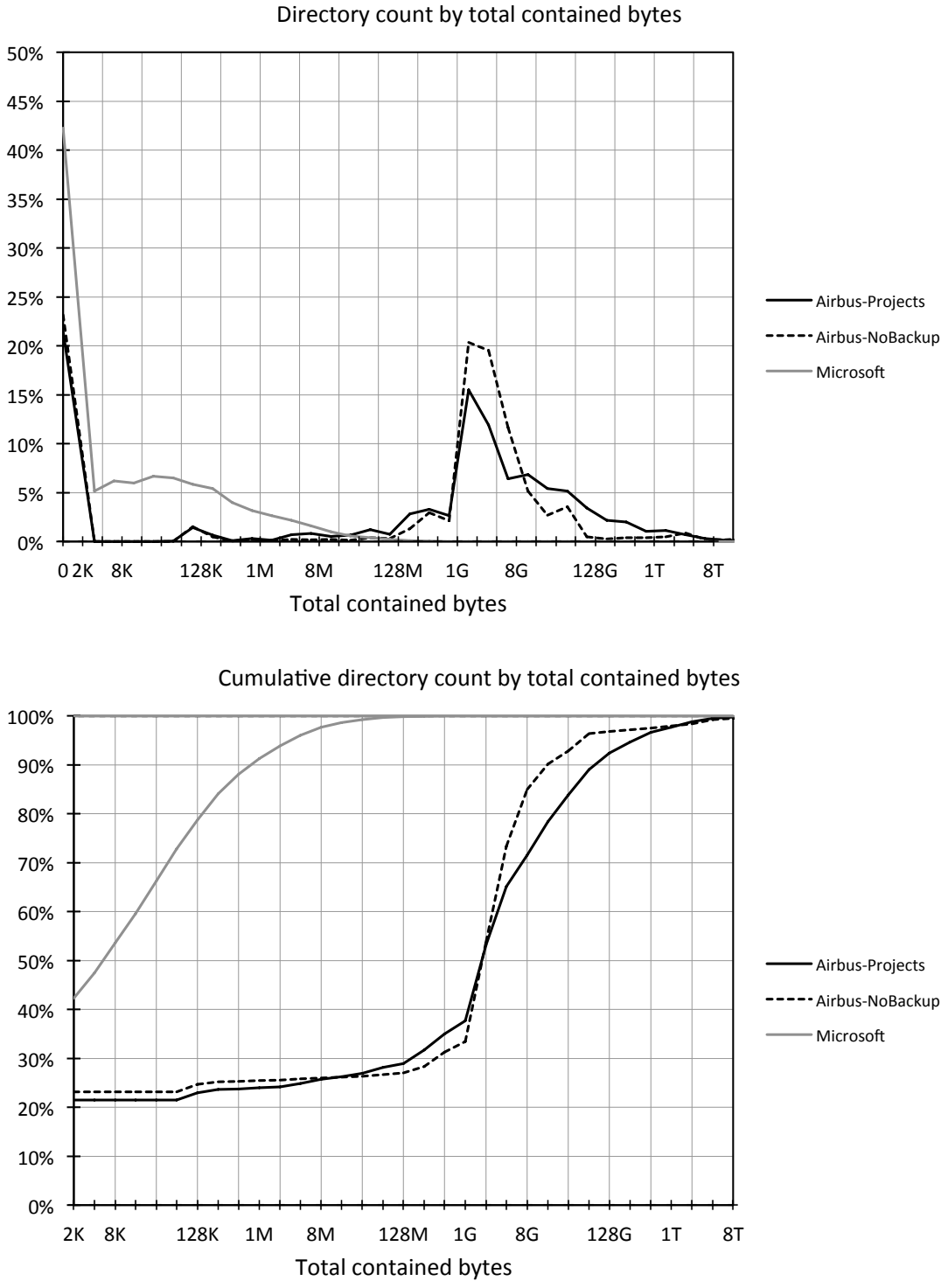

FIGURE 3.11: Histograms (top) and CDFs (bottom) of directory count by contained bytes.

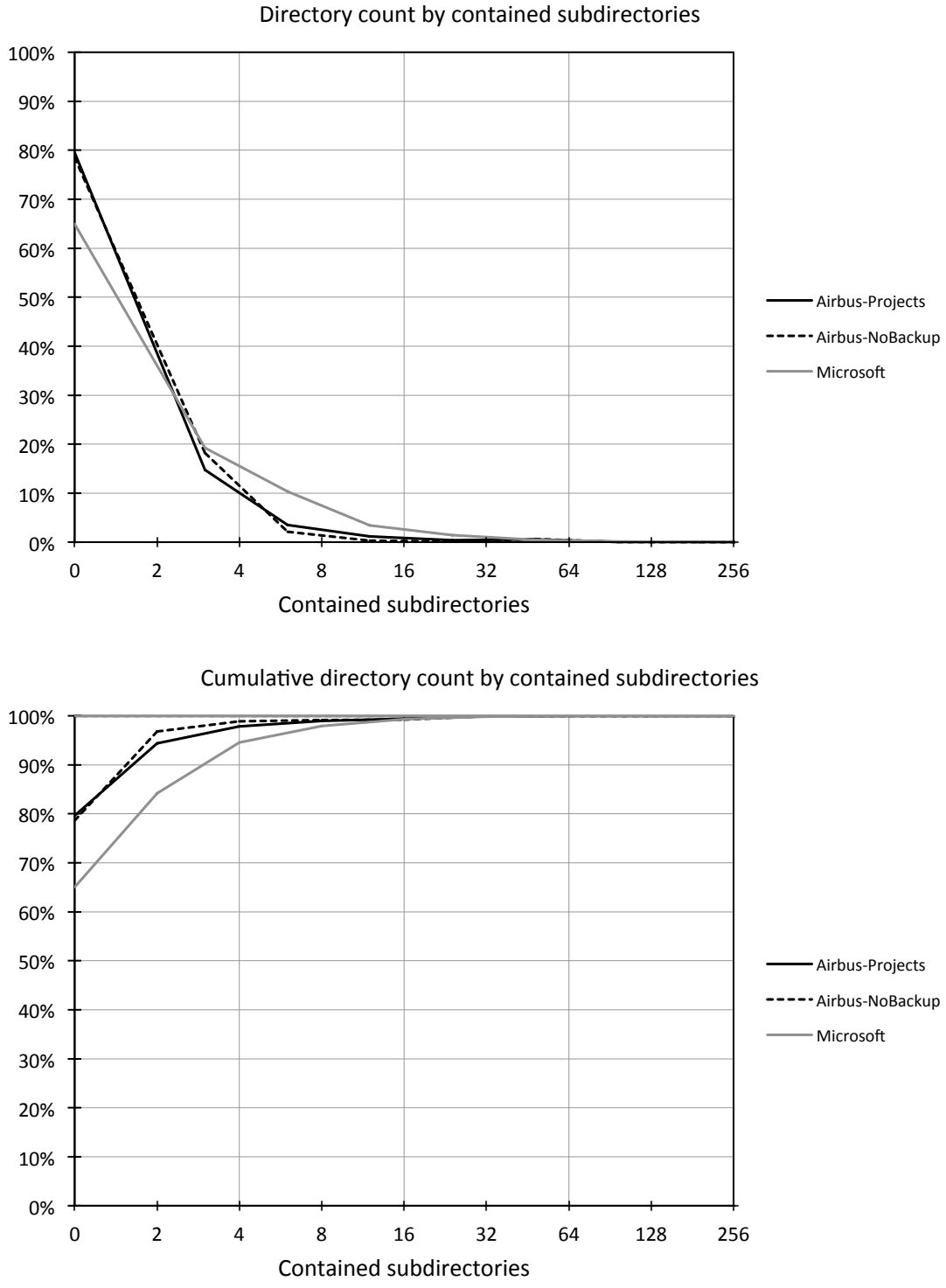

FIGURE 3.12: Histograms (top) and CDFs (bottom) of directory count by contained subdirectories.

The distribution of directory count by total contained bytes (excluding subdirectories) shows that the Airbus results are heavily influenced by the file size distribution shown in figure 3.2. The Airbus results show some directories containing very few bytes, but a peak occurring between 1 GB and 8 GB, corresponding to the asserted range of CFD flow solution sizes in subsection 3.5.1. The implication here is that most bytes are stored in directories containing one or more 3D flow solutions, which makes sense given the high proportion of total bytes in these files.

The final aspect of directory content that can be examined is the number of subdirectories per directory. Figure 3.12 shows the directory count by the number of contained subdirectories. Most interesting here is the similarity between the Microsoft and Airbus results, which demonstrates the way in which humans tend to organise data into subdirectories. In the Microsoft case this will have been in a manual fashion, whereas the Airbus directories will have been created and populated computationally. However, the process of computational directory creation will still have been set up by a human. It follows that there may be an underlying human behaviour in the way that hierarchical structures are designed. The actual psychology of this human behaviour is beyond the scope of this work, but it would certainly be interesting to investigate this trait further.

The similarity in data organisation with the Microsoft sample also indicates that general-purpose file systems are still adequate for this type of usage. In contrast, file systems such as GIGA+ support millions of file creation requests per directory, per second [65]. This study does not consider real time requests, but such a level of performance is much more applicable to direct HPC usage, rather than the supporting role played by the Airbus file systems examined in this study.

3.5.5 Namespace Depth

Another way to view the organisation of data is to consider the depth of files, bytes and directories in the namespace or hierarchy. Figures 3.13 and 3.14 show the distributions of files and bytes through the namespace hierarchy, zero being the root level. Considering the file distributions first, both of the Airbus samples show most of the files stored at a much greater depth than the Microsoft sample. This is because the Microsoft sample comprises a large number of individual desktop computers, each with their own hierarchy, but the Airbus samples all exist concurrently in the same network namespace. It follows that the smaller individual volumes in the Microsoft sample are relatively shallow in depth, but the larger individual volumes in the Airbus samples are deeper. As a result of the similar way in which humans tend to organise data, as already demonstrated in subsection 3.5.4, this shows that more layers are required to handle more data.

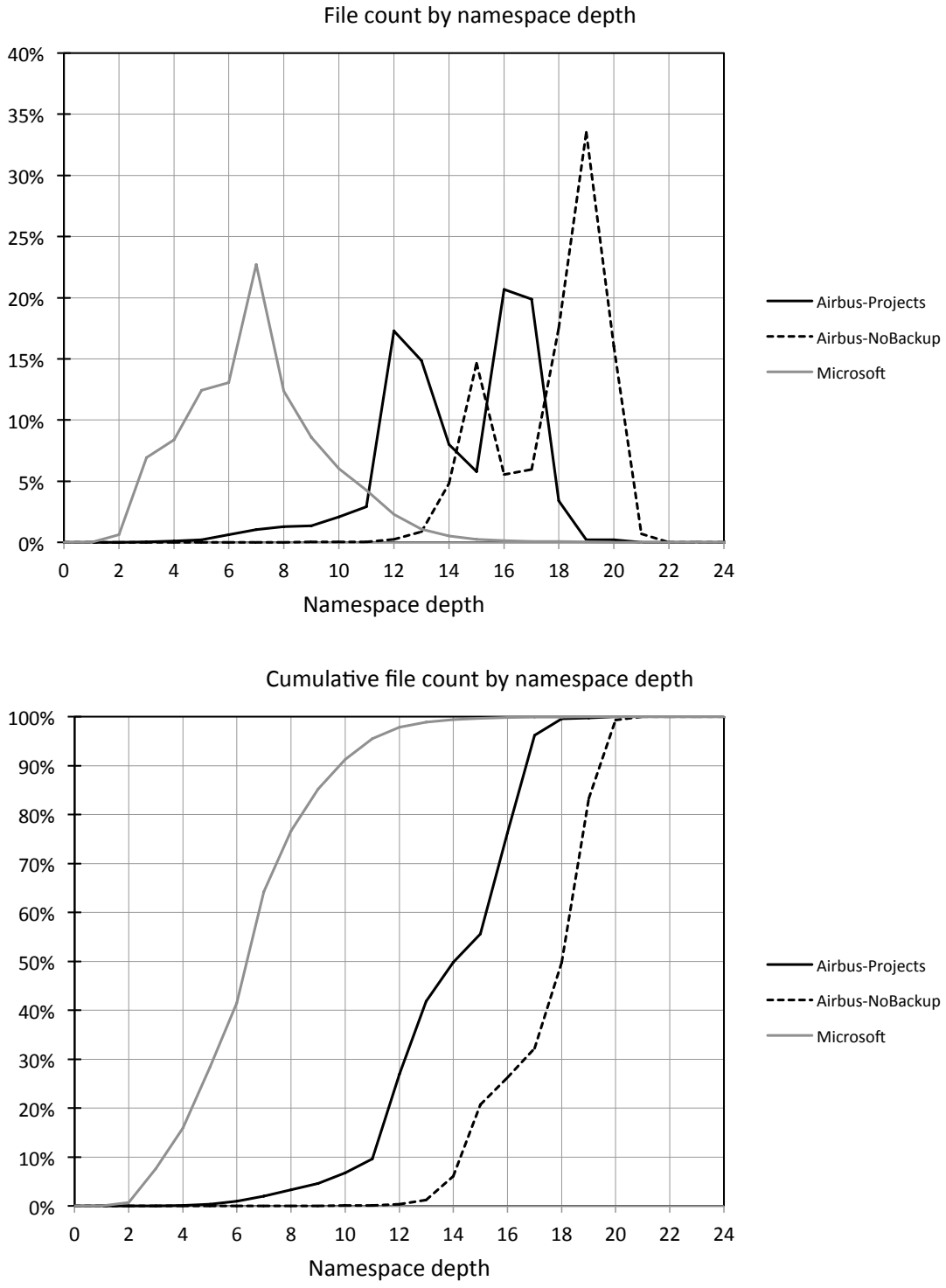

FIGURE 3.13: Histograms (top) and CDFs (bottom) of file count by namespace depth.

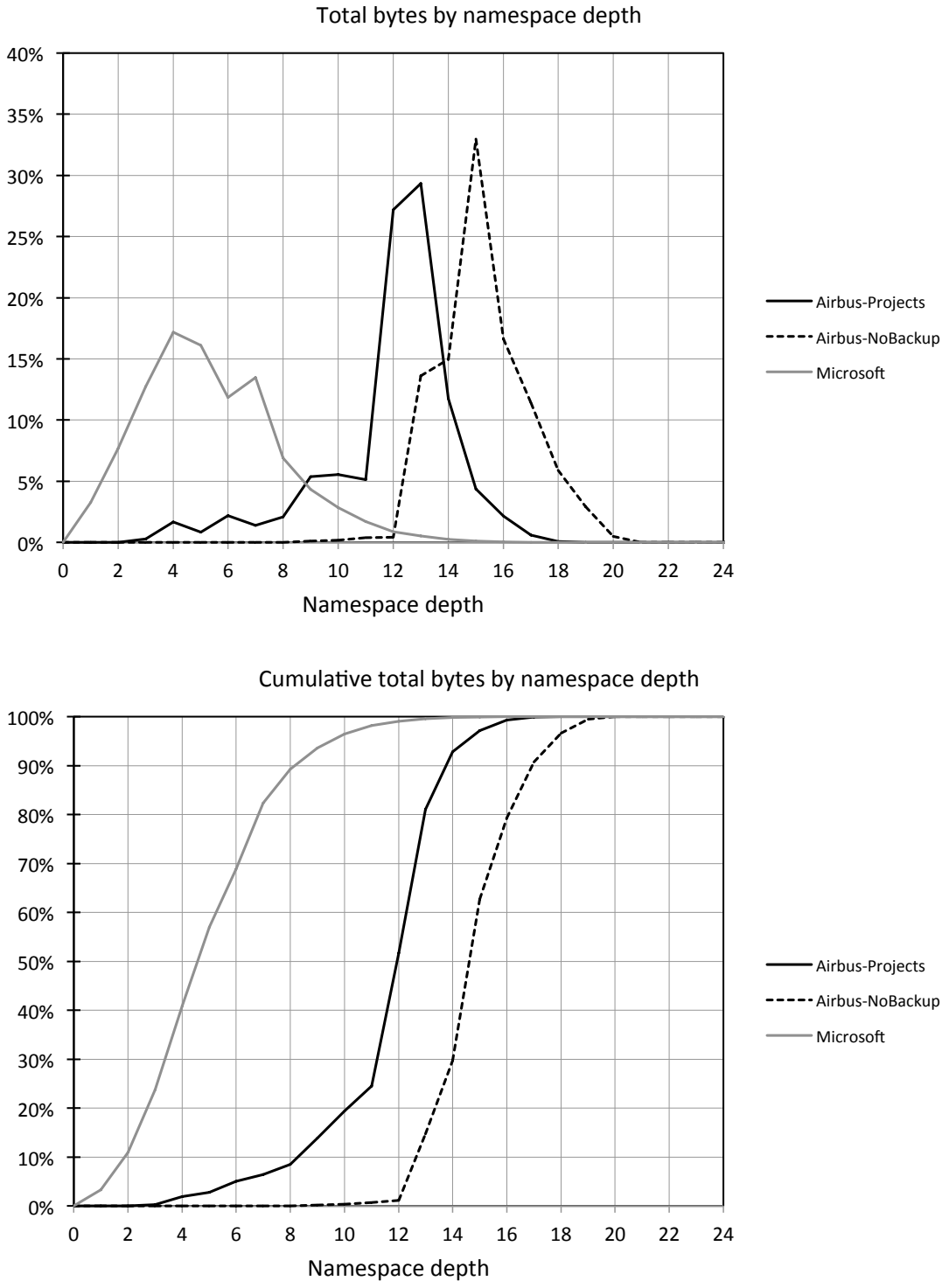

FIGURE 3.14: Histograms (top) and CDFs (bottom) of total bytes by namespace depth.

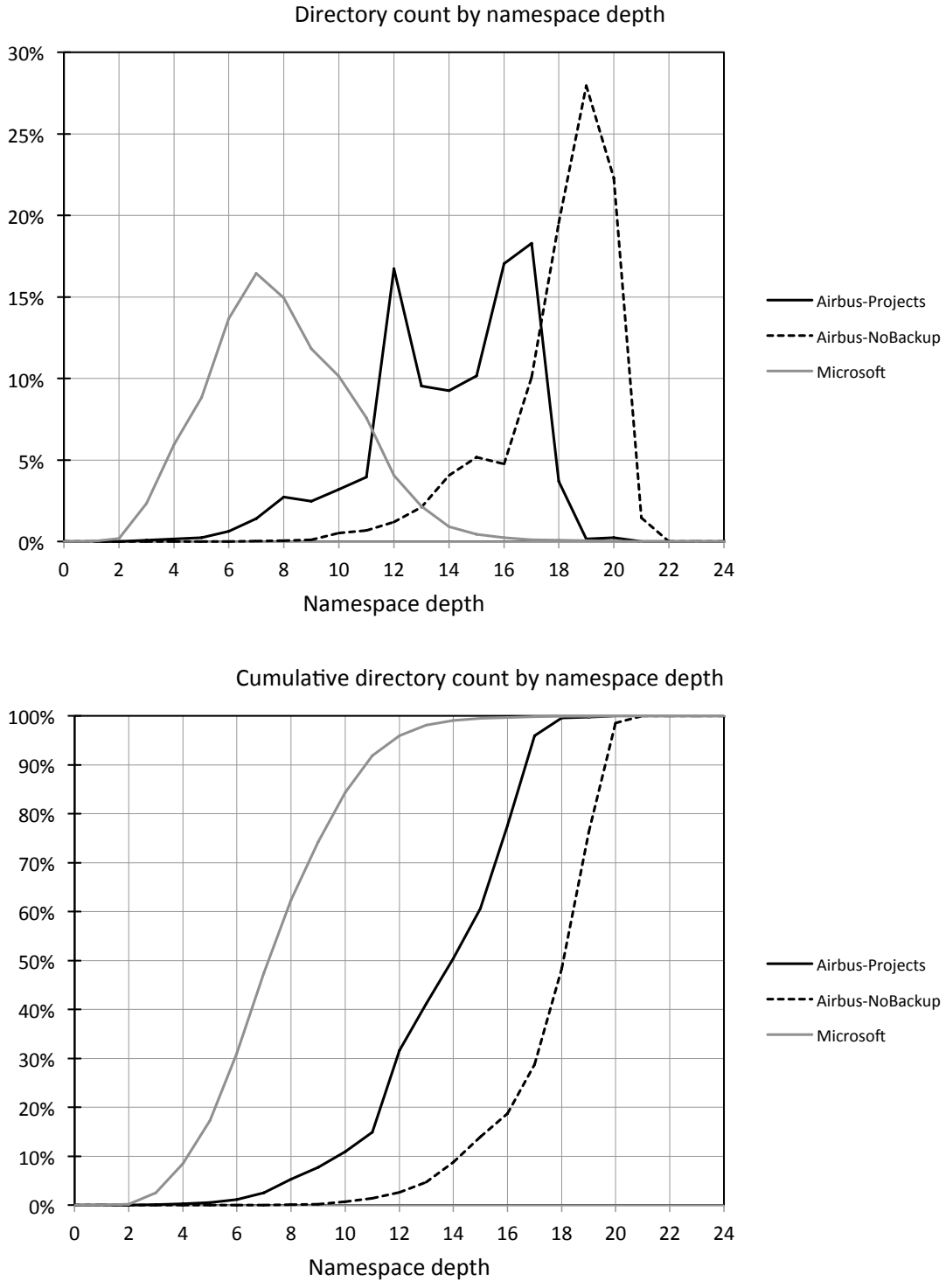

FIGURE 3.15: Histograms (top) and CDFs (bottom) of directory count by namespace depth.

Another observation that can be made is that while the Microsoft data forms a single peak, the Airbus samples each form two distinct peaks. In the Airbus-Projects case these are similar in size, although the Airbus-NoBackup case is skewed, showing more files in the deeper peak. This characteristic can be explained by considering the distribution of bytes, which shows the distribution of total bytes by namespace depth. Here, the Airbus samples both form single peaks, each in the same location as the respective shallower peaks in the file count distributions. This would suggest that larger files, namely the CFD solution files, are typically stored in the higher peak. Conversely, the smaller files, such as configuration files, documents, and other project files, are generally stored in the lower peak. A similar observation can also be made of the Microsoft sample, which shows the same characteristic, with more bytes stored higher in the namespace.

Similarly, figure 3.15 shows the distribution of directories by namespace depth. Again, the Microsoft sample shows a single peak, in line with the file count distribution, as do the Airbus distributions. However, the Airbus-NoBackup shows a much greater bias towards the deeper peak, more akin to the distribution of bytes. This an unexpected characteristic, possibly the result of computationally pre-creating all potentially required directories, rather than repeatedly checking for their individual existence throughout the workflow.

There is often a distinct corporate structure in the way that departments, teams and projects are organised, especially in larger organisations like Airbus. It makes sense that data should be organised and stored in a way that reflects the organisation of these departments, teams and projects. This can be observed in the namespace depth analysis, with the peaks in figures 3.13 and 3.15 representing the key areas in which different types of data are stored, and the levels above reflecting the corporate structure in which the data is organised. To a certain extent, the automation of data production contributes to this organisation pattern, although it has a greater impact on the lower levels in data for the same type of job is always stored in the same way.

The namespace depth results in general show that large hierarchical systems are still very much a reality, but it is becoming clearer that data can be hard to find in such systems, particularly where there are multiple criteria on which a logical hierarchy could be applied. The day-to-day search for data by original owners is still fairly straightforward, but any identification and location problems are greatly exacerbated when data is shared, exchanged or inherited, especially when users change roles or leave organisations entirely. This is one of the key challenges for data management in the engineering industry, as these cases are very common in large engineering organisations like Airbus, as mentioned previously in chapter 2

Such issues could be addressed by better understanding the meaning, purpose and relationships between data. For example, in the case of users changing roles files could be tagged with the role under which a user created them as well as their user ID, allowing the next user to take the role to inherit data more easily. Although it may be possible to infer such information from the hierarchy, depending on how the user chose to organise their files, it would be much simpler to identify files if this type of semantic metadata were available. A simple search based on this type of data may then be sufficient to overcome the challenge of users changing roles. However, given that this is a common occurrence it is certainly worth considering alternative ways to organise data that do not rely solely upon hierarchy to provide structure to file system content [70].

3.5.6 Implications for Data Cleaning and Distribution Modelling

We have demonstrated that the 3D solution file type alone accounts for 64.2% of the bytes in just 0.99% of the files, and related this to the file size distribution. Furthermore, we have identified that these files are only an intermediate stage in the CFD analysis workflow, and suggested that once any required post-processing has been completed, the retention policy could be reconsidered.

In addition, the timestamp analysis has shown that much of the data is quite old, with less than 40% of files or bytes having been accessed in the past year. When combined, these observations would suggest that substantial savings in disk space could be achieved through the archival or removal of old solution files. For example, removing 3D solution files older than 2 years from the no-backup volumes could save an estimated 16% of total disk space. This certainly merits further analysis, and is considered in greater detail in a separate study into data cleaning (see chapter 4.

Previous research has found that modelling file size distributions using different file size profiles for different file types yields substantially more accurate results [55]. The Airbus trends that we have presented here demonstrate significant clustering around key file sizes in the distributions of both the file count and total bytes with file size. Moreover, we have shown that the 3D solution file type directly corresponds to a key peak in the file size distribution, and assert that geometry models and specific log file types correspond to the other key types. Although we have not attempted to model the file size distribution, our results certainly support the view that file type profiles could yield more accurate file size models. These results will be of particular interest to those modelling distributions, generating synthetic data for benchmarking, or optimising file system designs for specific purposes.

3.6 Conclusions and Further Work

The Airbus results, although showing some similarity to previous work, demonstrate some notably unique characteristics. In particular, the file size distribution shows exceptionally strong clustering of disk usage by files at the environment-specific file size of 1 GB and 8 GB. This links to the file type analysis (figure 3.9), which showed that a key file type (namely the 3D solution files for CFD simulations) accounted for 0.99% of the files but 64.2% of the total bytes. Although numerically similar to previous work [33], it is interesting to see that this trait extends to much larger file systems, in an HPC rather than desktop environment, and when based on a much larger sample size. It was also noted that much of the data examined was quite old, with many files not having been accessed at all in the past few years. Since the 3D solution files are only intermediate files in the larger workflow, these observations lead to the conclusion that solution files could be considered for archival or deletion in order to make way for new data. Reviewing the retention policy for this particular file type could potentially yield great savings in disk space for relatively minimal human time and effort. These results will be of interest to file system designers and bench-markers, developers of system utilities such as backup and cleaning tools, and managers responsible for capacity planning, particularly those in industrial HPC environments.

In the future, these findings will be used to intelligently provide recommendations for data deletion and archival. It is also likely that a variety of important and useful information exists within the many log files already identified in this chapter, and extraction and analysis of metadata from log files may be able to further improve understanding of the data characteristics and better support the data cleaning process.

Chapter 4

Practical Data Cleaning

Due to the rapid production of data by high performance computing systems supporting the engineering design process, file system volumes rapidly become full. These volumes must then be either cleaned or extended. Although the cost of storage continues to fall, this is no reason to be overtly profligate or wasteful with the existing storage resources. Careful consideration of the longer-term retention of data on existing storage systems may provide substantial savings in disk space, increasing the service life of these storage systems and providing direct and indirect economic benefits.

The previous chapter revealed that the Airbus volumes examined contained one particular file type that accounted for 64.2% of the bytes in just 0.99% of the files. It also observed that this key file type, namely 3D solutions generated for computational fluid dynamics analysis, is only an intermediate result in the overall engineering design workflow. Although some circumstances require permanent retention of these files, there may also be many cases where files could be deleted.

This chapter follows on from the earlier snapshot work with a study of practical data cleaning at Airbus. The objectives were to develop a reusable methodology for identifying unwanted files that required minimal human input and intervention, and to apply this methodology to the Airbus-NoBackup volumes examined previously to save disk space. The methodology was based on the key observations made by the previous work. It led to the completion of two cleaning exercises, which yielded significant savings in disk space of 15.2% and 27.5%, respectively. Further analysis demonstrated a potential increase in service life of up to five years for the existing storage resources, based on the net rate of data production. It was also noted that the financial value of the savings made have already exceeded the total cost of this entire research programme.

4.1 Introduction

With ever increasing rates of data production, it is inevitable that file system volumes will become full over time. In high performance computing (HPC) environments, where data production is already very rapid, file system volumes can fill up very quickly. Full volumes present various problems, particularly disk full errors preventing data being written to disk. In HPC environments where lengthy computational tasks are commonplace, this can lead to significant resources being wasted in computing data that cannot be written to disk upon completion. Manually searching for free disk space in a large scale environment can also become a significant source of wasted time for end users.

This report details the methodology and results of two data cleaning exercises on a group of no-backup volumes within Airbus, specifically those used by the aerodynamics group in Filton, UK. These volumes are used extensively by the HPC systems, and are known to fill up very quickly. Prior to the cleaning, many of the volumes had become completely full. The purpose of the cleaning exercises was not only to create free space on the volumes, but also to develop a method for doing so which required only minimal human input and interaction. This method may be used in the future as necessary without requiring the same efforts in research and analysis, and could potentially be integrated and automated as part of a larger data management system.

4.2 Related Work

There are a number of existing tools for examining file system metadata in a manner conducive to the identification and removal of large files. Tools such as KDirStat [36] for Linux; SequoiaView [78] and WinDirStat [86] for Windows; and GrandPerspective [27], Disk Inventory X [22], and DaisyDisk [73] for Mac OS all provide a means to view disk usage statistics and clean unwanted files. However, they still require a somewhat labour-intensive approach in the selection process, which remains non-cognizant of the environment, context, and workflows responsible for the creation of the data. These tools can be highly effective in a desktop context, but are not necessarily appropriate for usage in large-scale HPC storage systems.

Previous work examined some of the characteristics of file system metadata in engineering design. In particular, it was noted that a single file type (the 3D solution files) accounted for most of the disk space but very few of the files; that this file type was only an intermediate file type in the CFD workflow; and that many of these files were quite old. It was suggested that the retention policies for this file type could be reconsidered [62]. This work extends that observation with a practical approach to data cleaning, and the results from two data cleaning exercises.

4.3 Industrial Context

The cleaning work discussed in this chapter was undertaken at a single site at Airbus, a leading aircraft manufacturer, specifically from a group of file systems supporting high performance computing (HPC) for aerodynamics. The volumes scanned are all UNIX NFS volumes with fast network access to the main HPC systems, operating independently of the internal HPC file system, from which data is copied back upon job completion. Specifically, the volumes are all part of the "no-backup" group, which exists for disk intensive work and as temporary storage until more permanent storage can be found elsewhere. Due to the constant data production these volumes rapidly become full, whereupon data must be archived or deleted as appropriate.

The workflow responsible for data generation mostly comprises the computational fluid dynamics (CFD) analysis of engineering components and assemblies under varying flow conditions. Given either an entirely new or modified geometry file, a mesh is generated to represent the geometry during the CFD analysis, which is used to create 3D solutions for the flow conditions of interest. These solution files are then post-processed to resolve the lift and drag forces acting on the object and sometimes to visualise the fluid flow, in order to inform future design decisions.

Further details of the engineering design process and the CFD workflow can be found in chapter 2. As mentioned previously, CFD in particular is widely used in many areas of the engineering industry, including the aerospace, motorsport and maritime sectors. As such, these environments may exhibit similar file system metadata characteristics and data cleaning potential to the results shown here.

In the context of industrial engineering design, it is important for a data cleaning utility to be aware of the workflow responsible for data generation. This is because the sorts of files that can be removed depend on the sorts of files that are created by the workflow, and the corresponding need to retain each of these types. It is also important to recognise that the outputs of one job may become the inputs for another. For example, CFD analysis outputs flow solutions which are the input for various optional post-processing stages. It is thus important to allow sufficient time for decisions to be made regarding any optional aspects of the workflow, rather than just removing intermediate files immediately once a the basic workflow has been completed. In addition, a cleaning tool must be sympathetic to future processes such as data audit, which may require collection and presentation of *all* data directly associated with a final design or configuration. It follows that simply purging *all* data older than a given age threshold is not an appropriate strategy for the type of data considered here.

4.4 Methodology

This section describes the development of the data cleaning method; the nature and reasoning behind the exceptions made to the rules used; and the details of the actual execution. Some of the file system metadata distributions used here have been shown previously, but the focus here is on their utilisation in the development of a file system cleaning methodology. Previ14ous work on file system metadata revealed some interesting trends and patterns in the distributions of basic file properties. In particular, the file size, file age, and file type distributions all showed characteristics with useful implications for the development of a rule-based data cleaning system [62]. This section revisits these earlier observations in order to develop a file system cleaning methodology. The process has been derived and applied with a view to cleaning file systems by deleted files, but could equally be used as a selection method for compression or archival in cases where deletion would be inappropriate.

4.4.1 Data Extraction and Analysis

The data analysis behind this work into data cleaning made use of the same internal Airbus Python program used to conduct the previous work into file system metadata analysis (pDisk) [62]. This program provides a snapshot of a file system for a fixed point in time, allowing detailed analysis to be conducted off-line, thereby minimising any disruption to the actual file system caused by the scans. New snapshots were taken for all of the no-backup volumes, using the same data extraction method as the file system metadata study. The data analysis presented in this chapter was conducted using a new set of queries that were made to the resulting PyTables database.

4.4.2 File Size

Figure 4.1 shows the histograms of file count and disk usage by file size, using power-of-two bins and plotted as line graphs on a logarithmic scale. Each x value represents the midpoint of the bin, and each y value is the number of files or bytes contained within that bin. It is clear that the majority of files are very small, but much larger files consume the majority of the disk space. When considered together, these trends demonstrate that over 90% of the disk space is occupied by less than 1% of the files. These distributions are very helpful in targeting large volume of disk space with only a small number of files. It is very easy to use the two graphs to assess how many files and how much disk space would be targeted by a simple size-based selection filter such as "files larger than x", and such a filter may be very useful in selecting a small number of files for large potential savings in disk space.

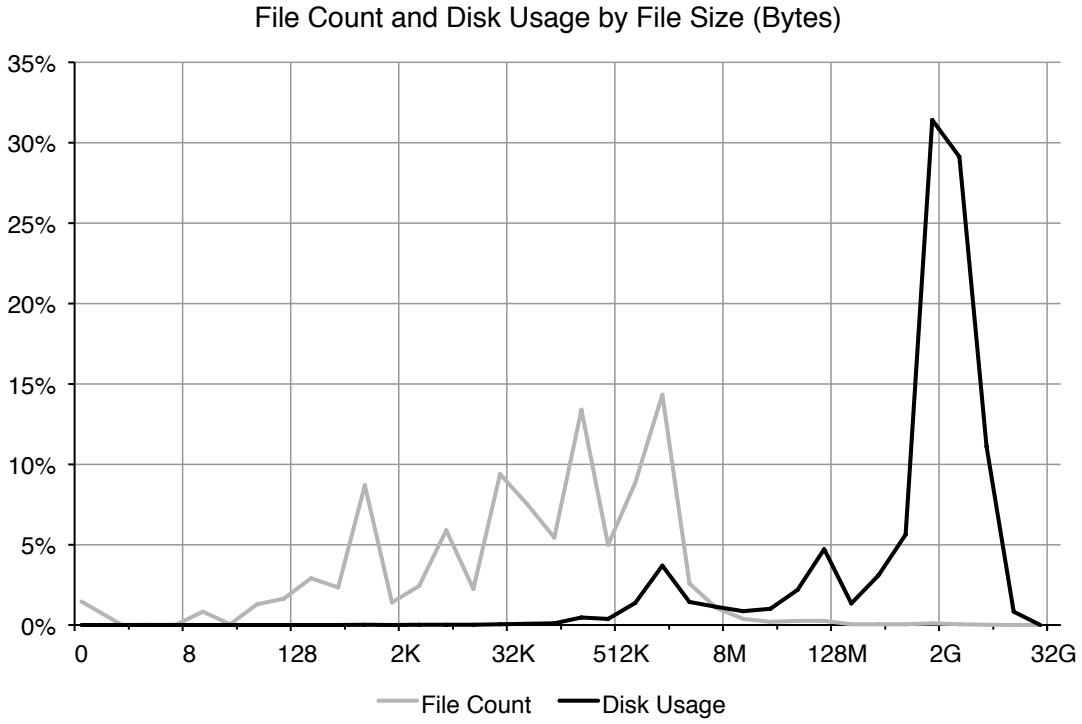

FIGURE 4.1: Histograms of file count and disk usage by file size.

Although the file size distribution helps to identify a small number of files that account for a large proportion of the disk space, it provides no insight into the content or nature of those files. Previous work asserted that the key peak was primarily composed of 3D solution files, but this is no guarantee that all large files are solution files. It follows that a solely size-based filter would still require significant human input to pass a decision on each individual file before any action could be taken. As such, the file size rule may be a useful secondary filter if the profile of otherwise selected files contains a large number of small files, but it is of little use as a primary condition.

4.4.3 File Age

Figure 4.2 shows the histograms of file count and disk usage by file age, as defined by the most recent of the last access timestamp (atime) and the last modified timestamp (mtime). Again, these histograms use power-of-two bins and are plotted as line graphs on a logarithmic scale. It is clear that most of the data are older than one month, and that there is a peak at approximately one year. As stated previously, this is quite old, certainly compared with many other HPC file systems [62]. Moreover, these are all no-backup volumes, and are thus supposedly used as a scratch area for disk intensive work, or as temporary storage until permanent space can be found elsewhere. In either case, valuable data should not be stored on these volumes for extended periods of time.

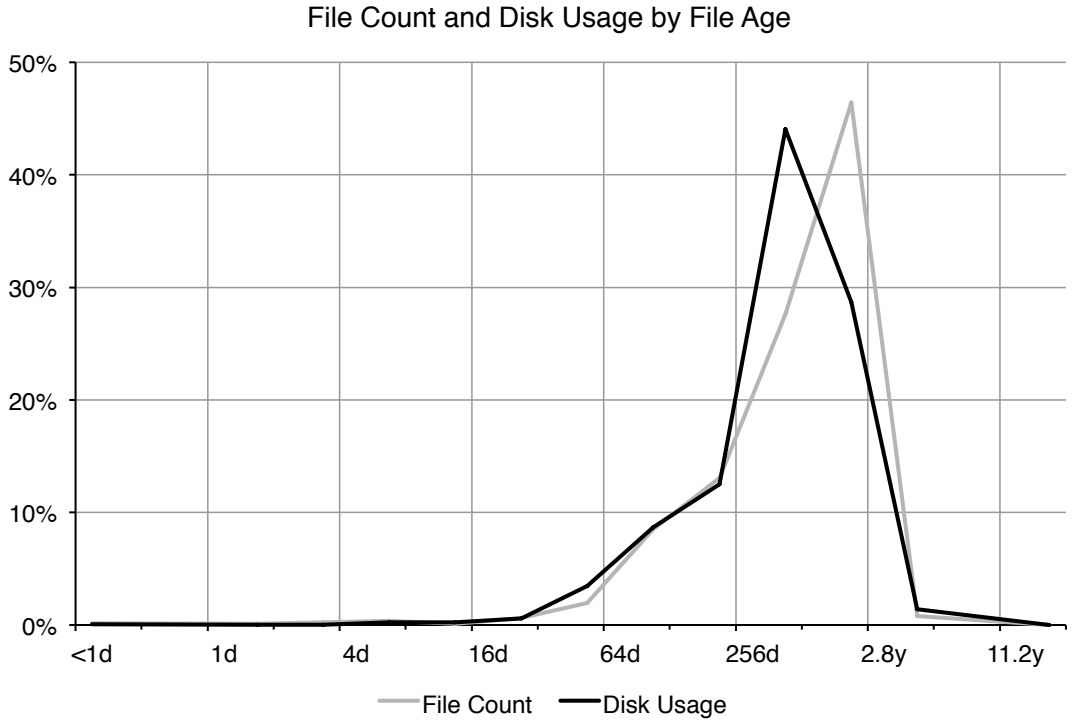

FIGURE 4.2: Histograms of file count and disk usage by file age.

Despite the logic behind the potential removal of all data beyond a certain age, there is an understandable reluctance to perform this action without knowing exactly which files and what content will be lost. Combining the file size and file age filters could provide a relatively short list of large files, which could then be approved or rejected manually. However, this is still quite a laborious approach since nothing is known about the content of the files selected. Again, more information about the type and content of files is needed in order to provide a suitable list of targets, especially if automation of the complete process is desirable.

4.4.4 File Type

More information about the nature of a file is needed in order to improve on the combination of simple attributes like file size and file age. Previous work in file type recognition used regular expressions to identify many key file types on the Airbus file systems, and it was found that 3D solution files accounted for most of the bytes, but only a tiny fraction of the files. Figure 4.3 shows stacked bar graphs of the file count and total bytes for some of the most common file types for Airbus data, derived from the no-backup volumes. The no-backup case slightly extends the previous observation, with 3D solution files accounting for 69.2% of the bytes in only 0.6% of the files.

This observation is the most ideal case for the basis of a file type filter, in that it provides a specific indication of the type of content, accounts for a large proportion of the overall

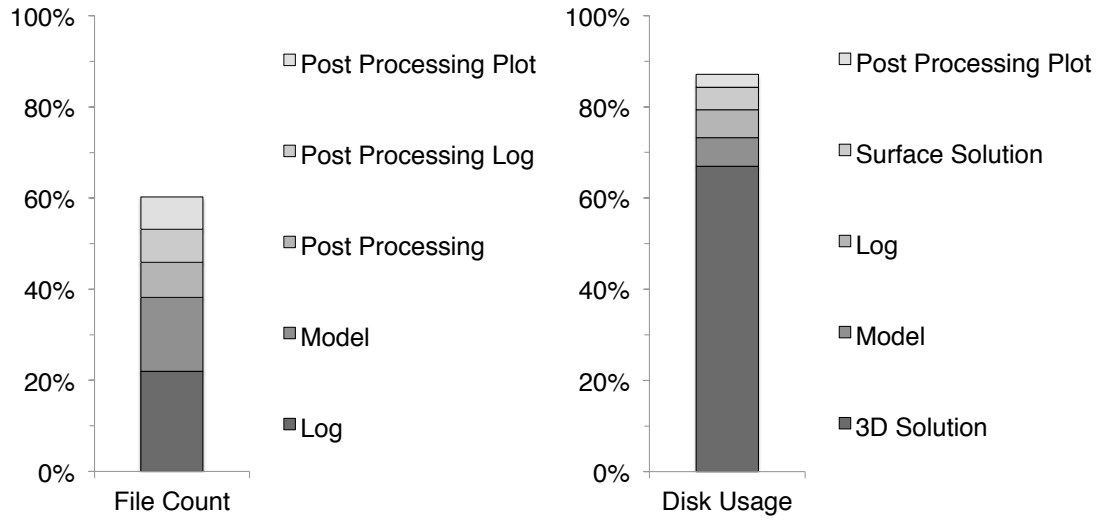

FIGURE 4.3: File count and disk usage for common file types, specifically on the group of no-backup volumes targeted for data cleaning.

disk space, and returns a relatively small number of files. To elaborate, the 3D solution file type is much more specific than `.log` files, in that log files may be generated by any number of applications for any number of reasons, but 3D solution files only ever describe the flow of fluid over an object.

The 3D solution files can be identified because they are generally created computationally and are named by convention. These files can thus be selected using regular expressions. For example, the regular expression `.*\.pval\.d+$` can be used to find all files that end with `.pval.` followed by an integer. This would select files with names like `A123V3_A0_M83.pval.1337` but would ignore those with additional file type extensions such as `.tar` or `.dat`. This is particular important because any deviation from the standard naming convention suggests that the content of such files may be different.

Moreover, first part of the file name is also constructed by convention, and includes more information regarding the flow parameters (Mach number M, Reynolds number Re and angle of incidence α). The path hierarchy also provides an indication of the parent project or study. These extra fields are the subjects of a separate study into additional metadata (see chapter 5), but their existence could be highly valuable to a file-type based rule, and for human assessors in any manual verification stage.

Furthermore, it has already been recognised that 3D solution files are an intermediate result in the larger workflow, and that in many cases there is no need for these files to be kept once the necessary post-processing stages have been completed. In principle, and setting aside the cost of computation, these files can also be regenerated from their original inputs. However, there are a number of issues regarding verification of the consistency of regenerated results, largely due to the iterative process followed and the secondary effects of parallel processing.

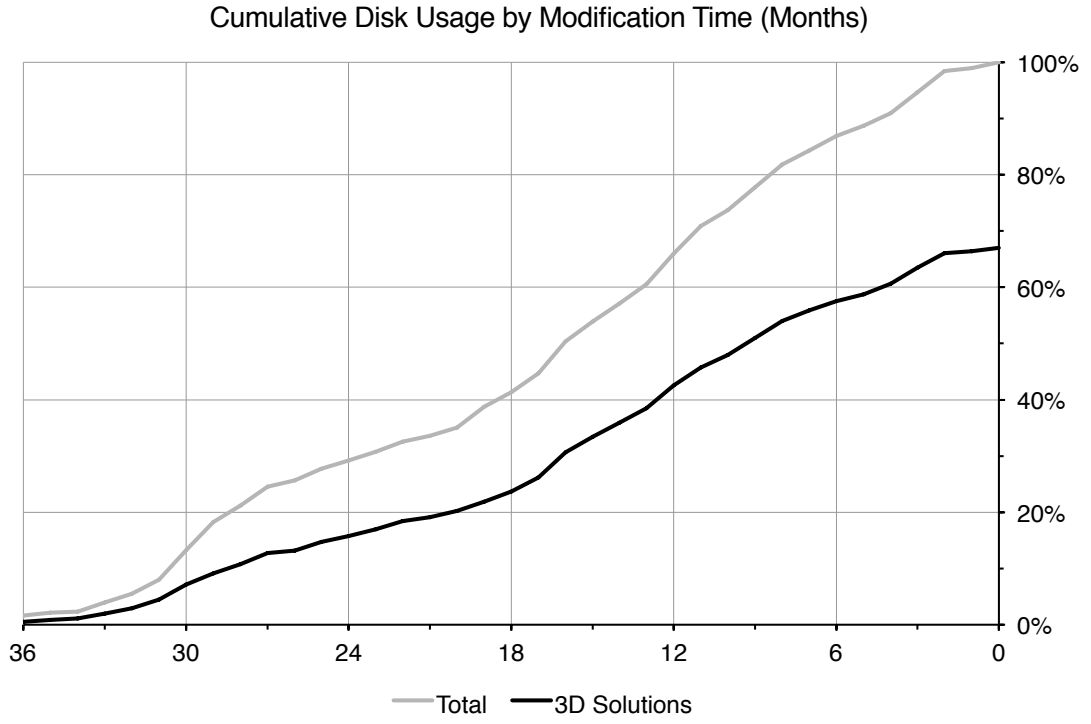

FIGURE 4.4: Cumulative disk usage by time since last modification, for all data and 3D solution files.

4.4.5 3D Solution Files by Modification Time

Based on the results examined thus far, a combination of file type recognition to select only 3D solution files, coupled with a file age filter to include only those older than an agreeable threshold, would seem to be the simplest and most effective cleaning method. An additional file size filter could also be considered, but due to the similarity in results between file size and file type for just flow solution files, this is likely unnecessary. However, since file age is a construct of the most recent of the "last accessed" and "last modified" timestamps, it is easier to simplify this and just use the last modification timestamp, `mtime`. This type of data is rarely accessed after creation, and the modification timestamp is unaffected by actions such as `touch`, or by utilities such as this disk cleaning program.

Figure 4.4 shows the total disk usage and the proportion occupied by 3D solution files, as a percentage of the total capacity of the volumes examined, against the elapsed time since last modification. Both distributions appear to be fairly linear, indicating that the total proportion of bytes in solution files is essentially constant over time. This illustrates the steadiness of the workflow, and suggests that it has not changed signficantly in the time period considered in the figure. The linearity also implies that the rate of data production remains constant for a given hardware configuration, assuming all systems are always in use, and indeed the change in gradient at 18 months corresponds to an

upgrade in HPC hardware. The linearity also means that there is no optimum choice for a time threshold, and that the decision could instead be made with respect to the business context.

The gradient of the total disk usage against time equals the net rate of data production. Since this study only considers a subset of the available storage systems, it is only possible to calculate the average rate at which these particular volumes are filling up, which figure 4.4 shows to be approximately 3.3% per month. Assuming no changes in the hardware capabilities and no changes in the workflow or data generating applications, knowledge of this rate of data production and the current state of the storage volumes allows the prediction of the remaining time until maximum capacity is reached, which is useful for both administration and capacity planning purposes.

4.4.6 Exclusions

Combining time-based trends with type information allows a selection of a small number of files which are good candidates for deletion. However, there may always be some exceptions that must be made for files that are particularly important, valuable or sensitive. Two key approaches to exclusions are possible: either exclusion by file or exclusion by user. Excluding individual files allows for very tight control over which files are excluded, whilst maximising the potential savings in disk space. However, due to the user effort required to select files in this way, the alternative user-based exclusion approach was adopted. Any users with any objections to the cleaning exercise were added to an exclusion list, which was used as an additional filter to ignore all files owned by any of these users when selecting candidates for cleaning.

4.4.7 Cleaning Rules

The first cleaning exercise was based on the selection of all 3D solution files that had not been modified in the last two years, excluding those owned by one user who opted out of the exercise. Due to the success of the first exercise, the second cleaning effort was made based on the selection of all 3D solution files that had not been accessed in the last eighteen months, again excluding the single user who chose to opt out. There appears to be no obvious optimum value for this time threshold, thus it can be reduced as far as deemed appropriate for the business and technical contexts of the data.

4.4.8 Execution

Prior to cleaning the no-backup volumes, a snapshot of each volume was made using the Python program developed for the previous study into file system metadata. These snapshots were then used to test the potential outcomes of different cleaning rules, particularly varying the time threshold. After settling on a rule (3D solution files older that two years, excluding those owned by users who opted out), a final search for files to be deleted was made directly on the file system using a Linux terminal. The resulting list was manually verified before the files were actually deleted, again directly from the command line. The volumes were then scanned again to support post-cleaning analysis.

Due to some early run-time problems, a few of the snapshot scans for the first study failed to complete. The vast majority of scans were successful, and the problem was eliminated for the second study. The snapshots were used to test potential cleaning rules, so the failures will have had a small impact on the assessment of these rules (although the proportion of failures was very small). The actual cleaning was conducted purely from a Linux terminal (i.e. using `ls | grep`), so the cleaning itself was unaffected by the snapshot failures. The results presented here are complete for volume fullness, where basic summary data was still available, but the rest of the analysis excludes the small number of failed volumes.

The first cleaning exercise was undertaken over May and June 2013. The actual cleaning operation was executed on June 7th; the remainder of this time was required in order to conduct the necessary pre-cleaning and post-cleaning scanning and analysis. The second cleaning exercise was initiated in November 2013, by which time most of the volumes were completely full again.

4.5 Results

This section presents the actual results of the two cleaning exercises. The most direct results are the changes to volume fullness, which describe the effectiveness of the cleaning process. The secondary effects of the cleaning process on the underlying metadata distributions are also presented, as these have implications for the sustainability of the method, and thus the effectiveness of any future data cleaning.

4.5.1 Volume Fullness

Prior to the cleaning exercises, the majority of the no-backup volumes were completely full. The cleaning process was instigated in order to clear space on these volumes to allow the continuation of on-going HPC work. As such, the selection of volumes by fullness presents an obvious source of bias, and the same level of savings may not be obtainable in other circumstances. The nature and usage of the volumes is another source of potential difference to other types of storage, such as project data volumes, which are backed up and as such are treated differently by the end users.

Figure 4.5 shows the impact of the first cleaning exercise on volume fullness. Each bar represents a single volume, and shows the total capacity; the remaining free space; the space cleaned as a result of the exercise; and the space that was used prior to the cleaning. The total disk space saved through data cleaning was 21.1 TB, which equates to a reduction in volume fullness of 15.2%. These are substantial savings in disk space, and no subsequent complaints of missing data have been made, indicating that the exercise was highly successful.

An interesting observation is that the volumes do not all have the same storage capacity. Rather than a uniform distribution, or even increasing volume capacities with time, figure 4.5 shows the older volumes with higher capacity than the newer volumes. This difference can be best explained as a result of managerial and purchasing decision making. However, the use of smaller individual devices does help to minimise the risk of data loss in the event of a catastrophic failure, and it is interesting to note the shift towards larger numbers of smaller devices.

Given the fill rate of 3.3% per month, as calculated previously from figure 4.4, it is possible to estimate the time to volume fullness to be 4.6 months. This is the estimated time from completion of the cleaning until all of the volumes are completely full again, and corresponds almost exactly to the true elapsed time between the end of the first exercise and the beginning of the second, although organisation and coordination held back the execution of the second exercise for a short time. However, this success again highlights the constant rate of data production in the Airbus engineering environment.

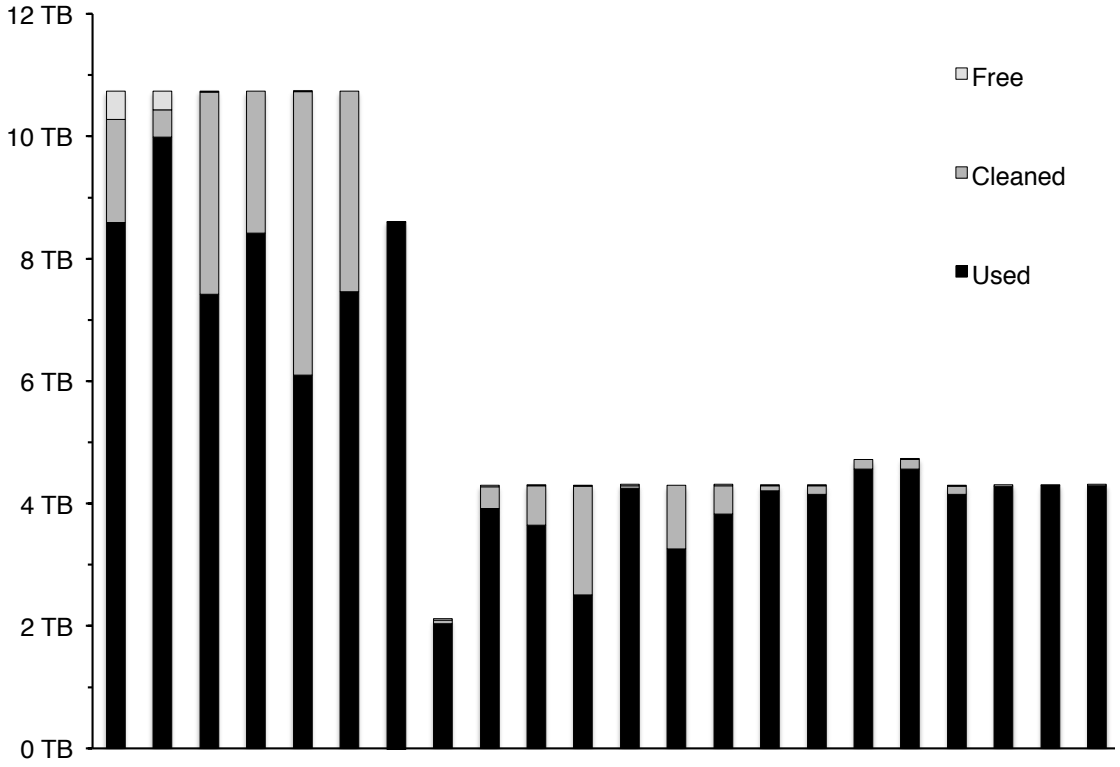

FIGURE 4.5: Volume fullness before and after the first cleaning exercise. Each bar on the x-axis represents a single (anonymous) volume that was cleaned.

It is also possible to estimate the potential disk space savings for the second exercise from the gradient of figure 4.4, using the revised time filter. Based on the six months between the two studies, and the change in time threshold from two years to 18 months, the second study effectively targets 12 months of data production. Multiplied by the mean data production rate (3.3% per month) and again by the proportion of disk space occupied by 3D solution files (69.2%), it is possible to estimate disk space savings of 27.4% for the second study.

Figure 4.6 shows the actual impact of the second cleaning exercise on volume fullness. Due to steady data production and the elapsed time between studies, the volumes were mostly full again prior to this study. As with the first exercise, the volumes are shown chronologically by installation date. Since the first study removed all of the very old data, it would be reasonable to expect somewhat more uniform savings across the volumes compared to figure 4.5, and this can be observed in the figure.

The second study identified and removed a total of 37.5 TB of unwanted data, which is again a substantial saving in disk space. The graph shows a total reduction in volume fullness of 27.5%, very close to the estimate of 27.4%. This indicates that the data production between the two studies closely followed the trend shown in figure 4.4. As previously, it is possible to translate these savings into a new estimate for the time until the volumes become full. Using the mean volume fill rate (3.3%) and the percentage of

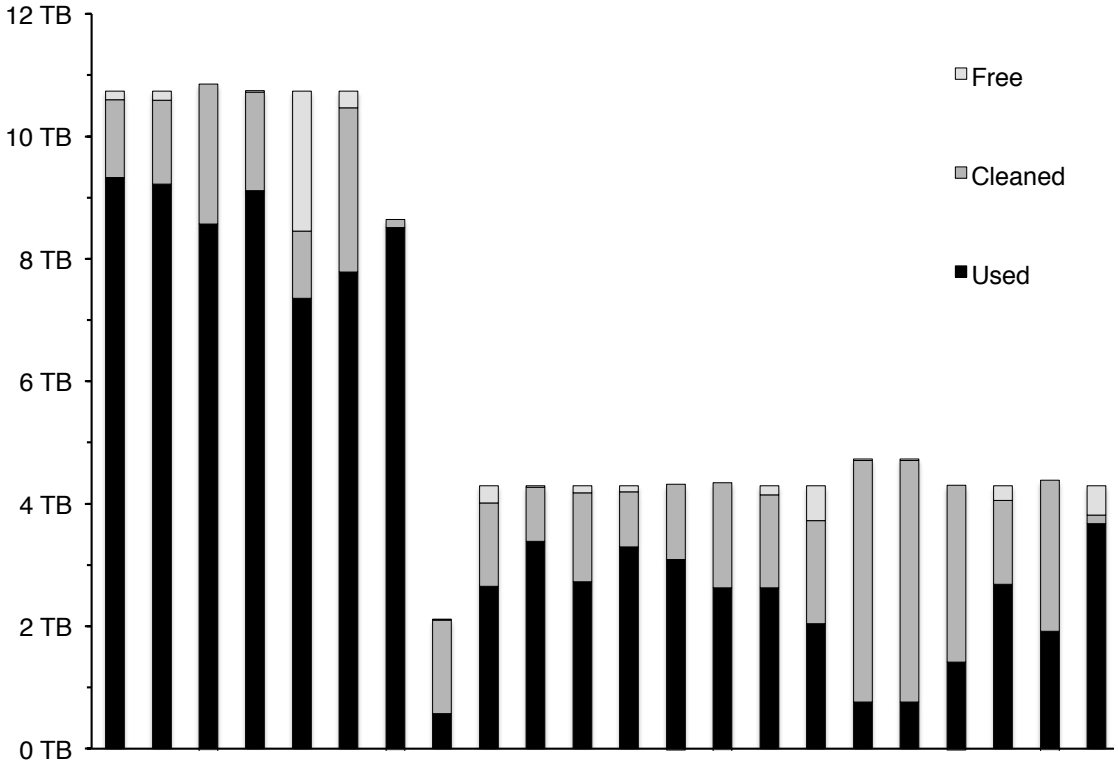

FIGURE 4.6: Volume fullness before and after the second cleaning exercise. Each bar on the x-axis represents a single (anonymous) volume that was cleaned.

volume savings achieved through the second exercise (27.5%), the new estimate for time to volume fullness is 8.3 months. Again, these results represent substantial savings in disk space, improve utilisation of the available resources, and an extended lifetime for the storage systems.

Considered together, figures 4.5 and 4.6 show evidence of some cleaning outside of this study. The free space before cleaning shown on some volumes in figure 4.6 exceeds the free space after cleaning shown in figure 4.5. These differences are either the result of end-users either removing files that are no longer required, or instead moving important data away from the no-backup volumes and onto a backed up storage system. However, the impact of this self-motivated movement or deletion of data is very small compared to the savings made through the cleaning studies.

4.5.2 File Size

Figures 4.7 and 4.8 show histograms of the file count and total bytes by file size, before and after the first and second cleaning exercises, respectively. Since only a very small proportion of the files were affected by the cleaning, the file count trends were the same before and after cleaning, thus only one of these is shown in each of the figures. In addition, both the pre-cleaning and post-cleaning distributions are normalised with respect to the pre-cleaning total disk usage, in order to better highlight the profile of the files that were removed during the cleaning.

In both cases, it is clear that only large files have been affected by the cleaning, primarily those in the main peak between approximately 256 MB and 8 GB. This is an expected result, since the 3D solution files were identified as a being typically large files within that range of sizes, and the file size filter formed the basis of the cleaning method. It is also clear that many 3D solution files remain on the volumes, and could potentially be cleaned in the future once suitable time since their last modification has been allowed to pass.

In addition, figure 4.7 shows some evidence of a small change in the mean file size of 3D solution files. The pre-cleaning distribution peaks at a slightly smaller file size than the post-cleaning distribution, indicating that the smaller files were targeted while the larger ones were not. This implies that there has been an increase in mean file size over time. Similar results have also been observed in environments where data sets have been examined over extended periods of time [2]. The effect is less noticeable in Figure 4.8, since the majority of the smaller files had already been deleted as part of the first study, and only a relatively short time had elapsed between the two.

4.5.3 File Modification Time

Figures 4.9 and 4.10 show histograms of the file count and total bytes by time since last modification, before and after the first and second cleaning exercises, respectively. Again, since the cleaning targeted only a small proportion of the files, the file count trends were the same before and after the execution, thus only one is shown in the figures. Likewise, the post-cleaning disk usage distribution is normalised with respect to the pre-cleaning total disk usage, again to highlight the profile of the files that were removed.

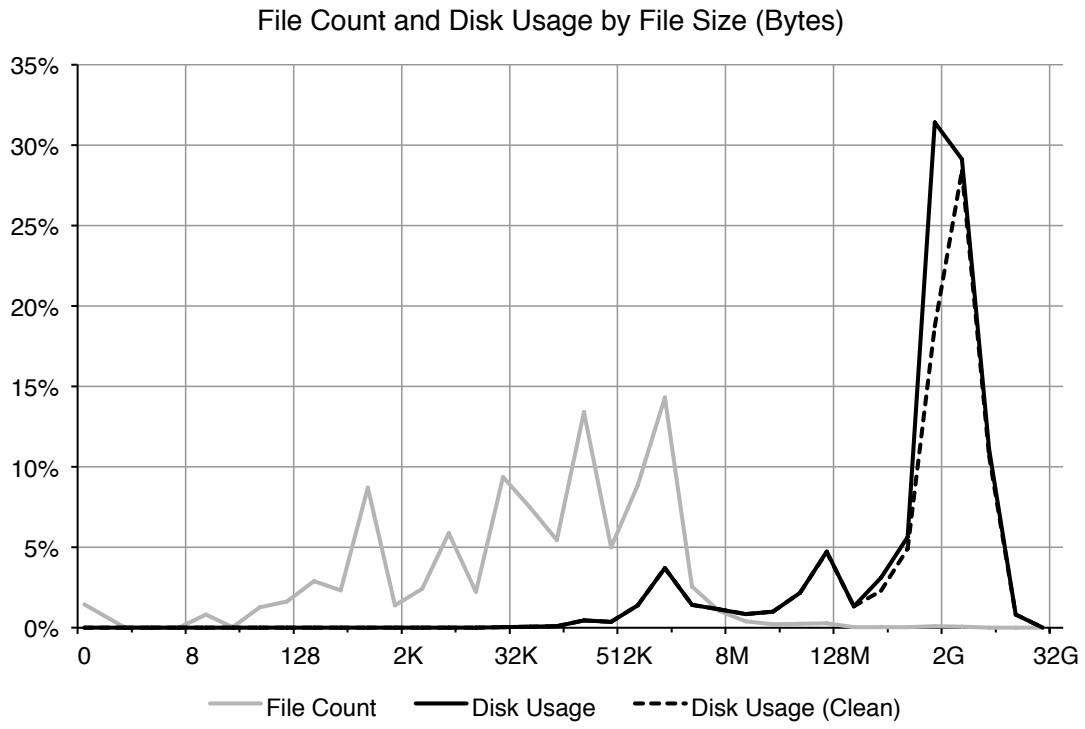

FIGURE 4.7: File count and disk usage by file size (first cleaning exercise).

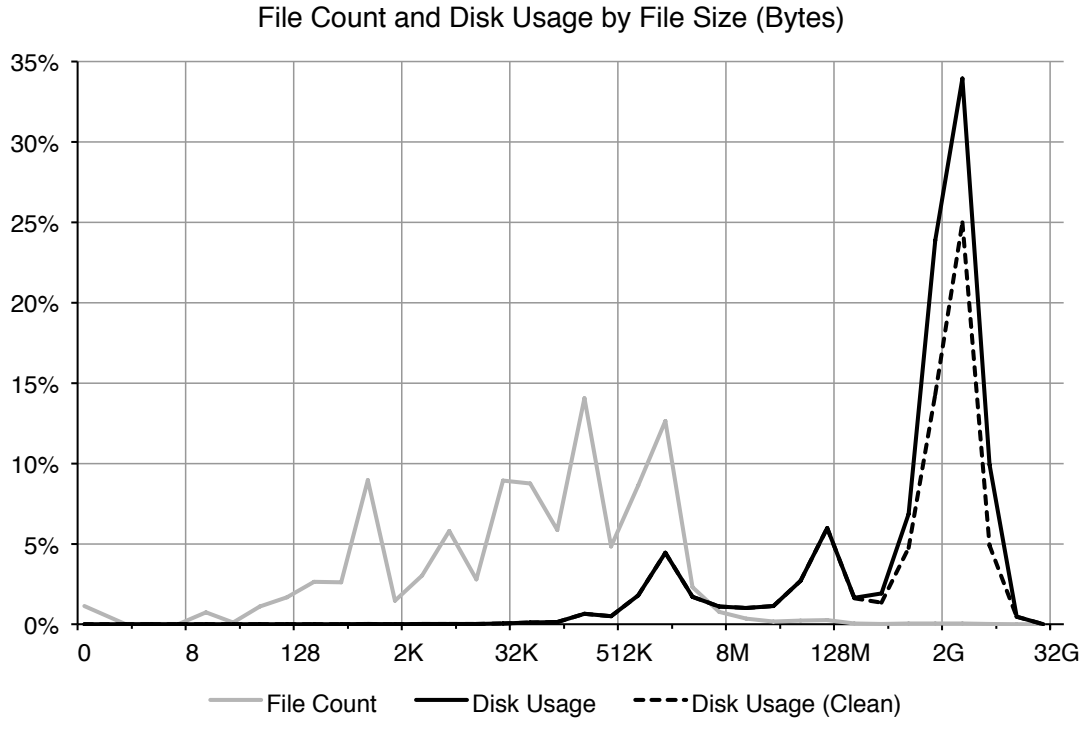

FIGURE 4.8: File count and disk usage by file size (second cleaning exercise).

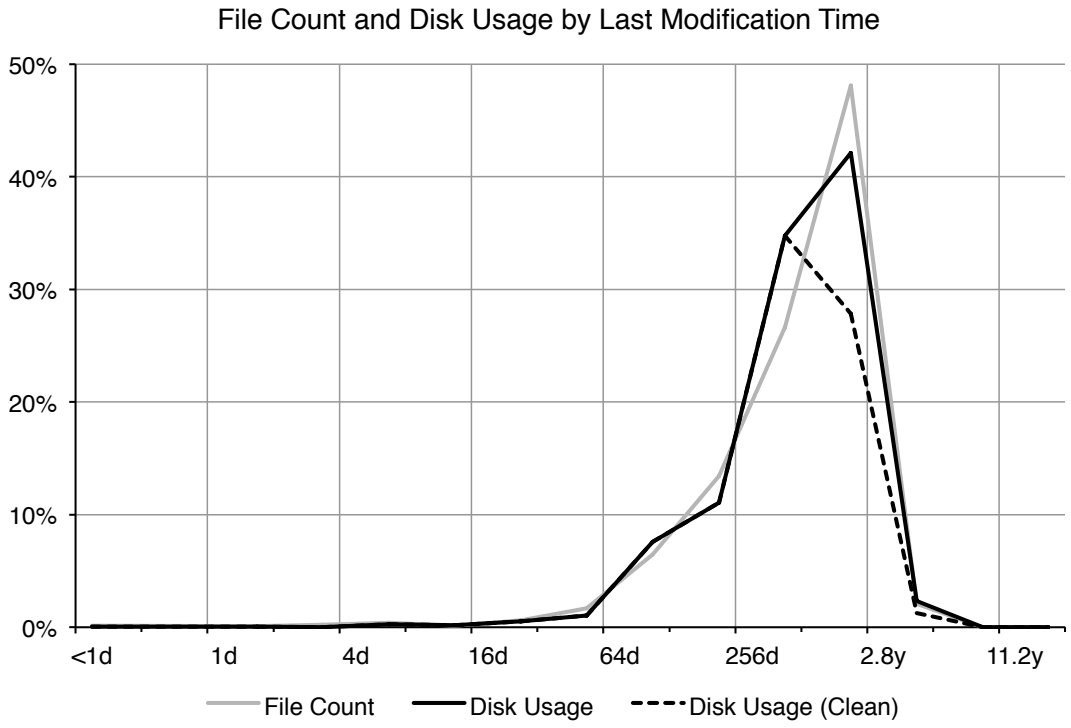

FIGURE 4.9: File count and disk usage by time since last modification, before and after the first cleaning exercise.

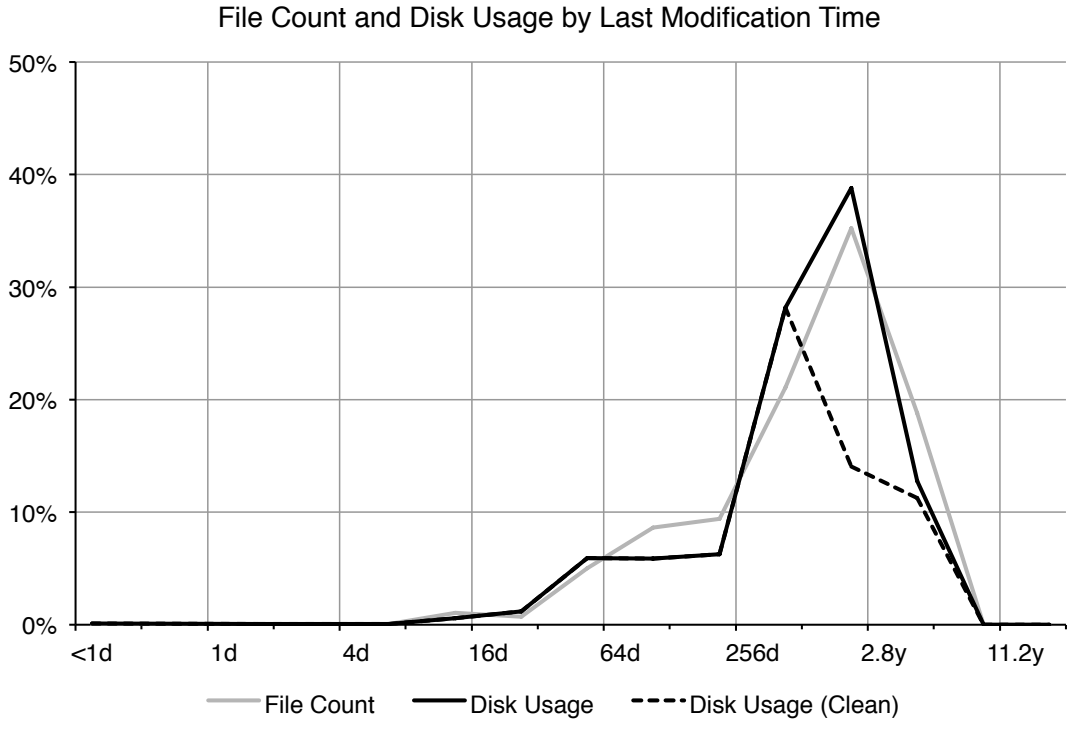

FIGURE 4.10: File count and disk usage by time since last modification, before and after the second cleaning exercise.

The graphs show that only the older files were targeted by the cleaning. Obviously this is another expected result, since the time filter was included in the cleaning method specifically to select only the older files. However, both graphs show some very old files still present after cleaning, since only the 3D solution files were selected for cleaning. Figure 4.10 shows a greater proportion of newer files targeted, partly since the older files were mostly removed during the first exercise, but also partly because of the difference in threshold in the time filter. It is also notable that the shapes of the distributions are quite different for the second study. This discrepancy relates to the missing snapshot files for the first study, as mentioned previously. As such, direct comparisons between the two graphs should be treated with caution, although the data is correct for each figure individually.

4.5.4 3D Solution Files by Modification Time

Figures 4.11 and 4.12 show the cumulative total disk usage, and the component of disk usage occupied by 3D solution files, before and after the first and second cleaning exercises, respectively. In both cases, it is clear that only files older than 24 and 18 months, respectively, were deleted as part of the cleaning process. It is also clear that significant savings were made in both cases, although a large proportion of the disk usage across the volumes is still occupied by 3D solution files. This means that there is still scope for savings to be made in the future, but in order to encompass more files either more time would need to pass, or the time threshold would need to be reduced.

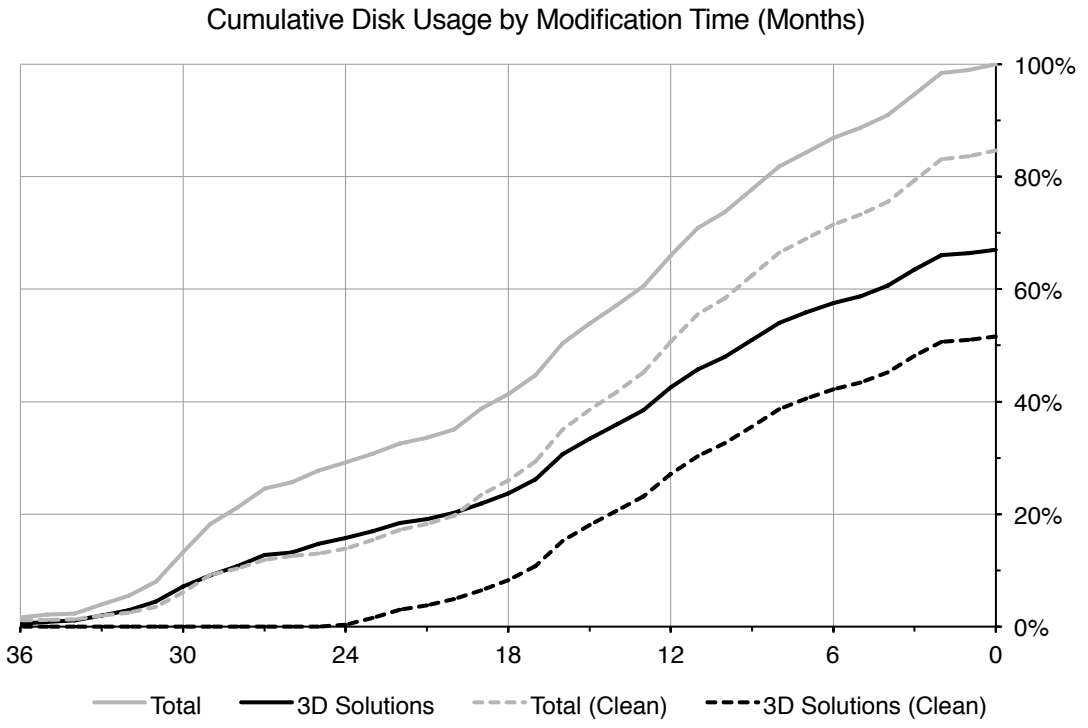

FIGURE 4.11: Cumulative disk usage by time since last modification, before and after the first cleaning exercise.

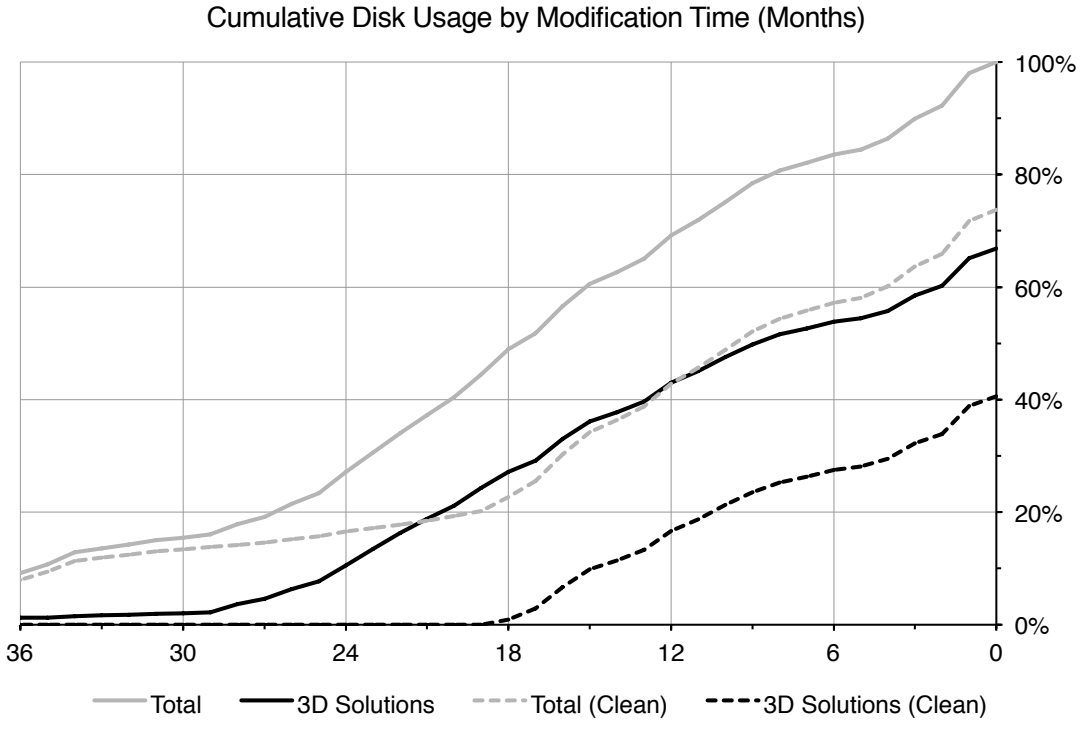

FIGURE 4.12: Cumulative disk usage by time since last modification before and after the second cleaning exercise.

4.6 Discussion

The results presented thus far show that the methodology was greatly successful in cleaning Airbus file systems. This section discusses the implications of the methodology and results, both for Airbus and for the wider engineering context.

4.6.1 Impact on Storage Systems

Both of the cleaning exercises proved to be highly successful, in that they each resulted in substantial savings in disk space. Figures 4.6 and 4.12, in particular, show that significant disk space was freed on many of the volumes, and that these savings were achieved without affecting the majority of files, by only targeting old 3D solution files. Since these files are only an intermediate stage in the workflow, it is highly unlikely that they will ever be needed in the future.

In addition to the savings provided through the data cleaning, the analysis used in preparation and execution of the methodology allow for the prediction of the time remaining until volume fullness. Such a prediction assumes that there are no changes to the HPC system, workflow, or applications responsible for data creation. Maintaining an accurate value for volume fill rate allows administrators and managers to plan cleaning work and/or the installation of further storage resources as necessary. Given the on-going nature of data production in engineering environments, it is crucial that sufficient capacity is always made available for new results.

Although the cleaning method has proven itself very effective in removing unwanted files and clearing disk space for new results, there is a limit to the sustainability of this approach. Only a portion of the overall data generated, namely the 3D solution files, are removed through cleaning. As time passes, and more cleaning exercises are undertaken, the proportion of disk space occupied by other file types, such as input files, log files, and post-processed outputs, will become progressively greater. This will lead to a reduction in the effectiveness of the cleaning methodology, yielding diminished savings in disk space as the 3D solution files occupy a shrinking proportion of the total disk space.

In the same way that it was possible to predict the time to volume fullness, it is possible to estimate the time until the volumes are filled with files that cannot be cleaned by the methodology outlined in this study. Solution files currently account for 69.2% of all new data, and storage is currently filled a mean rate of 3.3% per month. Combining these figures, the net fill rate accounting for the cleaning of solution files is (1 - 0.692) * 0.033 = 1.02% per month. After the second cleaning exercise the no-backup volumes already contained 33.1% non-solution files, so at a net fill rate of 1.02% per month, the volumes will be completely full of non-solution files in 65.6 months (approximately 5.5 years).

This represents a significant potential extension to the service lifetime of the no-backup storage systems. However, in order to achieve the maximum service lifetime on the no-backup volumes, cleaning will need to be performed progressively more often, and the retention time for 3D solution files will need to be steadily reduced. There will come a point where the retention time cannot be reduced any further, as good practice requires that all necessary post-processing tasks must be completed before any solution files are removed. Since the effectiveness of the cleaning method will diminish with time, the available space for new data will also diminish over time, even with regular cleaning. It follows that although the theoretical remaining service life for the no-backup systems is 5.5 years, practical considerations will reduce this value. The magnitude of this reduction will depend on the minimum retention time limit, and on the minimum storage volume that could support on-going data production.

It is also important to note that the estimates made throughout this report assume a constant rate of data production, which would be affected by any changes in HPC usage policy or HPC capabilities. Any changes in HPC usage, capabilities or workflow will thus require a re-evaluation of data production rates and the proportion of 3D solution files, in order to revise the time-to-fullness and service life estimates.

4.6.2 Cost Implications

Although the cost of disk-based storage devices has fallen greatly in recent years, storage on an industrial scale remains a non-trivial expense. Overheads such as management and maintenance costs, supplier costs, and running costs all add together to make storage costs much greater than the cost of disks alone. Tangentially, there are also environmental consequences to adding more storage systems, in that such an action requires more energy, thereby increasing the carbon footprint. It follows that although disk drives are relatively cheap, this is no reason to be wasteful or profligate. The data cleaning work demonstrated in this report represents significant cost savings through the improved efficiency and prolonged service life of existing storage systems.

Anecdotally, it was commented during discussion with Airbus managers and end-users that saving disk space also implies cost savings through the time spent by engineers trying to find sufficient space for their results. It emerged that although the suggestion may sound somewhat trivial, if an engineer must spend one minute per job to find a suitable storage location, then the cleaning already completed will have more than justified the funding for this research. Considering the future usage of the cleaning method too, it is clear that this research project has been highly successful.

4.6.3 Methodology Evaluation

Perhaps the key success of the cleaning approach demonstrated is the ease with which it can be applied. Although more sophisticated analysis was used in the development of the rules, the resulting selection and deletion can be achieved using only basic command line instructions, and there is no need to repeat any of the analysis work prior to execution. As explained in the introduction, the development of a cleaning methodology that required only *"minimal human input and interaction"* was one of the key goals for this work, and this goal has certainly been achieved.

One limitation of the methodology is that it relies heavily on the proportion of total bytes occupied by 3D solution files, and their identification through file naming conventions. A similar proportion of bytes occupied by solution files has been found in other environments where CFD analysis is prevalent [33]. There is no guarantee that naming conventions will be utilised elsewhere, but it is certainly likely that larger-scale environments will launch jobs through similar submission tools. As such, the methodology is likely to be of relevance to other environments using extensive CFD. However, some re-evaluation of the specific parameters and identification methods will be necessary, and a full analysis of the data would certainly be prudent.

Since the methodology is so heavily based on the 3D solution files, the applicability of the method to other environments also depends on the retention policies for this file type. The methodology was built around the assumption that these files are typically only an intermediate file in the larger workflow, and that the key results are those obtained through the post-processing stage. There are some cases where long-term retention of solutions is essential, such as aviation certification, where all data must be kept for the lifetime of the aircraft programme. Since the cleaning method removes these files, care must be taken to separately identify and handle any data that may require longer term retention. This process may be aided by the inclusion of additional metadata fields, the extraction and analysis of which are the subject of further work.

However, even if any of the deleted files were to be required in the future, the file system still holds all of the input, output and log files used and created as part of the same workflow. Given the original input files to the job, it should in theory be possible to regenerate the deleted data as necessary. It should be noted that the nature of numerical methods in high performance computing may result in some numerical differences between results, even if the engineering implications for the data are the same. In addition, any differences in the architecture of the HPC systems may lead to differences in the speed of computation and the number of iterations required for numerical convergence. These issues, and their implications for data regeneration, present some interesting challenges for further research, but are beyond the scope of this study.

4.7 Conclusions

In conclusion, this chapter has demonstrated the development and execution of a cleaning methodology that provides significant savings in disk space on a particular group of file system volumes. Two separate cleaning studies yielded savings of 21.1 TB (15.2%) and 37.5 TB (27.5%), showing that substantial disk space can be cleared by removing only files that are no longer required, thus making more effective use of existing resources. In addition, it was demonstrated that it is possible to make accurate predictions of the savings that could be made at any given time and the time remaining before the volumes become completely full. It was estimated that based on the current rate of data production, the cleaning method could effecively extend the life of the existing storage systems by up to 5.5 years. Ultimately, the volumes will still become full, and more capacity will need to be added, as the method targets only a particular subset of the stored data. However, it is still clear that the cleaning method provides a significant practical benefit, and will lead to a substantial extension in the service life of the examined volumes.

Chapter 5

Additional Metadata

Chapters 3 and 4 demonstrated that valuable metadata exists within file systems, and can be used to make significant improvements in data management, particularly in data cleaning. These studies were based on file system index-node metadata, which is readily available and easily accessible. However, in a highly computational environment such as engineering design, there are other sources of metadata, such as log files and naming conventions, which may be able to provide more information.

This chapter presents research into the identification and extraction of metadata from computational log files and naming conventions, again within the context of file systems supporting aerodynamics at Airbus. The focus was on obtaining useful information from these sources and associating it with the 3D solution files identified in chapter 3 and targeted for cleaning in chapter 4. The subsequent analysis then considered the availability of metadata, particularly in terms of the proportion of data for which it could be found and associated. It also considered the nature of trends and distributions in the metadata values, and linked these findings to relevant characteristics in aerodynamics and engineering design.

It was found that metadata availability varies depending on the specific details, but metadata could be identified, extracted, and associated with a significant proportion of the data. Log file metadata was found for 55.3% of the solution files, and 80.3% by bytes, corresponding to 53.7% of the overall bytes stored on the file system volumes examined. The availability of metadata from naming conventions was much more variable, but was still present for much of the overall data in many cases.

These findings are particularly useful in better describing existing data, and may be applicable anywhere that exhibits highly automated data production. The extracted metadata may be used to help improve data management practices by better understanding the data purposes and contexts.

5.1 Introduction

The engineering design process makes extensive use of high performance computing (HPC), and is responsible for the production of large volumes of data. These data must be stored, managed, and shared across worldwide networks, and demonstrate some notably unique characteristics, particularly in terms of file size and retention requirements. Previous work in file system metadata [62] and practical data cleaning in engineering design [61] demonstrated that generic and readily accessible file system metadata attributes can be used to develop effective data management methods, yielding significant time and cost savings, and extending the life of existing storage.

Beyond the basic hierarchy and index node attributes (e.g. file size, access time stamps, permissions, etc.), the engineering data examined in this thesis are essentially unstructured. There are no detailed schemas, formal rules for data identification and tagging, or well-defined groupings or collections of related files. At best, there are some recommended conventions for the naming of files, some commonly used ways of organising closely related data, and some fairly regularly produced log files that may contain useful information if suitably processed. The data are thus semi-structured, meaning that although fundamentally unstructured, there are some structural elements that may be present in a meaningful proportion of cases.

File systems supporting HPC systems, particularly in industrial-scale engineering environments like Airbus, contain data that is largely produced computationally as part of larger workflows. The processes responsible for data creation are largely automated, thus semi-structural characteristics may be present in a significant enough proportion of the data to be useful to data management methods. This chapter examines file naming conventions and log files in order to extract metadata describing both content and computational characteristics, and discusses the use of these attributes to develop more complex data management methods. Any additional metadata may be particularly useful in enhancing data search and retrieval, and also data cleaning. These were two of the key challenges outlined in chapter 2.

5.2 Industrial Context

This research into additional sources of metadata was undertaken at a single site at Airbus, a leading aircraft manufacturer, specifically on a group of file systems supporting HPC for aerodynamics. The files examined were predominantly aerodynamic files pertaining to the computational fluid dynamics (CFD) analysis undertaken as part of the aerodynamic design process. The file systems examined are the same no-backup UNIX NFS volumes studied previously in chapter 4, which serve as temporary storage between data generation and more permanent storage on a backed-up system.

Due to the constant data production these volumes rapidly become full, whereupon data must be archived or deleted as appropriate. The previous chapter showed how simple file system metadata can be combined with knowledge of the working environment and data lifecycle to develop a cleaning methodology to target unwanted data for archival or deletion. It went on to deploy this methodology within Airbus and demonstrated that significant savings in disk space can be made using only very basic information. However, more sophisticated cleaning tools might be made possible through access to a greater variety of context-specific metadata, such as the aerodynamic and computational metadata examined in this chapter.

The CFD process responsible for the production of most of the data examined comprises the analysis of the fluid flow over engineering components and assemblies under varying flow conditions. This chapter considers the metadata that can be identified and associated with the 3D solution files, which are intermediate files containing full details of the fluid flow for the given conditions. These files will have been computationally generated by CFD applications such as elsA [59] and Tau [23], and are usually subjected to further post-processing in order to extract key values such as the overall forces acting upon the designed object.

Each file type in this workflow may be tagged with a variety of metadata fields, including project information, aerodynamic parameters and computational values. This chapter primarily considers aerodynamic and computational metadata, and focuses on the 3D solution files, since these account for the greatest proportion of the overall disk space. It would be interesting to consider more file types and metadata fields; to track the parent-child relationships between files; and to consider the shared or total metadata for larger collections of related files. However, these goals are beyond the scope of the research described in this thesis.

5.3 Methodology

This section describes the methodology for associating log files with 3D solution files, extracting metadata from these log files for mapping to the solution files, and then extracting further metadata from file names based on known naming conventions. The work is based on file system snapshot data acquired for previous work in file system cleaning [61], although an additional scan was required to parse specific log files.

5.3.1 Log File Identification and Filtering

A common convention across many environments is the use of file type suffixes to indicate the type of data contained in a suffix. Although not typically enforced by the file system, the presence of well-known file type extensions is often sufficient to identify the likely type of content. For the purposes of this chapter, files using the common file name suffix `.log` were deemed to be log files and were treated accordingly.

However, since the `.log` suffix is a rather broad file type, the selection of all `.log` files yielded a large number of files that did not pertain to the production of 3D flow solution files. Since only the log files relating to solution files were of interest for this research, two types of filter were applied to the initial selection. The first ensured that only log files with a given proximity to a solution file were included in the analysis. Preliminary analysis revealed that solution files are nearly always stored in a subdirectory (often named `/sol/` or `/solutions/`) below the parent directory to the log files of interest, so this observation was used as the basis for the proximity filter.

The second filter excluded log files with common names that are known to pertain to other types of work, most often the post-processing of solution files into final results. In order to implement this filter, a list of substrings and regular expressions corresponding to large groups of log file names was gradually built up, until nearly all of the log files could be identified as more specific subtypes. The substrings and regular expressions were manually filtered to include only those corresponding to the creation of flow solution files, as these were the only types of interest.

5.3.2 Flow Solution Identification

Flow solution files do not use a simple suffix to denote their type. Instead, these files follow a naming convention that allows the file name to contain a name, a component that indicates that the file is a flow solution, and the number of iterations used to compute the data. The name component may contain information about the particular geometry, mesh and flow conditions used to create the solution, as this allows quick visual identification of particular solutions in larger collections.

For example, for a solution file named `A123V3_A0_M83.pval.1337`, it would be reasonable to expect that the corresponding model would be named `A123V3` and the CFD solver would have run for 1337 iterations before reaching the desired convergence. It is also possible to extract the aerodynamic flow conditions. The component `A0` provides α, the angle of incidence of the model to the direction of the fluid flow. In this case, the model was aligned with the direction of fluid flow. The component `M83` refers to the Mach number, the free-stream velocity of the fluid flow with respect to the speed of sound; in this case $M = 0.83$. Although this is just one example of extracting aerodynamic data from the file name, the actual methodology used was very similar, and is considered in more detail in subsection 5.3.5.

This naming convention allowed solution files to be identified using the regular expression "`\.pval\.d+$`". The end of the file name was explicitly specified in the expression, as this ensured that similarly named files, such as `.dat`, `.tar` and `.tar.gz` files were excluded from the selection. Although the type content for such files may actually be the same, it was much simpler to simply exclude this very small proportion of files than to try to perform more detailed analysis on the internal content of such container types.

5.3.3 Log-Solution Association

Solution files can be directly referenced by log files, in which case an explicit association can be made between the log file and the solution file. An explicit association is a strong basis for a relationship between a log file and a solution file. For example, a log file may contain a line such as the following:

```
Copying back final output /sol/A123V3_A0_M83.pval.1337
```

This indicates that the log file in question is the parent to the mentioned solution file, and could form the basis for an explicit association between the two files.

Alternatively, if explicit associations cannot be made between solution files and log files, implicit associations can be made if one or more solution files are present in a subdirectory to the parent directory of one or more log files. This is a much weaker association than the explicit referencing of solution files in log files, but the computational data generation means that files are typically grouped in the same manner. Care must be taken, particularly where there are multiple log files and multiple solution files, but implicit association provides an alternative to explicit association where the latter is impossible.

Early testing revealed that the explicit association method provided only very limited coverage of the data, providing the additional metadata for only 5.24% of the solution files, or 7.54% when weighted by file size. The implicit approach demonstrated much better coverage, providing additional metadata for 57.39% of the solution files, and 80.28% when weighted by file size. Although the more accurate explicit association

method would be the desirable implementation, the coverage of the data was clearly insufficient to provide a meaningful representation of the extra metadata. As such, the implicit approach was selected as the method of association for the remainder of the research.

5.3.4 Log File Metadata Extraction

Each log file was parsed line by line, using regular expressions to check for the presence of predetermined fields of interest, and extracting the values accordingly. The fields targeted for extraction were as follows:

CPU Cores: The number of CPU cores requested for the job upon submission. Not all of these cores are necessarily used, which will have an impact on the reported CPU time. Knowledge of the number of CPU cores is useful for capacity planning and research into scheduling algorithms, but also gives a one-dimensional view of the size of the job responsible for data creation.

Execution Time: Also known as the wall time, this is the literal time elapsed between the start and end of the job, excluding any time spent queueing. The wall time has a strong impact on the user, more so than the CPU time, since this is the actual time taken for their job to return the results.

CPU Time: The CPU time is the total amount of time used by all of the CPU cores, and is a measure of the computational cost of the job. Due to the potential discrepancy between the requested and actual number of CPU cores used, the compute time does not necessarily equal the number of cores multiplied by the wall time. In order to study this effect, the latter value was also calculated, and a direct comparison made.

It is common practice for multiple solution files to be created by the same computational job, for example when examining the effect of varying a particular parameter on the flow solution. The log files created by such jobs thus contain metadata that applies to the set of resulting solution files, rather than to any one in particular. It is not possible to calculate the exact distribution of a metadata field value over multiple solution files, but it is possible to make an approximation by assuming that the solutions were created sequentially. In this case, the number of CPU cores is unaffected, and the CPU time, wall time, and block time can be divided by the number of solution files to give the assumed mean values for the batch job.

5.3.5 File Name Metadata Extraction

The file names for 3D solution files often contain valuable information, particularly regarding the aerodynamic flow parameters used in the simulation, but also the number of computational iterations required to achieve the desired level of convergence. The following parameters were examined:

Mach Number (M): This is the *"non-dimensional ratio of the characteristic speed of the flow to the speed of sound"* [18], and represents the degree of aerodynamic compressibility. A typical airliner will not usually exceed the *critical Mach number*, which is the *"free-stream Mach number at which sonic flow is first achieved on the airfoil surface"* [6], as the effects of sonic flow *"may be such as to cause the aircraft to become uncontrollable"* [39]. As such, typical values for the primarily civil aviation context at Airbus will fall within the range $0.7 - 0.9$, although some experiments will undoubtedly have taken place outside this range.

Angle of Attack, Angle of Incidence, Alpha (α): This is the *"angle between an airfoil or wing and the free-stream flow velocity vector"* [18]. The lift produced by an airfoil or wing usually increases quite linearly up to a given maximum lift, which occurs at the *stalling angle*, typically in the range $10° - 15°$, beyond which the lift decreases rapidly due to aerodynamic stall [39].

Lift Coefficient (C_L): This is a non-dimensional quantity representing the lift force generated by an object, and is often used in aerodynamics in place of the actual lift force due to the variation of the latter with other parameters, such as velocity and air density [39]. Lift coefficients for typical wings and airfoils rarely exceed about 2, and are often rather smaller.

Number of Iterations: The nature of the numerical methods used in CFD analysis requires the process to be repeated over a number of iterations until the desired level of convergence is achieved, assuming that such a solution can be found. The number of iterations is notable as a key component in the regular expression used to identify 3D solution files.

Project(s): This attribute refers to the presence of any high-level Airbus aircraft programmes (e.g. A320, A350, A380, etc.) that are present in the complete path. The key difference between this attribute and the others is that an unknown number of valid project fields might be detected in the path. The most common scenario is that an umbrella project will have a set of dedicated file system volumes, each of which

may contain data pertinent to various spin-off projects. This attribute thus contains n-number of fields, which are assigned numerically ascending names upon detection (i.e. `project1`, `project2`, `project3`, etc.). The detection works from left to right, thus the fields will always be sorted by their depth in the hierarchy, from top to bottom.

5.3.6 Data Presentation

Data distributions are generally shown as normalised histograms and CDFs (cumulative density frequency), weighted by both file count and total bytes, and plotted as line graphs. Any exceptions to this convention are explained as and when they occur. Some of the trends use power-of-two bins, although others use linear bins, depending on the range and nature of the particular attribute. For the aerodynamic attributes in particular, scales have been selected to best demonstrate the natural groupings evident in the data.

TABLE 5.1: Availability of Additional Metadata Attributes. The table shows a summary of the proportions of files and bytes for which additional metadata was successfully extracted and associated with 3D solution files. As explained in section 5.4.1, the "absolute" column is calculated with respect to all files and bytes; the "relative" column with respect to all 3D solution files; and the "implicit" column with respect to all 3D solution files for which a suitable log file could be associated.

Attribute	Absolute (%)		Relative (%)		Implicit (%)	
	Files	Bytes	Files	Bytes	Files	Bytes
CPU Cores	0.12	53.68	55.26	80.26	100.00	100.00
Wall Time	0.12	53.68	55.26	80.26	100.00	100.00
CPU Time	0.12	53.68	55.26	80.26	100.00	100.00
Mach Number	0.14	41.51	65.25	62.06	56.64	58.60
Angle of Attack	0.04	20.01	21.08	29.91	25.33	31.91
Lift Coefficient	0.15	42.44	70.10	63.45	71.41	62.96
Iterations	0.20	66.79	96.29	99.85	100.00	100.00
Projects	0.18	61.25	86.81	91.57	87.00	91.80

5.4 Results

This section examines various metadata attributes, their availability in the actual data set, and the trends that can be observed in their characteristics.

5.4.1 Metadata Availability

Although the metadata attributes are of significant value, one of the key factors to examine is their availability. In considering large collections of files in this way, any additional metadata must be available for a significant proportion of the actual data in order to be particularly useful. Given the unstructured (or at best, semi-structured) way in which the files are stored, it is unlikely that extra metadata could be extracted for every file, or even every solution file, thus calculation of the additional metadata availability is imperative.

Defining availability as the proportion of files for which a particular attribute could be identified, there are three frames of reference that could be used. Dividing the number of files with the desired attribute by the total number of files in the file system yields an *absolute* value. Alternatively, dividing by the number of solution files yields a *relative* value, of significance because a perfect scenario would always yield 100% availability if calculated this way. A similar *implicit* relative calculation could also be made with respect to the number of implicitly-matched solution files, i.e. those for which appropriate log files have already been identified and associated. This final option yields 100% where log file data can be found, and enables the relative comparison of availability between attributes from different sources, such as log files to naming conventions.

In addition, each frame of reference can be calculated in terms of the number of files or in terms of the total number of bytes. As demonstrated previously, solution files in the source data account for a large proportion of the total bytes, but only a very small proportion of the files [62]. Calculating metadata availability both by file count and total bytes provides an extra view of the results, highlighting any potential anomalies caused by a skew in the file size distribution.

Lastly, the `projects` attribute, as already discussed, contains a variable number of fields for each file, depending on the complete file path and any project names identifiable within. In the calculation of availability for this attribute, files with at least one project field were counted, but any superfluous project fields were ignored. Availability for projects thus applies to the existence of at least one project field.

Table 5.1 shows the availability of each of the extracted attributes, calculated for the absolute, relative, and implicit frames of reference, in terms of file count and total bytes, as explained previously. The absolute availability shows that there is much variation in availability between different attributes, from about 20% to nearly 70% by bytes. It is

worth noting that angle of attack and lift coefficient are never both present for the same file, since the purpose of CFD analysis is typically to determine one of these two values. Due to this mutual exclusivity, it is perhaps better to consider these attributes together, yielding much more favourable availability values. With this in mind, the extraction of aerodynamic attributes (Mach number, angle of attack and lift coefficient) from file names appears generally to have been very successful.

The attributes extracted from log files (CPU cores, wall time, CPU time) show consistent availability, and the 100% values in the implicit relative availability indicate that the suitable log files contain all of the desired information when they can be identified. In relative terms this means 80% of the solution data (by bytes) can be tagged with extra computational metadata, and in absolute terms this still corresponds to 54% of the bytes. These results demonstrate that the methodology for extracting additional metadata from log files is highly successful, especially considering the fundamentally unstructured nature of the data.

The explicit association method discussed earlier in section 5.3 would be a significant improvement to the confidence that can be placed in the extracted metadata, but only if similar availability could be achieved. Since this research deals with existing data, in which explicit references appeared to be only very sparsely present at best, there is little that can be done to further this approach. This demonstrates that there is tremendous value in storing this type of association information along with the relevant metadata.

5.4.2 Computational Attributes

Computational attributes such as CPU cores, CPU time, and wall time describe the computational resources that were used to create a particular solution file. These attributes relate to the previous work in data cleaning in that they illustrate the cost of data creation. As such, these fields could potentially be very useful in the future development of more sophisticated data cleaning rules. However, this chapter considers only the availability and nature of these additional metadata.

The number of CPU cores is selected before a job is submitted, depending on best practice policy, and on the perceived size and urgency of the work. Histograms of the file count and total bytes by number of CPU cores are shown in figure 5.1. The number of CPU cores appears to fall within the range of 32 to 512 cores, with the peak at around 128 to 256 cores. This is to be expected given the nature of the work and the need to return results fairly promptly. Interestingly, the distribution for total bytes appears to be shifted slightly towards larger numbers of CPU cores. The difference is quite small, but this would suggest that jobs that are expected to be particularly large are allocated more CPU cores.

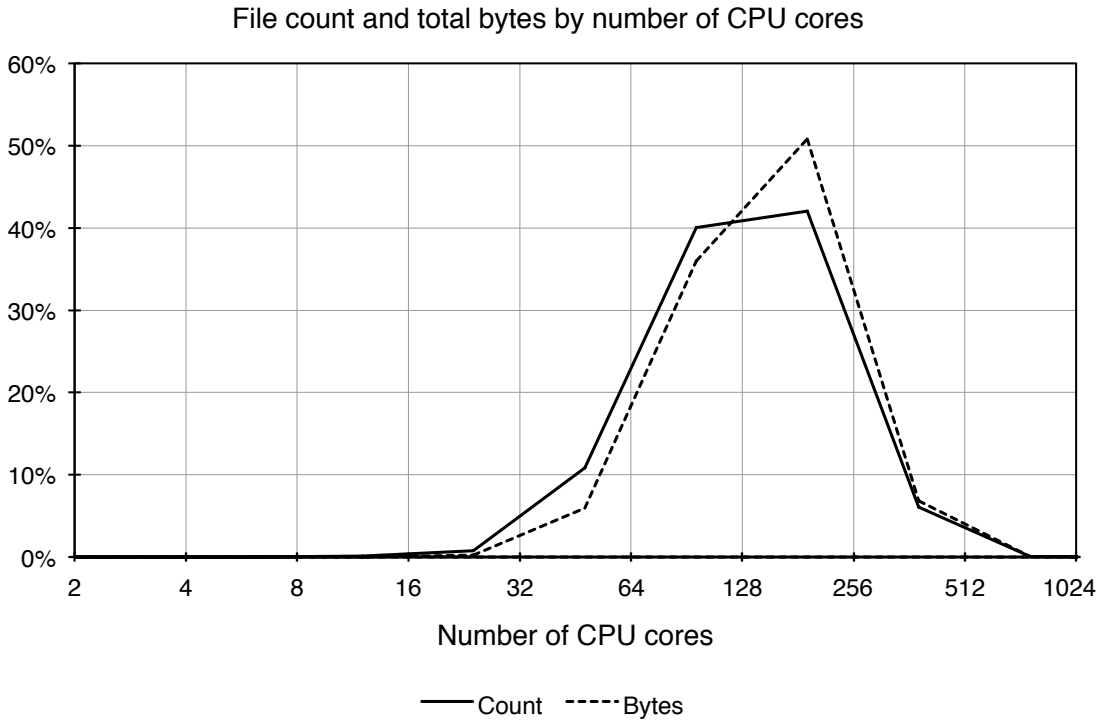

FIGURE 5.1: Histograms of file count and total bytes by number of CPU cores. Power-of-two bins on a logarithmic scale, based on metadata availability of 55.25% metadata by files and 80.26% by bytes, relative to the overall collection of 3D solution files.

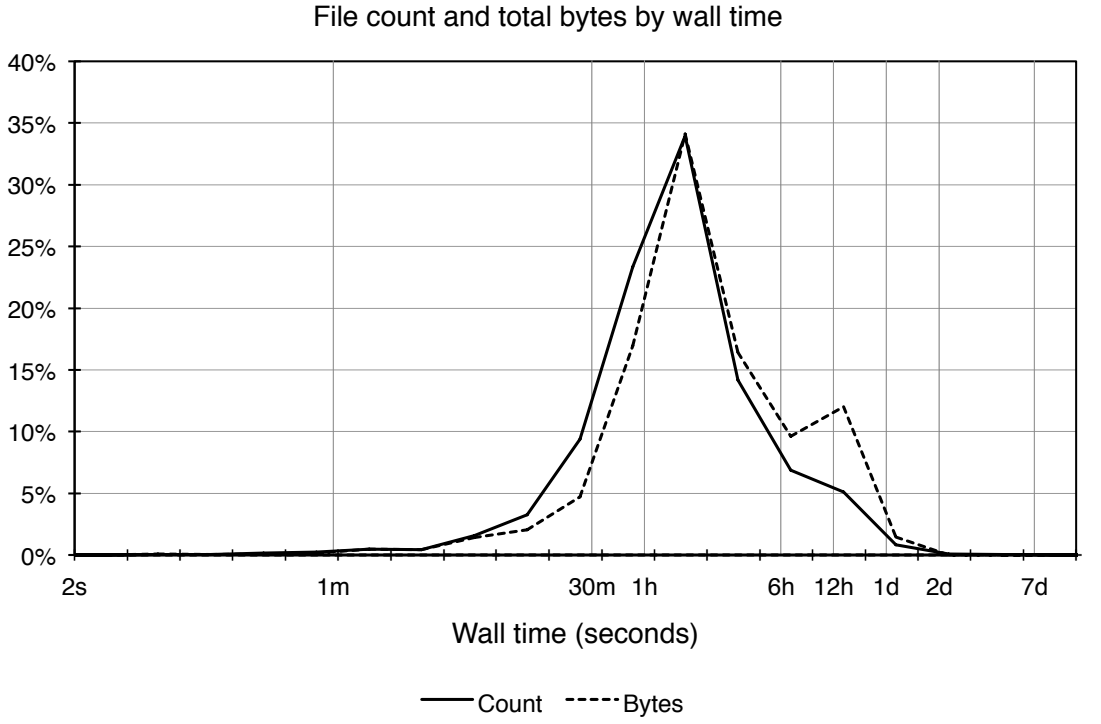

FIGURE 5.2: Histograms of file count and total bytes by wall time. Power-of-two bins on a logarithmic scale, based on metadata availability of 55.25% metadata by files and 80.26% by bytes, relative to the overall collection of 3D solution files. In order to improve comprehension over the large time scale, tick marks on the x-axis correspond to powers-of-two, while the vertical grid lines are placed at more recognisable locations, in minutes, hours, days, etc.

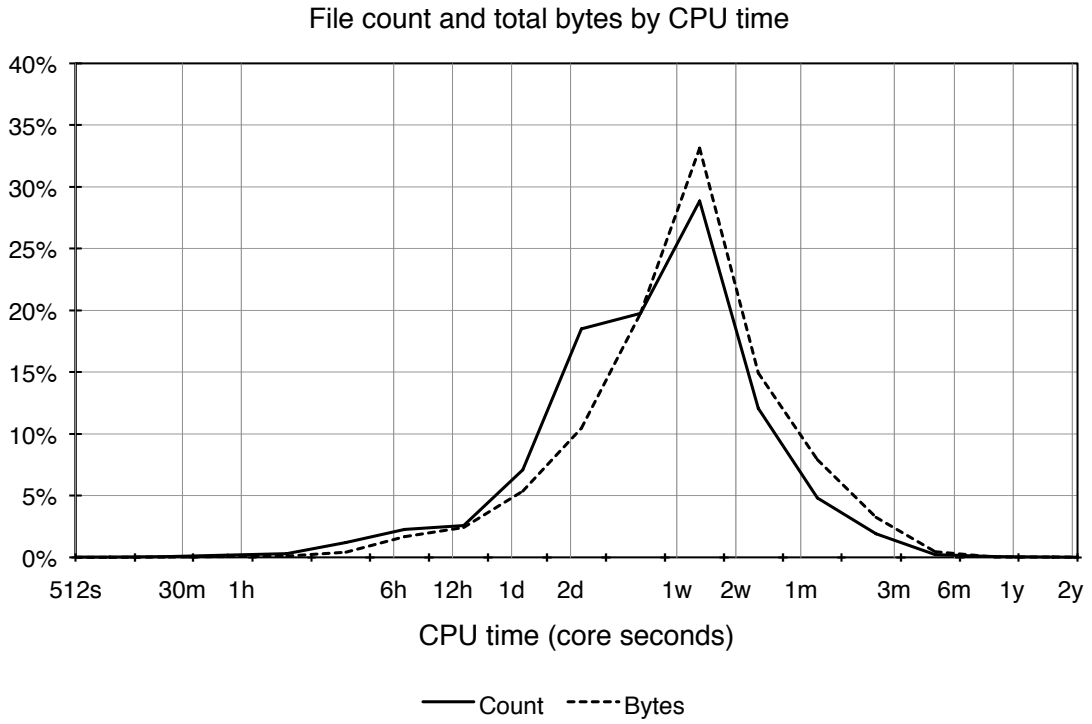

FIGURE 5.3: Histograms of file count and total bytes by CPU time. Power-of-two bins on a logarithmic scale, based on metadata availability of 55.25% metadata by files and 80.26% by bytes, relative to the overall collection of 3D solution files. See the caption of figure 5.2 for an explanation of the x-axis ticks and grid lines.

Figure 5.2 shows the histograms for file count and total bytes by execution time (wall time). The results show that most jobs are shorter than about one day, and many are shorter than six hours, although these figures do not take into account the time spent queueing. Although scheduling is beyond the scope of this chapter, the execution times are typically sufficient to meet the demands of a one-day turnaround time for CFD work (i.e. results to be returned within one day, in order to allow the greater workflow to progress). This is a common objective in aerodynamics, especially if results can be returned overnight.

As with figure 5.1, the distribution of total bytes appears to be shifted slightly towards longer execution times when compared with the distribution of solution files. Again, this indicates some evidence of some larger jobs taking longer to execute and producing slightly larger files. The overall difference is still quite small, but is particularly noticeable for the jobs with an execution time of 6 to 24 hours. This implies that this group of files corresponds to a different type of work, perhaps a larger component or assembly of components that requires longer to compute and produces larger files by virtue of covering a larger 3-dimensional space. Alternatively, it is possible that these files correspond to a different fidelity, and simply consider the same type of components and space but at a finer granularity.

It could be argued that the CPU time is a better single measure of computational cost,

since it takes into account both the number of cores and the time for which they were used. The distributions of file count and total bytes by CPU time are shown in figure 5.3. Again, the distribution of bytes appears to correspond to slightly higher values for CPU time than the file count. However, the difference is again very small, and unlike figure 5.2 there is no obvious grouping of larger and more computationally expensive files. This would suggest that the grouping previously observed in figure 5.2 may be the result of using fewer CPU cores.

As mentioned previously, the CPU time refers to the actual time utilisation of all CPU cores used for a particular job. In contrast, it is possible to calculate an alternative value by multiplying the execution time by the number of cores. This instead refers to the time for which CPU cores were allocated to the job, and thus could not be used by another job even if they were not being fully utilised. The distributions of file count and total bytes by this calculated value are shown in figure 5.4. It is clear that the ranges of values are similar to those shown for CPU time in figure 5.3.

The distribution of bytes is again noticeably shifted towards larger values in calculated CPU time than file count. The discrepancy between file count and total bytes is also substantially more pronounced than in the reported CPU time (figure 5.3). The distribution of file count is skewed in favour of lower calculated CPU time values, while the distribution of total bytes is skewed in favour of larger values. This would imply, as formerly suggested, that some larger files may have required greater CPU resources for creation, possibly due to differences in the size of the model geometry or alternatively due to the granularity of the mesh.

Lastly, the distributions of file count and total bytes by the number of iterations run by the job to achieve results with convergence to a given tolerance are shown in figure 5.5. The results show almost identical profiles for files and bytes, indicating that the number of iterations does not affect the size of the output files. It would be reasonable to expect that a tighter tolerance would not require the creation of larger files, since no additional data is required. Moreover, the single peak distributions imply that no changes to the required tolerance are present in the data, meaning that the number of iterations simply varies log-normally.

On the one hand it would be intuitive to expect that within one particular field broadly conducting the same type of work, jobs requiring more CPU resources should produce more data. On the other hand, two almost identical jobs may require different CPU resources simply due to the effects of aerodynamics on the convergence. This view is supported by figure 5.5, which shows very similar file count and total bytes distributions with the number of iterations. However, figures 5.1, 5.2, 5.3 and 5.4 do show some evidence to support the former argument in the distributions with CPU cores, wall time, and reported and calculated CPU time. Overall, there is clearly some truth in

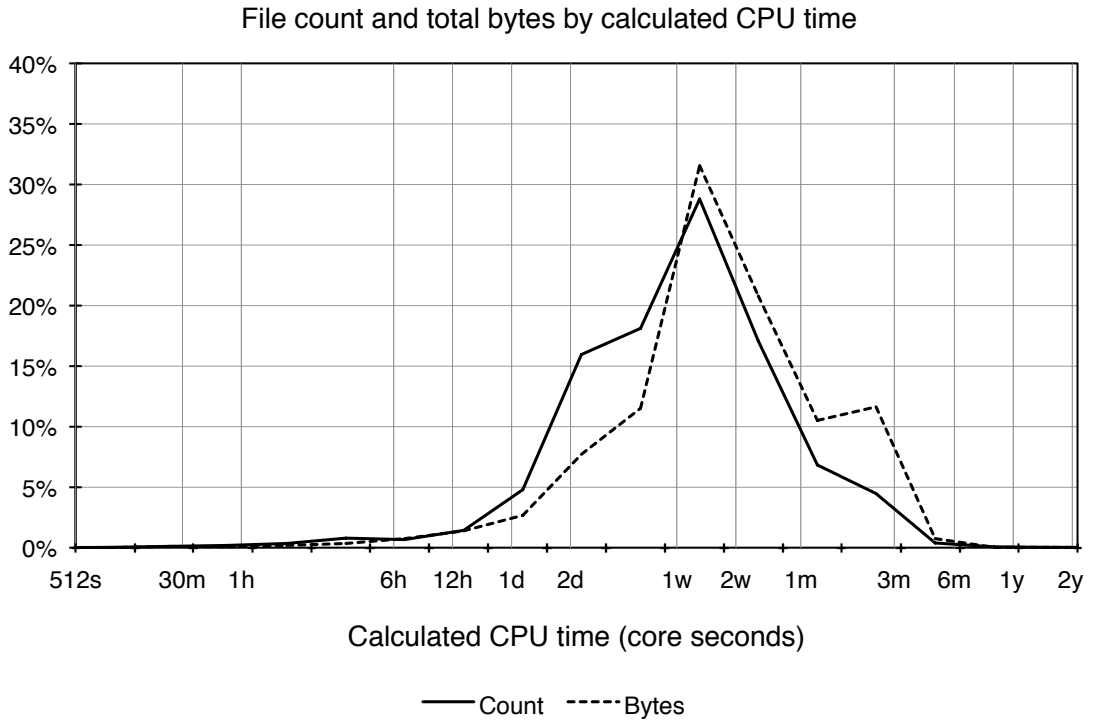

FIGURE 5.4: Histograms of file count and total bytes by CPU cores times wall time. This is similar to CPU time, but refers to the time CPU cores were allocated to a job, rather than the time for which they were actually used. Power-of-two bins on a logarithmic scale, based on metadata availability of 55.25% metadata by files and 80.26% by bytes, relative to the overall collection of 3D solution files. See the caption of figure 5.2 for an explanation of the x-axis ticks and grid lines.

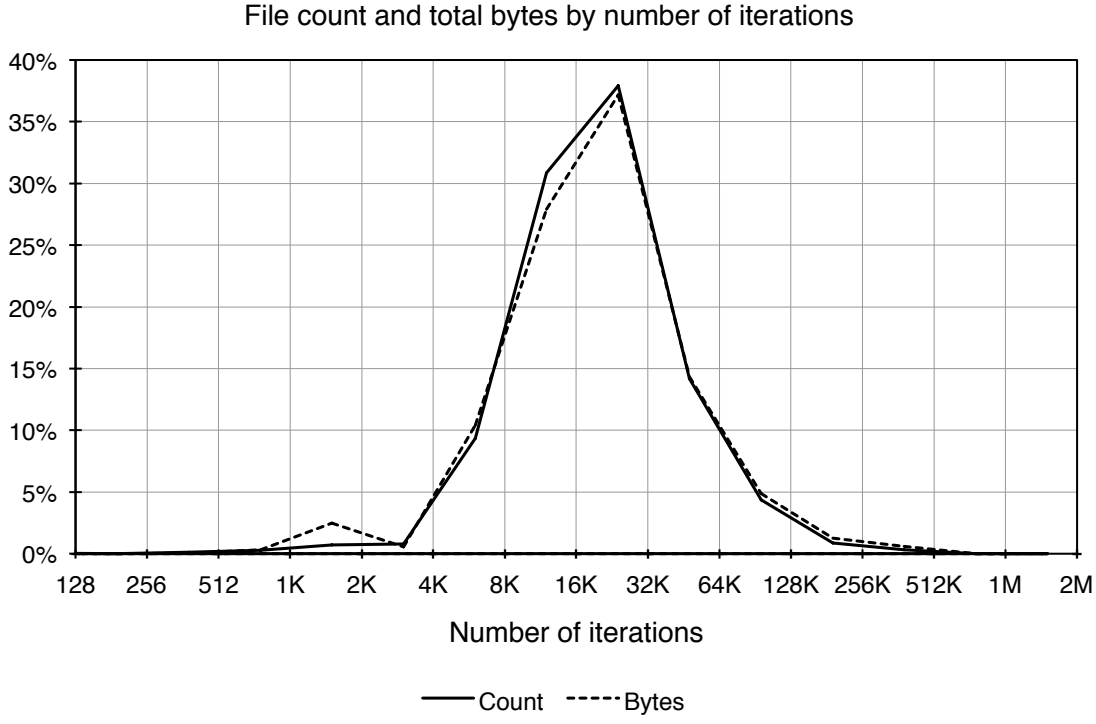

FIGURE 5.5: Histograms of file count and total bytes by number of iterations. Power-of-two bins on a logarithmic scale, based on metadata availability of 96.29% by files and 99.85% by bytes, relative to the overall collection of 3D solution files.

both arguments, and the more subtle groupings to the former simply suggest that the indirect effects of CPU resource usage on output file sizes are relatively minor.

5.4.3 Aerodynamic Attributes

The aerodynamic attributes (angle of attack, lift coefficient and Mach number) were all extracted from the file names, using regular expressions to impart information from common naming conventions. These metadata thus describe the nature of the data more in terms of the content, in contrast with the computational metadata that instead provide information about the cost of computation.

Figure 5.6 shows the distributions of file count and total bytes by angle of attack. It is clear that some very detailed experiments are conducted across a range of angles and at a very fine granularity. However, it is clearly much more common for such polar experiments to be conducted at a granularity of one or two degrees, as the results show noticeable peaks at these locations. This would certainly be a logical best practice, since detailed polars may have their uses, but are unlikely to be necessary in all cases.

In addition, there is some evidence that higher angles of attack correspond to larger file sizes, although the effect is very minor. It is certainly plausible that higher angles of attack occur in cases where more geometrically complex high-lift devices are included, and that the increased complexity corresponds to finer granularity, leading to increased file sizes.

Figure 5.7 shows the distributions of file count and bytes with Mach number. It is very clear that the majority of the solution files were computed for Mach numbers between 0.8 and 0.9, with the main peak at approximately 0.85. These values are typical for a commercial aircraft in a cruising state, and are the expected result. Similarly to the angle of attack, it is clear that Mach number values are often incremented by 0.1. However, a finer level of detail can be observed from Mach 0.7 to 0.9. This is presumably to examine the more subtle effects of different Mach numbers, but perhaps also in order to find an optimum cruise speed. Lastly, there appears to be no relationship between Mach number and file size, as the file count and total bytes distributions remain extremely similar throughout the range.

Figure 5.8 shows the distributions of file count and bytes with lift coefficient. In this case, the increments in lift coefficient appear to be less standardised. There are three obvious groupings between 0.4 and 0.7, but there are also short gaps between these groups. The reasons for this somewhat surprising result are unknown.

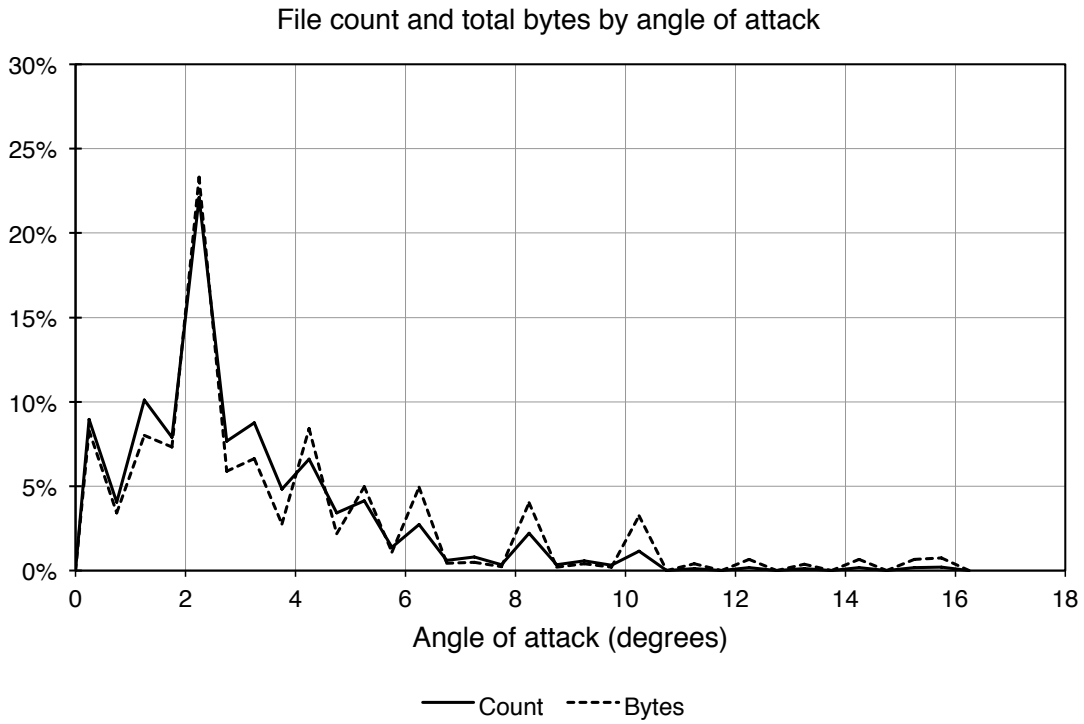

FIGURE 5.6: Histograms of file count and total bytes by angle of attack (degrees). Bins of width 0.5° on a linear scale, based on metadata availability of 21.08% by files and 29.91% by bytes, relative to the overall collection of 3D solution files.

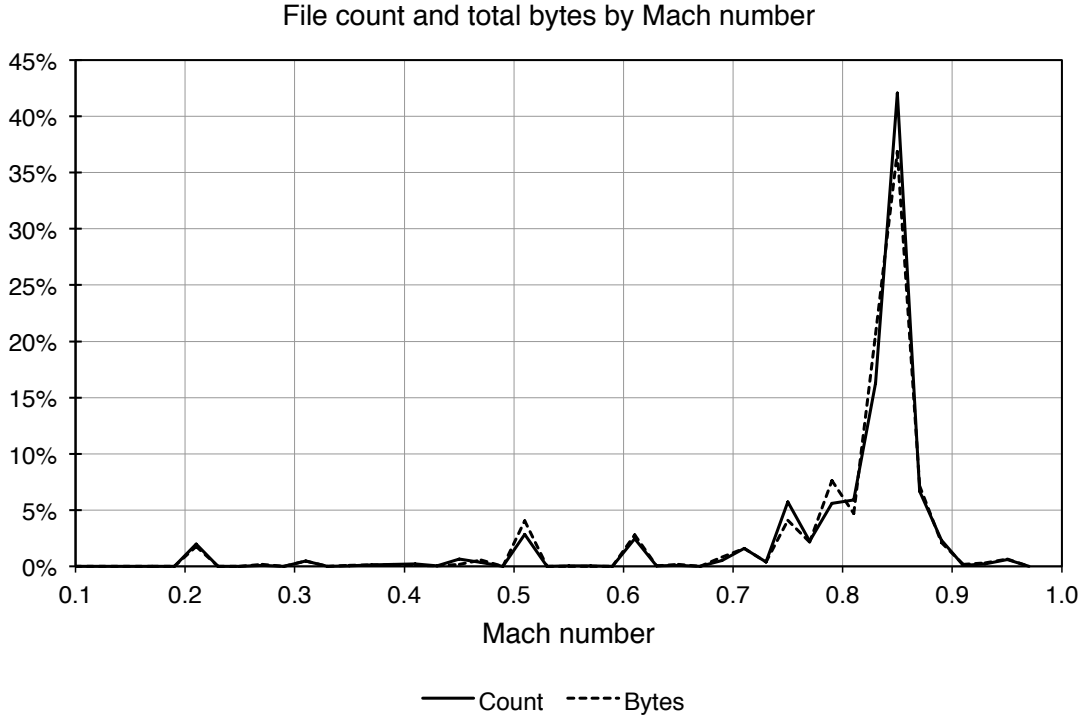

FIGURE 5.7: Histograms of file count and total bytes by Mach number. Bins of width 0.02 on a linear scale, based on metadata availability of 62.25% by file count and 62.05% by bytes, relative to the overall collection of 3D solution files.

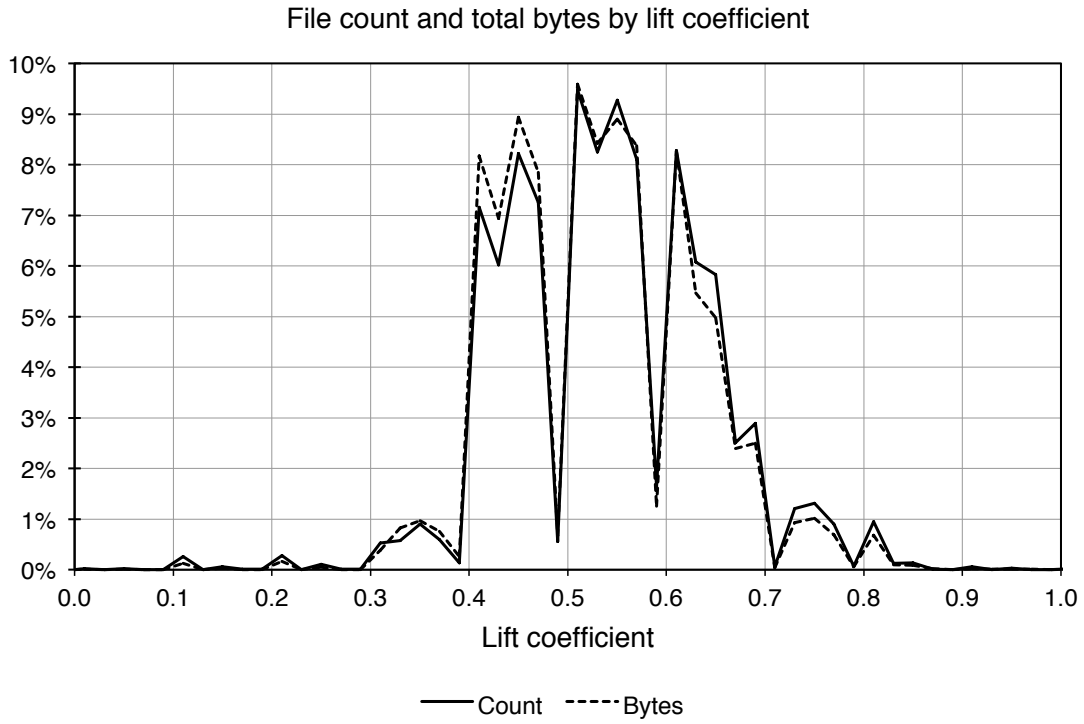

FIGURE 5.8: Histograms of file count and total bytes by lift coefficient. Bins of width 0.02 on a linear scale, based on metadata availability of 70.10% by file count and 63.45% by bytes, relative to the overall collection of 3D solution files.

5.5 Discussion

This section discusses the implications of the methodology and results for industry, some of the applications for which this research may be useful, and the limitations and improvements which could be made to better support such tasks.

5.5.1 Applications

Perhaps the key success of this research is the fact that a substantial quantity of metadata was successfully extracted from readily available sources. One of the key challenges in managing large volumes of data is that it can be very difficult to determine useful information about the content or value of different files. It is common for users to change roles and responsibilities, or to leave companies entirely, and under these circumstances it is particularly difficult to maintain knowledge about the purpose or content of data. Extracting and associating the type of metadata examined in this chapter provides exactly the type of information required to support more sophisticated data management operations, and is not dependent on user knowledge.

The combination of both aerodynamic and computational metadata is particularly useful in providing multiple options for viewing, sorting and categorising the parent files. As demonstrated previously, even basic file system metadata can be used to improve data management practices [61]. Business-specific metadata such as the aerodynamic fields examined in this chapter enhance the understanding of the data content. Similarly, computational metadata help to describe the value or cost of creation. Combining these metadata types provides greatly enhanced flexibility in drawing together these different aspects of data description.

The availability of this metadata provides a means to compare two files that may appear otherwise identical, particularly in terms of their inode metadata. This could be useful in some circumstances, such as when deciding which of two seemingly identical files should be retained and which should be deleted. Here, any additional information at all could be invaluable, particularly the computational type of metadata extracted from log files in this chapter , which describes the computational cost of data creation.

Additional metadata may also support data audit and exploration. On such a large scale, it may be useful to be able to search for data relating to a particular project, or created under certain flow conditions. The aerodynamic and project metadata extracted from naming conventions may be very useful in identifying particular files in this way, and could support improved data search and retrieval methods, and enhanced traceability in auditing the provenance of particular files.

Furthermore, the additional metadata extracted from both log files and file names (along with any other metadata sources that could also be integrated), may be used to produce detailed data analytics. Managers responsible for high performance computing systems and their associated storage systems may find it very useful to be able to examine, in detail, precisely how the data on storage systems relates to the jobs. This cross-examination of computing resources could help to develop a better understanding of HPC usage, and support improved fair-share policies. This is particularly significant in large scale industrial environments where the prioritisation of some tasks over others is an especially complex interdisciplinary task, often requiring human intervention.

5.5.2 Other Environments

Many other environments will also exhibit naming conventions and store similarly detailed log files. It follows that these other environments may be able to extract similar metadata in the same way as this chapter . The applicability of such a capability will depend on the nature of the data. If it is similarly unstructured, stored on a file system without a database, but exhibits similar structural characteristics, then extraction from the sources discussed in this chapter may be very useful. In contrast, well-structured storage systems may have no need for this type of metadata extraction, and the methodology would likely prove ineffective for totally unstructured data.

Overall, the usefulness of this particular methodology will depend on the nature and structure of any naming conventions and log files. As mentioned previously, the process-based data production in engineering design greatly improves the availability of this type of metadata, so other environments exhibiting significant automation and repetition in data production are likely to demonstrate the greatest success.

5.5.3 Limitations and Improvements

The metadata extracted in this chapter may be very useful for the many reasons already discussed. It has also been shown that the methodology is quite effective in providing additional metadata for a significant proportion of the actual data. As such, the methodology used in this paper provides a way to identify and associate metadata with much of the data on existing file systems without troubling users for any input.

However, it would be much more logical to store such information explicitly upon data creation. This could guarantee explicit references rather than implicit associations between data and metadata, and also provide information specific to individual files, rather than needing to apply values retrospectively to larger groups. Eliminating the need for these assumptions would greatly improve the accuracy and quality of the metadata.

Although storing metadata at the time of data creation is undoubtedly a fundamentally stronger and more flexible approach, the retrospective methodology used in this chapter still has tremendous value in that it can be used on existing systems. It may be much more difficult to guarantee the quality or accuracy of the information retrieved, but the ability to obtain useful metadata from existing systems provides a way to import legacy data into newer, and potentially more intelligent and sophisticated, data storage systems.

5.6 Conclusions

This section outlines the key findings made by this research; makes recommendations for the improvement of metadata storage in order to improve data management; and outlines some potential research areas that benefit from this work in the future.

5.6.1 Summary

This research extracted a substantial quantity of useful metadata from naming conventions and log files, which were then associated with a key file type (CFD 3D solution files) accounting for the majority of bytes within the file system. It was found that computational metadata from log files could be associated with 55% of these files, and 80% if calculated by bytes instead of by file count. The aerodynamic metadata from naming conventions demonstrated more variable availability, ranging from 21% to 96% by file count. The discovery, processing and tagging of files with these metadata provides greatly enhanced descriptive capabilities for these key files and their content, by incorporating both business and computational information.

Analysis of the values extracted revealed the characteristics of the data in terms of the computational resources used to execute the parent job, and in terms of the aerodynamic parameters configured as CFD flow conditions. The aerodynamic fields showed patterns corresponding to the types of sampling used in the configuration of larger parameter sweeps or polars. The computational metadata showed the typical values used in the setup of HPC jobs, but from the perspective of the resultant data rather than from the jobs.

5.6.2 Recommendations

The following recommendations can be made to improve the quality and availability of the types of additional metadata examined in this chapter :

- *Data-Metadata Creation.* Metadata tagging should be part of the main data creation process, since much of the desired metadata can be inherited from the parent workflow. As with existing data creation, metadata tagging could be largely automated, but the additional knowledge capture would be invaluable for future data management.

- *Data Collections.* Relationships between input, intermediate and output files should be detailed in a manifest, so that the collection of files may be managed together. This would avoid the need for many of the assumptions made in this chapter, thereby increasing the accuracy and certainty with which the information may be interpreted.

- *Metadata Storage.* Although the semi-structured sources examined in this chapter yielded useful results, the quality of the metadata would be greatly improved if fields were stored in a more structured format. For example, an XML file could be included for each collection of files in an unstructured storage system. This could also contain manifest information for file-metadata linking.

5.6.3 Future Work

One of the key research areas that stands to benefit from this research and its recommendations is that of graceful data degradation. Previous work demonstrated that simple file system metadata can be used to guide data cleaning processes [61]. However, more detailed metadata describing content and value could be used to extend this approach into a more graceful degradation process. A more gradual cleaning process could utilise compression techniques to reduce the size of data collections steadily over time, rather than simply removing files after a predetermined time threshold.

Similarly, additional metadata and more structured storage techniques could be used to help facilitate data regeneration. The solution files examined in this chapter are only intermediate results in the larger workflow, and the input and output files remain unaffected by the cleaning work demonstrated previously [61]. It follows that detailed information regarding input files could be combined with knowledge of the expected outputs to validate regenerated solution files. The regeneration of numerically identical output files is problematic due to the uncertain nature of distributed numerical methods, but the regeneration of suitably similar results for engineering purposes may be much more achievable.

These areas of future research are considered in more detail in chapter 6, which draws together the findings of all the studies conducted thus far. In particular, it delves into more depth regarding their feasibility and provides a more holistic view of the impact of the metadata findings discussed in each of the studies presented in this thesis.

Chapter 6

Discussion

Chapter 2 described the background, particularly the fields of data management, high performance computing, and engineering design. Within the context of aerodynamic engineering design at Airbus, chapter 3 then considered the characteristics and properties of file system metadata in engineering design, and chapter 4 demonstrated how these findings could be put to practical use in data cleaning. Following on, chapter 5 explored the identification and extraction of additional metadata from other sources such as computational log files and naming conventions.

The previous chapters each made their own findings and drew their own conclusions, with some of the earlier observations forming the basis for later work. However, the implications for some of these findings are broader than the individual studies, and merit separate discussion in greater depth. This chapter draws together some of the common thoughts and ideas presented in the previous chapters. It explores the feasibility and potential usefulness of regenerating deleted data from original inputs. It discusses the future of data management, particularly how the findings made thus far could be integrated and implemented with existing concepts and technologies. Finally, it considers the general applicability of the work carried out in this thesis to other environments beyond engineering design at Airbus.

6.1 Data Regeneration

Previous chapters have stated that in high performance computing (HPC) environments, it may be possible for deleted intermediate files to be regenerated from their original input files, and the accuracy of the regenerated results verified against the original output results. It is certainly possible to run the same experiment for the original input files, but the verification of similarity or equality of the results presents many challenges. This section discusses the latter.

6.1.1 Numerical Equality

The key issue in regenerating deleted intermediate files in this way is ensuring that the regenerating data matches the original data. Intuitively, it would seem reasonable to expect that identical data could be generated from the same inputs using the same process. Although there may be circumstances where this is the case, the complexity of many-core applications running on HPC systems means that this is much more difficult in practice. The problem is that although regenerated results may be the same for all practical purposes, this is not the same as being numerically identical.

For example, a simple way to compare the equality of two files may be to take a checksum or hash sum of the file contents. If the hash matches, then it is fair to say with reasonable certainty that the contents must also match. However, a very small difference between the files should result in different hashes for the two files, since they are no longer exact matches. The problem here is that the recreation of numerically identical results is very difficult, but that even the smallest discrepancy will prevent verification in a hash-based comparison.

Even in single-core computing, numerical inconsistencies can occur as a result of floating point precision. In the many-core environment of a HPC system, numerical errors can also be propagated from any inconsistencies in networking or synchronisation, as processing messages in a different order may yield slightly different results. Numerical errors such as these are unlikely to cause significant differences in the overall outputs. However, even trivial inconsistencies will preclude the use of hash-based verification methods.

Further numerical errors can be found in the convergence of a solution to a specified tolerance in an iterative numerical method. Although convergence can guarantee that a solution is no longer changing significantly with each iteration, it does not provide a mechanism to verify that two independently executed experiments will converge on the same value. Moreover, convergence to within a specified tolerance still only provides equality to within that tolerance, and does not guarantee numerically identical results. Again, this means that hash-based verification methods will be ineffective.

It is possible that a flow solution could be deliberately reduced in precision, considering only the few most significant figures at any point. A hash sum of this reduced-precision temporary data may be able to match data to within a reasonable tolerance, rather than requiring an exact match. However, this may be computationally costly to implement in practice, and still only mitigates mismatches caused by minor numerical errors and discrepancies.

6.1.2 Computational Fluid Dynamics

Since the suggestion was that intermediate files could be regenerated, it may then be possible to continue the workflow, execute any required post-processing, and compare the regenerated final outputs with the originals. In cases such as computational fluid dynamics (CFD), where the final outputs may only be a simple set of values, this final value-based verification may be very straightforward. For example, if the final computed values for the lift, drag and moment forces acting on a wing are equal, or even within a predefined tolerance, then in practical terms the two results may be considered equal.

However, although the final outputs can be verified as equal, this still does not guarantee any real similarity in the original and regenerated intermediate files. Verification of the equality of some key values may be useful for some engineering purposes, but an entirely different flow solution may be able to produce the same key values, so the visualisation of the solution may appear to be very different for the regenerated case. It follows that the regenerated data may be of limited usefulness if it can only be verified through these key values.

Moreover, the type of aerodynamic problem being solved may complicate equality verification at a much more fundamental level. In predictable cases such as steady, laminar flow, it would be reasonable to assume that regenerated results should be at least similar to the original files. However, in unsteady or turbulent cases, the CFD is attempting to model something inherently much less predictable, thus it is far less reasonable to assume any similarity in the regenerated results.

The usefulness of regenerating data is thus highly dependent on the circumstance and on the motivation. If it is possible to verify only key values in the output, then there is no guarantee that a visualisation will demonstrate the same flow features. Neither can it be guaranteed that any new experimentation will yield the same results from the new data as it would from the original data. Any regenerated data is thus unlikely to be fit for purposes such as certification, although it is perhaps equally unlikely that any data pertaining to such a process would be intentionally deleted in the first place. However, there are clearly some fundamental problems with the repeatability of CFD experiments, and further work would be required to properly evaluate how these issues could be mitigated.

6.1.3 Hardware and Software Evolution

Computer hardware is evolving rapidly, especially in high performance computing, which benefits greatly from any improvements not only in compute power, but also thermal and electrical efficiency. However this evolution comes at the price of consistency, since changes in architecture and processing may alter the way in which numerical calculations are performed. This aspect affects the floating-point precision problems already discussed.

Similarly, software is also changing fast, as new developments, algorithms and techniques reveal better ways to accomplish the same tasks and many more besides. However, this again causes problems with consistency, and two versions of the same software may produce different results for the same problem. These effects may be subtle, but at the cutting edge of technology may just as easily be very significant. This is particularly true in CFD, where accurate turbulence modelling remains an on-going problem.

Hardware emulation may be able to mitigate some of the problems associated with data regeneration on different hardware, although it may also impose a significant performance penalty. Similarly, software virtualisation may provide a means to quickly roll back to earlier software versions if more accurate data regeneration is required, although again there may be some associated performance costs. As such, although hardware and software versions pose some challenges, these may be less significant than the more fundamental aerodynamic and numerical issues already discussed.

6.1.4 Summary

In reality, regenerating data for intermediate files in a CFD context faces many challenges. Hardware and software versioning issues may be mitigated through emulation and virtualisation. However, the deeper problems with floating point precision and the more fundamental issues with unsteady and turbulent CFD cases still limit the viability of regeneration as a useful tool. Nonetheless, regeneration may provide increased flexibility, for example by enabling engineers to revisit previously deleted results, double check final outputs, and run new experiments on the regenerated data. More work is needed to assess the true extent of the limitations and challenges, but capturing more metadata may be very helpful in studying and potentially solving these issues. In particular, describing the provenance of the data with metadata such as the exact model, mesh, software version and settings, and hardware configuration used in data creation would be critical to any detailed research in this area.

6.2 Future Data Management

This thesis has provided several findings that have implications for future data management systems, particular in the chapters 4 "Practical Data Cleaning" and 5 "Additional Metadata". This section considers how these findings fit together, and how they may be used to enhance current and future data management systems. Although it focuses on the engineering design context in which most of this thesis has been based, it also outlines the applicability of the findings to other environments.

6.2.1 Metadata

Metadata is a hugely valuable resource for data management methods, capable of describing and characterising data in many different ways. Chapter 3 "File System Metadata" showed the characteristics of file system metadata in engineering design; chapter 4 "Practical Data Cleaning" demonstrated how these metadata can be used to identify and remove unwanted files; and chapter 5 "Additional Metadata" found and extracted a variety of business and technical metadata from log files and file names. Most significantly, this metadata extraction was all implemented using only general insight into the environment and context, and without interacting with the end-users.

Finding meaningful and accurate ways to categorise and describe data is important to the development of good data management methods. As mentioned previously, the most accurate way to capture this metadata is by doing so at the time of data creation. In highly computational data production, this metadata capture could easily be embedded into the creation process. Many useful fields, beyond those captured for this thesis, could be identified from the workflow. Moreover, many of these could be intercepted or extracted from existing sources, or included in common profiles for projects and particular job types. This minimises the required human input and ensures greater metadata availability.

After capturing valuable metadata, it is important to store it in an easily accessible format. Chapter 5 suggested that this metadata could be stored as XML in the file system, along with the relevant files. There are alternative ways in which metadata could be stored, particularly in more structured or sophisticated systems: the XML recommendation simply allows the metadata to be stored on existing NFS volumes alongside current data. However, a well-structured XML schema for this type of metadata could be later used to port the data to a more sophisticated system.

One of the biggest strengths of this thesis is that it was carried out exclusively on real data on current systems. This demonstrates that although a more proactive approach to capturing and storing metadata may be ultimately a more powerful approach, substantial metadata can still be extracted from existing data.

6.2.2 Tiered Storage and Collection Management

The same discussion of metadata storage also suggested that the stored metadata could include a manifest detailing links to related files. This could allow management of files by collection, rather than just individually. Since a job will create many closely related files this may be a much more natural way to treat some data management tasks, particularly data retention and degradation.

Chapter 4 "Practical Data Cleaning" considered the data retention policy for 3D solution files in a CFD context, and used this policy to perform data cleaning in order to save disk space. However, managing files in collections instead of individually could be very powerful in designing and implementing a more sophisticated cleaning or data management system. For example, the policy could be adapted to be cognizant of more of the metadata, dependent on more than just file age and file type, and similar files within a collection could be managed together. Files could also be compressed before being deleted, in more of a graceful data degradation scheme.

The work in this thesis considered data residing on file systems with and without offsite backup. However, data can be managed across many more tiers of storage, varying in their provision of security, redundancy and backup capabilities. For example, more frequently accessed data can be stored on faster, more expensive media, while less frequently accessed data can be stored on slower, cheaper media [44]. Conceptually this is very similar to the management of data between RAM and a CPU cache.

The automated management of tiered storage systems could potentially be based on more substantial information than simply the time of last access or modification. Conceptually, multiple tiers of storage and different levels of redundancy combine very naturally with graceful data degradation. Critical files may be stored accordingly, in order to protect against data loss, while files nearing the end of their predicted life may be moved to less backed-up systems for cheaper storage before being finally deleted. Coupled with collection management and a wide variety of metadata, this may be a very powerful data management technique, as collections may be viewed together for user purposes, treated together for data management purposes, but stored separately for cost-effective storage.

Additional metadata may be very useful in this type of system, because it could be used to develop more sophisticated data retention policies that consider more than just the age or type of a file. By also considering information pertaining to the parent project, flow conditions, and so on, it should be possible to develop policies that are much more reflective of the true data requirements.

6.2.3 Hierarchical and Search-Based Systems

It has been recognised that hierarchical file systems may no longer be the most effective way to organise files [70]. Users do not necessarily wish to place their data into virtual filing cabinets, and the next-generation of end-users increasingly favours search-based approaches to data retrieval. There is not necessarily an immediate need for all file systems to be converted or indexed for search-based data retrieval, but it is worth noting that semantic file systems support many analogous operations to hierarchical systems [47], but with the addition of greatly enhanced content-based retrieval [40] and potentially significant performance benefits [35].

The ability to search for data can be very powerful. However, the effectiveness of such an approach depends on the ability of a user to articulate their desired search terms, and on the ability of the system to identify useful information within its files [35]. It is thus imperative that a search-based system be provided with meaningful metadata to describe its files. Equally, these metadata must correlate with the type of search terms that a user is likely to require. As recommended in chapter 5 "Additional Metadata", it is worth considering how useful metadata might be inherited from the workflow and captured at the point of data creation, as this may provide a great deal of extra information, and much more accurately than a post-creation approach.

There are many benefits for end-users in using semantic file systems versus traditional hierarchical systems. Since a file can exist in many virtual directories, it is possible to create multiple views of the same data [56], in contrast with the single organisational structure in hierarchical systems. There is also support for file collections, where a file class may refer to several related files [29], which could aid in the type of graceful degradation discussed previously. In addition, there is the potential for greatly enhanced multidisciplinary collaboration through better data sharing and improved descriptive capabilities [26].

An important limitation in semantic file systems is that of backwards compatibility, which may be one of the greatest challenges preventing the more widespread adoption of such systems. Since existing applications rely on hierarchical namespaces, new file systems must be able to support legacy applications [40]. Furthermore, tighter integration with the workflow and applications can enhance metadata capture, but this will not benefit any legacy data. There is thus real value in tools that are capable of examining and extracting useful information about existing data, precisely as demonstrated in chapter 5 "Additional Metadata".

6.3 Applicability to Other Environments

This thesis has focused on the context of HPC for aerodynamics within Airbus. However, many other environments exhibit somewhat similar characteristics, and thus may be able to benefit from this work. The findings and recommendations may be applicable to many other engineering design environments that make extensive use of high performance computing (HPC), particularly those using computational fluid dynamics (CFD), including the aerospace, maritime and motorsport sectors. Some aspects may also be useful to non-engineering cases, depending on the particular circumstances.

It was mentioned in chapter 3 "File System Metadata" that the characteristics of file system metadata may be of interest to file system designers, software utility developers, and HPC managers. The metadata itself is useful in that it reveals characteristics and patterns in the nature of the data. For example, the file size and file type distributions led to the development of the cleaning methodology in the subsequent chapter, and the distribution of bytes with age allowed prediction of the time remaining until maximum capacity was reached. Developing a better understanding of the characteristics of the data being stored thus has useful implications for data management, and this applies anywhere that data management is taken seriously.

In addition, the comparison of Airbus trends and characteristics with those from other sources helps place Airbus HPC data in the wider context. It was demonstrated that there were many similarities with other HPC environments, although the Airbus data still showed some unique characteristics. Those unique characteristics may be of particular interest to file system designers, particularly those working with synthetic data, as the metadata distributions may be used to improve the modelling of this virtual data.

The data cleaning methodology demonstrated in chapter 4 was developed very specifically for Airbus. However, the approach used is only dependent on the presence of intermediate files that do not need to be retained indefinitely. As such, it may apply to any number of other environments that meet this simple criterion, although the significance of any savings made will depend on the context and nature of the specific files targeted. Other CFD-based workflows may find the method particularly useful, and would require the least modification in implementation.

Similarly, the methodology used to identify and extract additional metadata in chapter 5 was developed specifically for Airbus applications. The approach relies on log files and naming conventions to extract information, and hierarchical structure to associate metadata with files. These broad conditions may be well suited to highly computational workflows in many HPC environments. Again, any CFD context may be able to benefit from this type of metadata extraction, but any other environment with a suitably automated data generation workflow may also it useful.

Chapter 7

Conclusions

Finally, this chapter draws the thesis to a close. It describes how the specific research objectives laid out in chapter 1 have been met by the work detailed in the subsequent chapters; summarises the key findings made by the studies; and presents a some brief suggestions for the potential direction of future research in the field.

7.1 Fulfilment of Objectives

This section describes how the specific objectives of the thesis have been met, discussing the significance of each objective and outlining how it has been met by the research detailed throughout this thesis.

7.1.1 Airbus HPC Data Comparison & Contrast

> *"Examine large scale HPC data from industrial engineering design and compare and contrast with other data from HPC environments."*

It was important to understand the similarities and differences between Airbus file systems and those from other comparable environments in order to identify both the applicability of existing data managements techniques to Airbus data, and also the potential applicability of any newly developed methods and tools to non-Airbus systems. Many of these similarities and differences can be observed by examining the file system metadata, including the general characteristics of the actual data, but also some hints regarding the differences in workload.

The research undertaken to meet this goal was a crucial precursor to the later work in data cleaning and additional metadata, both in terms of the specific findings made through analysis of the data examined, and also in placing the challenges faced within Airbus into the wider context of data management in high performance computing. It also provided the data management community with a snapshot of the data characteristics from Airbus systems, which may be particularly valuable due to the limited availability of snapshot data for similar corporate HPC file systems.

Chapter 3 "File System Metadata" examined file system metadata from Airbus, Microsoft and a number of HPC sites, comparing properties such as file sizes, modificiation and access timestamps, directory structure and namespace depth. It was found that many of the metadata fields within the Airbus data set show similarities with the other environments. Analysis revealed the following specific similarities:

- As with many other data studies, most of the files are very small, but a few very large large files occupy most of the disk space. Most cases showed over 90% of files to be smaller than 4 MB.
- Directory and namespace organisation are fundamentally very similar, meaning that human organisational characteristics remain present in HPC environments.
- The proportion of CFD solution files on an HPC scale is similar to that previously observed in personal files [33], by both file count and total bytes.

However, there were also some cases where the Airbus data appeared to be in contrast with the other data sets. These notably unique characteristics are as follows:

- The file size distribution was notably more pronounced at Airbus than in other environments, with a very narrow band of file sizes including most of the bytes, and over 90% of the bytes in files larger than 4 MB.
- Files at Airbus are retained on active file systems for unusually long periods of time, often exceeding one year even on the no-backup (scratch) volumes. In other environments, files are instead either deleted or archived much sooner, often within six months.
- A few key file types account for a significantly higher proportion of the Airbus data than in other environments, which display much greater diversity. These few types are key input and output file types in the CFD workflow, and the top five types account for 64% of the files and 88% of the bytes.

7.1.2 Useful Characteristics for Data Management

> *"Identify metadata trends and characteristics that may be used to improve data management practices, particularly data cleaning."*

One of the challenges that led to the start of this research programme was the need to make better use of existing data storage devices, as the ever-increasing rates of data production from HPC facilities cause file system systems to fill up very rapidly. Although it is often possible to simply add more storage, this is no reason to be profligate or wasteful, thus it was important to identify any data characteristics that could be used to identify unwanted or redundant files.

Chapter 3 "File System Metadata" revealed several key trends and characteristics that may be used to improve data management practices. The following observations formed the basis for the subsequent work in practical data cleaning:

- File sizes between approximately 1 GB and 8 GB account for a very small proportion of the files, but a huge proportion of the total bytes.
- These files were identified by file type as the 3D solution files from the CFD workflow, accounting for 64.2% of the bytes in only 0.99% of the files.
- It was recognised that the 3D solution files are only temporary files in the larger workflow, and once fully post-processed could be safely removed.

7.1.3 Practical Data Cleaning

> *"Research practical data cleaning methodologies on real engineering file systems at Airbus, maximising savings in disk space whilst retaining all wanted data."*

Having examined basic file system metadata, chapter 4 "Practical Data Cleaning" then demonstrated that the observations made regarding the metadata trends and characteristics could be used to develop a methodology for data cleaning. Targeting old 3D solution files, an intermediate result in the CFD workflow, facilitated the cleaning of significant volumes of disk space. Following the development of this methodology, two cleaning exercises were undertaken on the Airbus-NoBackup volumes:

- The first exercise removed 3D solution files older than 24 months, and yielded 15.2% savings in disk space.
- The second exercise removed 3D solution files older than 18 months, and yielded 27.5% savings in disk space.

In addition to providing excellent savings in disk space, the methodology was developed to require only minimal human input, allowing the end-users to focus on engineering (the primary objective) rather than computing (a non-value-added distraction). The final approach provided a means to exclude unwilling participants, allowing users with sensitive data to remain unaffected by the data cleaning, thereby removing only unwanted data.

Lastly, careful consideration of the effectiveness, short-term impact and long-term sustainability of the cleaning method concluded the following:

- Ultimately, volumes will still become full. However, at the current rate of data production, continued usage of the developed method may extend the life of existing storage systems by up to 5 years before capacity expansion is required.
- Anecdotally, it was recognised that just by considering the time saved by engineers having to locate free disk space, the disk space savings made have already paid for the entirety of this research.

The research undertaken in data cleaning yielded substantial savings at Airbus in terms of disk space, human time, and also financially. Although the details of the methodology were specific to Airbus, the targeting of unwanted 3D solution files may be of benefit to other organisations working extensively with CFD, and the principle of removing intermediate files may be of use in any number of other environments.

7.1.4 Additional Metadata

> *"Identify and extract additional metadata from engineering file systems, using novel sources to learn more about the file content, purpose and value."*

The earlier research was based solely upon simple file system metadata properties such as file size and timestamps. However, there are other sources of metadata within the file system, particularly where the content of certain files can be examined and analysed for useful information. Chapter 5 "Additional Metadata" described the extraction of business, technical and computational metadata from additional sources, namely the conventions used in file naming, and the log files generated by computational jobs. Many fields were examined, incorporating computational metadata such as CPU time, cores, and the number of iterations; and aerodynamic metadata including Mach number, angle of attack and lift coefficient. The key findings were as follows:

- Additional metadata was demonstrated to be available for the majority of 3D solution files, which account for the majority of disk usage in the Airbus file systems examined. See table 5.1 for details.
- The aerodynamic metadata revealed the types of sampling used to perform large parameter sweeps, and conformed to the expected ranges of values for typical engineering applications.
- The computational metadata showed the typical configurations for CFD jobs within Airbus. In particular, it was observed that execution times often correspond to an overnight timescale, meaning that engineering work can be done during the day, and HPC jobs are executed in time for the next working day.

The significance of collecting additional metadata about key files is that the extra information could be used to further improve the effectiveness of data management methods, such as the cleaning methodology detailed in chapter 4. Using this information, it may be possible to begin to understand the value of individual files, which may be greatly beneficial to such tools. Access to additional metadata may also improve search-based data retrieval, and developing a better understanding of the relationships between different data could significantly impact data auditing, which is particularly important in engineering processes such as safety and certification.

Ultimately, a much more powerful approach would be to gather this type of metadata upon data creation, where a wide variety of information could be easily added, especially in highly automated environments. However, the ability to extract metadata from existing data means that new information can be inferred about existing files, allowing old data to be handled by newer, more sophisticated data management applications.

7.1.5 Recommendations

> *"Make recommendations regarding future data storage and management practices to Airbus (the sponsor) and to other practitioners in the field."*

The purpose of this reseach was to examine the valuable metadata that exists within engineering file systems and to consider how it could be used. Particular emphasis was placed on practically applying the findings to data cleaning, since this was one of the key challenges faced at the time. However, it is also important to consider the future direction of data management, within Airbus and the wider field.

The underlying principle behind the work in data cleaning was to only keep data for as long as it is required, and this principle can certainly be applied to any number of environments. Indeed, in the UK this is one of the guiding principles of the Data Protection Act [64], although this applies to personal rather than corporate data. A number of new suggestions were also made, and chapter 5 "Additional Metadata" in particular made several recommendations for the future management of data within Airbus. These recommendations are all suitably general rules to also apply to other practitioners in the field:

- *Data-Metadata Creation.* Metadata tagging should be part of the data creation process, since much of the desired metadata can be inherited from the parent workflow. As with existing data creation, metadata tagging could be largely automated, but the information captured could be invaluable for future data management.
- *Data Collections.* Relationships between input, intermediate and output files should be detailed in a manifest, so that the collection of files may be managed together. This would avoid the need for many of the assumptions made in this study, thereby increasing the accuracy and certainty with which the information may be interpreted.
- *Metadata Storage.* Although the semi-structured sources examined in this study yielded useful results, the quality of the metadata would be greatly improved if fields were stored in a more structured format. For example, an XML file could be included for each collection of files in an unstructured storage system. This could also contain manifest information for file-metadata linking.

These recommendations focus on capturing additional metadata at the point of data creation, and on storing this data in a more structured format in order to increase flexibility in supporting future data management processes. It is very difficult to anticipate all possible future requirements for such processes. However, it should certainly be possible to at least store some of the types of metadata that might be helpful in deducing meaningful information about the purpose, content and value of engineering data.

7.2 Future Direction

The studies and discussions presented in the thesis have made many references to potential applications and the possible directions for future research. There are many strong arguments for the tighter integration of data management with applications and workflow, particularly in the capture of valuable and accurate metadata at the point of data creation. These metadata could then be stored in a semantic file system on tiered storage systems, supporting enhanced content and context-based search, improved performance, and better data sharing and multidisciplinary collaboration. In addition to the automated movement of data between storage tiers, there could be further graceful data degradation, such as compression and deduplication, and the governing data retention methods could be made highly cognizant of the data context and purpose. It could be possible to integrate features controlling data visibility and accessibility subject to different user roles, identities and security clearance levels. Under some circumstances it may also be possible regenerate deleted data from original inputs, although the reliability and practical usefulness of the resulting data may be questionable. Some of these aspects are already well understood, and some are already implemented in existing products and services. The key point for future direction is the integration of all of these features into a single package. This must be suitably abstract for applicability to many other fields, but also suitably customisable for the specific purposes and requirements of the intended deployment environment.

In addition to directly benefiting Airbus through the work in data cleaning, this research may benefit other companies working extensive with CFD, or possibly with other highly computational workflows with similar characteristics. Highlighting the savings that can be made through data cleaning in particular may help prompt organisations to reconsider their own data management practices. The principles demonstrated and recommendations made throughout this thesis may then be useful in identifying how and where data management could be improved, especially regarding data cleaning where much of the emphasis of this research has been placed.

Data management is not typically a value-added activity, so it makes sense to minimise the time spent on it by most users. However, it can have a strong impact on working practice, especially in environments where a significant proportion of time is currently spent on on data administration. Intelligently capturing metadata from the workflow; automating the storage, archival and eventual deletion of data; and enhancing tools for search and retrieval may greatly reduce the time required for data management, allowing users to instead focus on value-added activities. It follows that the impact of better data management is increased productivity. Furthermore, as data sets become larger, and their management correspondingly increases in complexity, the impact of data management on productivity will become more and more significant.

Following on from increased productivity, the indirect effects of good data management are faster analysis through greater focus on value-added tasks; a cheaper computing infrastructure through more effective utilisation of existing resources; and better data auditing through more detailed metadata and better understanding of the relationships between different data. Again, the significance of these effects will only become greater as the size and complexity of data sets increases.

This thesis has focused on data management for HPC systems running CFD workflows to support the engineering design process. However, the same techniques could easily be applied to other workflows, such as stress analysis or multidiscplinary optimisation, and similar results may be achieved. Moreover, these techniques not limited to engineering design, and could be used to examine data management in non-engineering fields such as medicine and healthcare, meterorology, or any number of other environments where large data sets are prevalent. Although much of the emphasis here has been placed on data cleaning, the same approach could equally be used in archival in situations where long-term or permanent data retention is required but the data are less frequently accessed.

7.3 Summary

It is clear that data management is becoming increasingly crucial, and in this thesis we have considered just some of the ways that improvements in this field can be enacted. As we move forward into an era where data is recognised as being of significant importance for advancing science, engineering and business understanding, the implications of the research we have presented demonstrating the importance of and methods for managing data appropriately will become ever more critical.

Bibliography

[1] Adobe Systems Incorporated. Adobe Photoshop Lightroom 5. http://www.adobe.com/products/photoshop-lightroom.html, July 2014 (last accessed July 2014).

[2] Nitin Agrawal, William J. Bolosky, John R. Douceur, and Jacob R. Lorch. A five-year study of file-system metadata. *Trans. Storage*, 3(3):Article 9, October 2007.

[3] Airbus. Airbus, a leading aircraft manufacturer. http://www.airbus.com, July 2014 (last accessed July 2014).

[4] Amazon. AWS — Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting. http://aws.amazon.com/ec2/, July 2014 (last accessed July 2014).

[5] Eric Anderson. Capture, conversion, and analysis of an intense NFS workload. In *Proceedings of the 7th Conference on File and Storage Technologies*, FAST '09, pages 139–152, San Francisco, CA, USA, February 2009. USENIX Association.

[6] John D. Anderson. *Fundamentals of Aerodynamics*. McGraw Hill, 4th International edition, 2007.

[7] Anonymous. Extended attributes: The good, the not so good, the bad. http://www.lesbonscomptes.com/pages/extattrs.html, July 2014 (last accessed June 2015).

[8] Apple Inc. Introduction to Spotlight. https://developer.apple.com/library/mac/documentation/Carbon/Conceptual/MetadataIntro/MetadataIntro.html, August 2013 (last accessed June 2015).

[9] Apple Inc. Apple - Aperture - pro performance with iPhoto simplicity. http://www.apple.com/aperture, July 2014 (last accessed July 2014).

[10] Apple Inc. Apple - iMac - Performance. https://www.apple.com/imac/performance/, July 2014 (last accessed July 2014).

[11] Apple Inc. Apple - iTunes - everything you need to be entertained. `http://www.apple.com/itunes`, July 2014 (last accessed July 2014).

[12] Mary G. Baker, John H. Hartman, Michael D. Kupfer, Ken W. Shirriff, and John K. Ousterhout. Measurements of a distributed file system. *SIGOPS Oper. Syst. Rev.*, 25(5):198–212, September 1991.

[13] Alexander Ball. Review of data management lifecycle models. `http://opus.bath.ac.uk/28587/1/redm1rep120110ab10.pdf`, 2012 (last accessed June 2015).

[14] Bruce Barnett. Inodes - an Introduction. `http://www.grymoire.com/Unix/Inodes.html`, (last accessed June 2015).

[15] André Borrmann, Markus Schorr, Mathias Obergriesser, Yang Ji, I-Chen Wu, Willibald Günthner, Thomas Euringer, Ernst Rank, et al. Using product data management systems for civil engineering projects–potentials and obstacles. In *Proc. of the 2009 ASCE International Workshop on Computing in Civil Engineering. Austin, TX, USA*, 2009.

[16] Carnegie Mellon University. Parallel Data Lab. `http://www.pdl.cmu.edu`, April 2015 (last accessed June 2015).

[17] Carnegie Mellon University and University of Pittsburgh. Pittsburgh Supercomputing Center. `http://www.psc.edu`, June 2015 (last accessed June 2015).

[18] Yunus A. Çengel and John M. Cimbala. *Fluid Mechanics: Fundamentals and Applications*. McGraw Hill, 2006.

[19] Cornell University Library (CUL) Data Working Group (DaWG). Digital research data curation: Overview of issues, current activities, and opportunities for the cornell university library. `http://ecommons.cornell.edu/bitstream/1813/10903/1/DaWG_WP_final.pdf`, May 2008 (last accessed June 2015).

[20] Kevin Crowston and Jian Qin. A capability maturity model for scientific data management: Evidence from the literature. *Proceedings of the American Society for Information Science and Technology*, 48(1):1–9, 2011.

[21] Shobhit Dayal. Characterizing HEC storage systems at rest. Technical Report CMU-PDL-08-109, Parallel Data Lab, Carnegie Mellon University, Pittsburgh, PA, USA, July 2008.

[22] Tjark Derlien. Disk Inventory X. `http://www.derlien.com`, July 2014 (last accessed July 2014).

[23] DLR. DLR - TAU. `http://tau.dlr.de`, July 2014 (last accessed July 2014).

[24] John R. Douceur and William J. Bolosky. A large-scale study of file-system contents. In *Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '99, pages 59–70, Atlanta, Georgia, USA, May 1999. ACM.

[25] Allen Downey. The structural cause of file size distributions. *SIGMETRICS Perform. Eval. Rev.*, 29(1):328–329, June 2001.

[26] Oliver Eck and Dirk Schaefer. A semantic file system for integrated product data management. *Advanced Engineering Informatics*, 25(2):177–184, 2011.

[27] Eriban. GrandPerspective. `http://grandperspectiv.sourceforge.net`, July 2014 (last accessed July 2014).

[28] Kylie M Evans and Geoffrey H Kuenning. A study of irregularities in file-size distributions. In *Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, SPECTS '02, San Diego, CA, USA, 2002. SCS.

[29] Sebastian Faubel and Christian Kuschel. Towards semantic file system interfaces. In *International Semantic Web Conference*, page 60, 2008.

[30] Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley and Sons, 1st edition, 2008.

[31] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. *ACM SIGOPS Operating Systems Review*, 37(5):29–43, 2003.

[32] David K Gifford, Pierre Jouvelot, Mark A Sheldon, et al. Semantic file systems. *ACM SIGOPS Operating Systems Review*, 25(5):16–25, 1991.

[33] B. J. Hicks, A. Dong, R. Palmer, and H. C. Mcalpine. Organizing and managing personal electronic files: A mechanical engineer's perspective. *Trans. Inf. Syst.*, 26 (4):Article 23, September 2008.

[34] Yingjie Hu and Krzysztof Janowicz. Improving personal information management by integrating activities in the physical world with the semantic desktop. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 578–581, New York, NY, USA, 2012. ACM.

[35] Yu Hua, Hong Jiang, Yifeng Zhu, Dan Feng, and Lei Tian. Semantic-aware metadata organization paradigm in next-generation file systems. *IEEE Trans. Parallel Distrib. Syst.*, 23(2):337–344, February 2012.

[36] Stefan Hundhammer. KDirStat. `http://kdirstat.sourceforge.net`, September 2006 (last accessed July 2014).

[37] William Inmon, Bonnie O'Neil, and Lowell Fryman. *Business Metadata: Capturing Enterprise Knowledge*. Morgan Kaufmann, Burlington, MA, 2008.

[38] Andy J. Keane and Prasanth B. Nair. *Computational Approaches for Aerospace Design: The Pursuit of Excellence*. John Wiley and Sons, 1st edition, 2005.

[39] A. C. Kermode. *Mechanics of Flight*. Pearson, 11th edition, 2006.

[40] Andrew Leung, Aleatha Parker-Wood, and Ethan L. Miller. Copernicus: A scalable, high-performance semantic file system. Technical Report UCSC-SSRC-09-06, University of California, Santa Cruz, October 2009.

[41] Andrew W. Leung, Shankar Pasupathy, Garth Goodson, and Ethan L. Miller. Measurement and analysis of large-scale network file system workloads. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ATC'08, pages 213–226, Boston, Massachusetts, June 2008. USENIX Association.

[42] Los Alamos National Security, LLC. Los Alamos National Laboratory: National Security Science. `https://www.lanl.gov`, June 2015 (last accessed June 2015).

[43] Lustre. Lustre. `http://www.lustre.org`, July 2014 (last accessed July 2014).

[44] Stuart E. Madnick. Design of a general hierarchical storage system. Technical Report CISR-6, Massachusetts Institute of Technology, March 1975.

[45] Mallik Mahalingam, Chunqiang Tang, and Zhichen Xu. Towards a semantic, deep archival file system. In *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, FTDCS '03, pages 115–, Washington, DC, USA, 2003. IEEE Computer Society.

[46] P. Marsh. High productivity computing for engineering design optimisation (Proof of Concept II). Technical Report CFMS WP-15, Airbus and University of Southampton, 2008.

[47] Ben Martin. Formal concept analysis and semantic file systems. In Peter Eklund, editor, *Concept Lattices*, volume 2961 of *Lecture Notes in Computer Science*, pages 88–95. Springer Berlin Heidelberg, 2004.

[48] James Martin. *Managing the data-base environment.* Prentice Hall, 1st edition, 1983.

[49] Mike Mesnier, Gregory R Ganger, and Erik Riedel. Object-based storage. *Communications Magazine, IEEE*, 41(8):84–90, 2003.

[50] Dutch T. Meyer and William J. Bolosky. A study of practical deduplication. *Trans. Storage*, 7(4):Article 14, February 2012.

[51] Microsoft. Azure: Microsoft's Cloud Platform — Cloud Hosting — Cloud Services. `https://azure.microsoft.com`, July 2014 (last accessed July 2014).

[52] Microsoft. Windows Media Player. `http://windows.microsoft.com/en-GB/windows/windows-media-player`, July 2014 (last accessed July 2014).

[53] Microsoft. Delete files using Disk Cleanup - Windows Help. `http://windows.microsoft.com/en-us/windows/delete-files-using-disk-cleanup#delete-files-using-disk-cleanup=windows-8`, June 2015 (Last accessed June 2015).

[54] Microsoft. Windows Search - Microsoft Windows. `http://windows.microsoft.com/en-us/windows7/products/features/windows-search`, June 2015 (last accessed June 2015).

[55] Michael Mitzenmacher. Dynamic models for file sizes and double pareto distributions. *Internet Mathematics*, 1(3):305–333, 2003.

[56] Hung Ba Ngo, C. Bac, F. Silber-Chaussumier, and Thang Quyet Le. Towards ontology-based semantic file systems. In *Research, Innovation and Vision for the Future, 2007 IEEE International Conference on*, pages 8–13, March 2007.

[57] National Information Standards Organisations (NISO). Understanding metadata. `http://www.niso.org/publications/press/UnderstandingMetadata.pdf`, 2004.

[58] Office of Science and U.S. Department of Energy. NERSC: National Energy Research Scientific Computing Center. `https://www.nersc.gov`, June 2015 (last accessed June 2015).

[59] Onera. elsA. `http://elsa.onera.fr`, May 2014 (last accessed July 2014).

[60] Jonathan D. Owen and Simon J. Cox. Additional metadata in engineering design. Technical Report ME1448106, Airbus and University of Southampton, December 2014.

[61] Jonathan D. Owen and Simon J. Cox. Data cleaning in engineering design. Technical Report ME1448107, Airbus and University of Southampton, December 2014.

[62] Jonathan D. Owen and Simon J. Cox. File system metadata in engineering design. Technical Report D15017455, Airbus and University of Southampton, June 2015.

[63] Panasas. Panasas — high performance parallel storage for big data applications. `http://www.panasas.com`, July 2014 (last accessed July 2014).

[64] UK Parliament. Data Protection Act, Schedule 1, Section 1. `http://www.legislation.gov.uk/ukpga/1998/29/contents`, 1998 (last accessed July 2014).

[65] Swapnil Patil and Garth Gibson. Scale and concurrency of GIGA+: file system directories with millions of files. In *Proceedings of the 9th USENIX conference on File and storage technologies (FAST'11)*, FAST '11, pages 177–190, San Joe, CA, USA, February 2011. USENIX Association.

[66] Vijayan Prabhakaran, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Analysis and evolution of journaling file systems. In *USENIX Annual Technical Conference, General Track*, pages pages 105–120, 2005.

[67] Drew S Roselli, Jacob R Lorch, Thomas E Anderson, et al. A comparison of file system workloads. In *USENIX Annual Technical Conference, General Track*, ATC '00, pages 41–54, San Diego, CA, USA, June 2000. USENIX Association.

[68] Mahadev Satyanarayanan. A study of file sizes and functional lifetimes. *SIGOPS Operating Systems Review*, 15(5):96–108, December 1981.

[69] Bernhard Schandl and Bernhard Haslhofer. Files are siles: Extending file systems with semantic annotations. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 6(3):1–21, 2010.

[70] Margo Seltzer and Nicholas Murphy. Hierarchical file systems are dead. In *Proceedings of the 12th Conference on Hot Topics in Operating Systems*, HotOS'09, Monte Verità, Switzerland, May 2009. USENIX Association.

[71] Siemens PLM Software. Enabling innovation through enterprise data management. `http://m.plm.automation.siemens.com/en_us/Images/8937_tcm1224-38461.pdf`, 2010 (last accessed June 2015).

[72] Keith Smith and Margo Seltzer. File layout and file system performance. Technical Report TR-35-94, Harvard University, 1994.

[73] Software Ambience. DaisyDisk - analyze disk usage and free up disk space on mac. `http://www.daisydiskapp.com`, July 2014 (last accessed July 2014).

[74] Craig AN Soules and Gregory R Ganger. Connections: using context to enhance file search. In *ACM SIGOPS Operating Systems Review*, volume 39, pages 119–132. ACM, 2005.

[75] Standardization Committee. Standard of the camera & imaging products association, cipa dc-008-translation-2012, exchangeable image file format for digital still cameras: Exif version 2.3. `http://www.cipa.jp/std/documents/e/DC-008-2012_E.pdf`, December 2012 (last accessed July 2014).

[76] Storage Networking Industry Association (SNIA). Storage networking industry association: IOTTA repository. `http://iotta.snia.org`, July 2014 (last accessed July 2014).

[77] Andrew S. Tanenbaum, Jorrit N. Herder, and Herbert Bos. File size distribution on UNIX systems: Then and now. *SIGOPS Oper. Syst. Rev.*, 40(1):100–104, January 2006.

[78] Technische Universiteit Eindhoven. Technische Universiteit Eindhoven: Info SequoiaView. `http://w3.win.tue.nl/nl/onderzoek/onderzoek_informatica/visualization/sequoiaview/`, July 2014 (last accessed July 2014).

[79] TOP500. TOP500 supercomputing sites. `http://www.top500.org`, May 2014 (last accessed July 2014).

[80] Avishay Traeger, Erez Zadok, Nikolai Joukov, and Charles P. Wright. A nine year study of file system and storage benchmarking. *Trans. Storage*, 4(2):Article 5, May 2008.

[81] University of Alaska Fairbanks. Home — ARSC. http://www.arsc.edu/arsc, May 2015 (last accessed June 2015).

[82] U.S. Department of Energy. Pacific Northwest National Laboratory. http://www.pnnl.gov, May 2015 (last accessed June 2015).

[83] Yifan Wang. A statistical study for file system meta data on high performance computing sites. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, USA, May 2012.

[84] Qingsong Wei, Bharadwaj Veeravalli, and Zhixiang Li. Dynamic replication management for object-based storage system. In *Networking, Architecture and Storage (NAS), 2010 IEEE Fifth International Conference on*, pages 412–419. IEEE, 2010.

[85] Sage A Weil, Scott A Brandt, Ethan L Miller, Darrell DE Long, and Carlos Maltzahn. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 307–320. USENIX Association, 2006.

[86] WinDirStat. Windirstat - Windows Directory Statistics. http://windirstat.info, July 2014 (last accessed July 2014).

[87] Marcia Lei Zeng and Jian Qin. *Metadata*. Facet Publishing, London, UK, first UK edition, 2008.