Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH Open Access

# Clouds in Space: Scientific Computing using Windows Azure

Steven J Johnston*, Neil S O'Brien, Hugh G Lewis, Elizabeth E Hart, Adam White and Simon J Cox

**Abstract**

In this paper we report upon the cloud-based solution that we designed and implemented for space situational awareness. We begin by introducing the background to the work and to the area of space situational awareness. This concerns tracking the hundreds of thousands of known objects in near-Earth orbits, and determining where it is necessary for satellite operators to conduct collision-avoidance manoeuvres to protect their satellites. We also discuss active debris removal, which would be necessary to stabilise the debris population at current levels. We examine the strengths that cloud-based solutions offer in general and how these specifically fit to the challenges of space situational awareness, before describing the architecture we designed for this problem. We demonstrate the feasibility of solving the space situational awareness problem with a cloud-based architecture and note that as time goes on and debris levels rise due to future collisions, the inherent scalability offered by a cloud-based solution will be invaluable.

## Background

A variety of software and infrastructure solutions are referred to as cloud products, and although there is not a formal definition for cloud computing, the solutions tend to have much in common. Most cloud providers, such as Apple, Amazon, Google and Microsoft offer a pay-as-you-go pricing model for software and infrastructure, which is often referred to as a utility pricing model. Many cloud products offer a finished software solution rather than just infrastructure; for example, Microsoft, Google and Apple offer cloud based services, such as Hotmail, Gmail and iCloud respectively, directly to end users. The key cloud-based solutions can be divided into three categories: Infrastructure as a Service (IaaS) *e.g.* virtual machines, Platform as a Service (PaaS) *e.g.* a managed OS, and Software as a Service (SaaS) *e.g.* email services. The variety of available cloud-based architectures combined with a utility pricing model makes using a cloud-based architecture applicable to many scientific and engineering problems.

In this paper we use a case study from aerospace engineering to showcase the applicability of a cloud-based architecture. The case study looks at the issue of space situational awareness (SSA). SSA involves looking at near Earth objects and understanding the risk they pose to Earth. This has been highlighted in the news by many events including the Upper Atmosphere Research Satellite (UARS) re-entry in 2011, the International Space Station having to perform a collision-avoidance manoeuvre in 2010, and the collision between Iridium-33 and Cosmos-2251 in 2009. Figure 1 shows the orbits of these two satellites, and the debris produced by their collision.

The UARS NASA satellite was launched in 1991 as an orbital observatory and its mission was to study the Earth's atmosphere. UARS was decommissioned in 2005 and in 2010 the International Space Station had to perform a manoeuvre to avoid colliding with this debris. UARS gained considerable attention when it re-entered the Earth's atmosphere in 2011 with NASA predicting that large parts could reach the Earth's surface.

The "Clouds in Space" project demonstrated how a cloud-based architecture can be applied to SSA to produce an active debris removal solution. This paper begins by giving a more detailed introduction to the field of SSA, before discussing the strengths of cloud computing. The application of cloud-based architectures to SSA is then discussed in terms of these areas of strength. Next, we describe the cloud-based architecture that we designed for SSA. We then detail some of the observations made while architecting, implementing and demonstrating the solution, and finish with discussion and conclusions.

*Correspondence: sjj698@zepler.org
Faculty of Engineering and the Environment, University of Southampton, Southampton, UK

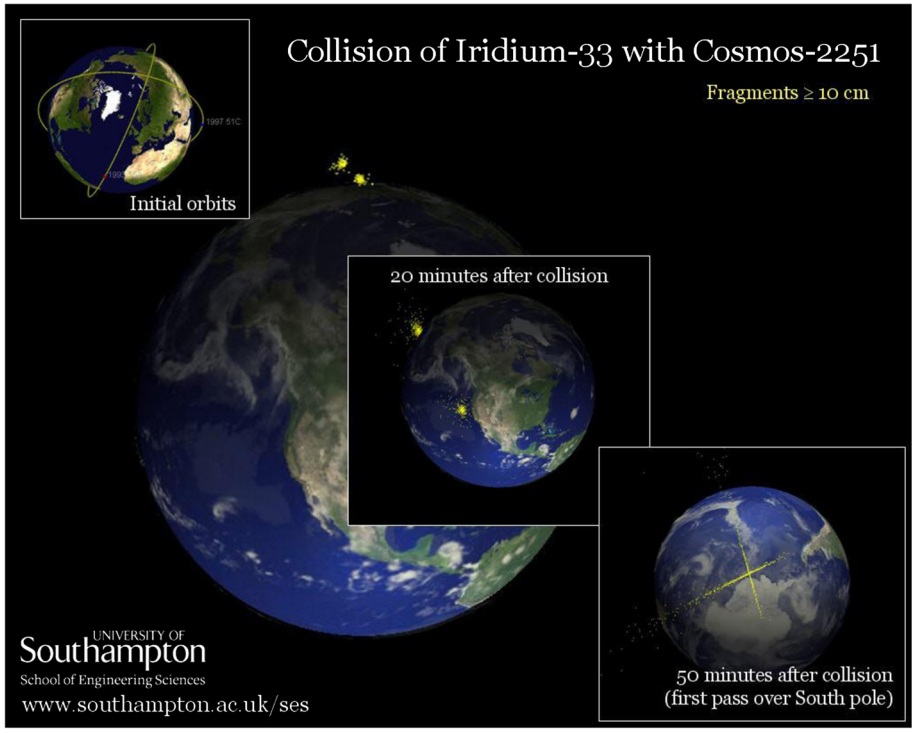


**Figure 1 Iridium-33 and Cosmos-2251 collision in 2009.**

## Space situational awareness

Within the last two decades, the downstream services provided by space-based assets have become a ubiquitous component of everyday life within the European Union and internationally, from satellite television and navigation to environmental monitoring. The European Space Agency (ESA) and European national space agencies currently rely on information from outside sources to form an awareness of these assets and the environment in which they operate. In the near future, this awareness will be provided by a European space situational awareness (SSA) system, which will provide "*a comprehensive knowledge, understanding and maintained awareness of the population of space objects, the space environment, and the existing threats and risks*"(User Expert Group of ESA SSA requirement study, 2007).

Through its SSA Programme (and its Preparatory Programme), ESA aims to provide key services and information regarding the space environment. The SSA system will comprise three main segments:

1. Space surveillance and tracking (SST) of man-made space objects,
2. Space weather (SWE) monitoring and forecasting,
3. Near-Earth object (NEO) surveillance and tracking.

The provision of timely, high quality data via the space surveillance and tracking segment is required to maintain an awareness of operational space assets as well as the population of debris objects in Earth orbit. This awareness provides key knowledge that supports space missions and includes the detection of conjunction events, the detection and characterisation of in-orbit fragmentations and the re-entry of risk objects. In addition, knowledge of overall space traffic is required to understand the evolution of the space (debris) environment and to support space debris mitigation and remediation activities.

Space debris represents a significant risk to satellite operations, particularly in the low Earth orbit (LEO) region. Approximately 19,000 objects larger than 10 cm are known to exist, with around 500,000 larger than 1 cm. The number of smaller particles likely exceeds tens of millions [1]. Conjunctions between satellite payloads and other catalogued objects occur at an average rate of 2,400 per day, with operators having to perform collision avoidance manoeuvres in cases where the risk cannot be reduced to an acceptable level by dedicated tracking campaigns [2]. Whilst mitigation guidelines have been adopted and measures implemented by space-faring nations, predictions made by computer models of the space debris environment indicate that the population of orbiting objects will continue to grow even in the absence of future space launches [3]. The remediation of the near-Earth space environment is now widely accepted as a requirement for the long-term, sustainable use of this vital

resource. A reliable and robust SSA infrastructure will be essential for the development and operation of any remediation technology.

The computational and data-intensive challenges presented by the requirements of a SSA system can be met using a cloud-based computational approach. In this work, we establish the applicability of a cloud-based architecture for space surveillance and tracking, algorithm development and comparison.

## Application of cloud computing to space situational awareness

In this section, we first introduce the strengths of cloud computing in general terms. We then go on to illustrate how the problem of space situational awareness is naturally suited to take advantage of all of the areas of strength that are inherent in a cloud-based architecture. We discuss how SSA could benefit both in terms of the available computational power and data storage opportunities offered by cloud providers, and how financial economies may be found by opting to use this approach rather than locally-provided data centres. This provides the background for the next section, in which we will introduce the architecture we designed for the SSA problem.

### Strengths of cloud computing

Cloud-based computing allows Internet-based resources, software, data and services to be provisioned on demand using a utility pricing model. Where solutions are architected for scalability, a cloud-based architecture can provide the ability to trade computation time against costs. This is readily applicable to applications that require frequent bursts of computational activity. Many individuals and businesses use cloud-based services for email, web searching, photo sharing and social networking. Scientists and engineers use a similar paradigm to make use of massive amounts of compute and data handling resources provided by companies such as Amazon, Microsoft and Google.

Central to a cloud-based architecture is the ability to purchase compute and storage resources using a flexible, on-demand billing model, much like the way traditional utilities (*e.g.* electricity) are purchased. This utility pricing model changes the way compute and storage can be exploited, encouraging scalable architectures and shifting the focus to almost unlimited, instant and on-demand resources with a direct monetary cost. Provisioning resources from a cloud provider is fast (typically taking times on the order of 1 min to 1 hour) and there is usually no minimum rental period, reducing or eliminating the need for large capital expenses as projects start-up or expand.

Cloud providers benefit from economies of scale; bulk purchasing hardware and electricity, and optimising machine administration. When combined with a flexible on-demand billing model, cloud providers can operate data centres very efficiently, in theory resulting in cost savings for end users. Owning and maintaining a data centre or cluster of machines is costly; hardware which is not being utilised is wasted (and probably wasting energy), so it is important to keep the hardware utilisation as high as possible to get best value from the hardware. Using cloud resources ensures that hardware utilisation is high, as unutilised resources can be returned to the provider (for use by others) and no longer incur a cost.

One of the key architecture patterns for cloud computing is to decouple a problem into independent discrete operations, and implement each with a *worker*. A worker consumes messages from a queue, completes the work stored in the message and then outputs a message to a different queue, as shown in Figure 2. Each message is a discrete piece of work which can result in data being created or consumed from storage (tables, SQL, blobs); the output message indicates work that has been completed and can easily become the input for another worker. This architecture is very flexible as workers can be reordered or substituted to achieve different objectives, or as a queue starts to get too long more workers of the same type can be created, speeding up the overall process. The key benefits of using a cloud-based architecture are described below [4].

- Data dissemination
  Cloud offerings are inherently global, highly available and have large bandwidth capabilities, making them ideal for data aggregation and dissemination. Often sharing data involves copying the data (perhaps multiple times) to ensure that the data and compute reside near each other; but using a cloud-based resource, sharing can be as simple as changing access permissions. Once a dataset resides in a globally accessible cloud resource it too becomes a valuable resource [5] suitable for third party data mashups [6,7]. The data owner can provide access to a third party, who can purchase compute resources with the same cloud provider and immediately start processing the data set. The data owner is responsible for data storage costs but the third party is responsible for their own computational resource costs.
- Burst capability
  Figure 3 shows how a data centre copes with predictable demand (top left) and unpredictable demand (top right). When sizing a data centre for such a scenario it has to be able to cope with the peak load; for the majority of the time this hardware remains unused. Where the data centre can cope with demand, the end user applications are unaffected. Once the demand exceeds the capability of the data

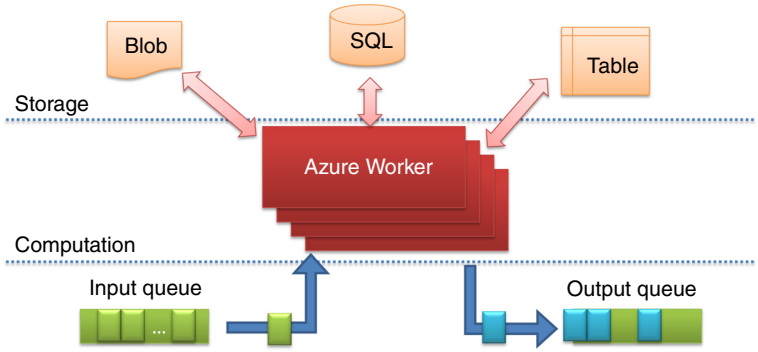


**Figure 2 Windows Azure worker architecture pattern.** Workers consume messages from input queues, and write data to storage systems and output queues. This pattern enables a dynamic number of workers to process queue messages.

centre, the under-resourced demand has a negative impact on the end user application. Moving such an application to a cloud provider ensures that you only incur costs for the resources utilised, whilst enabling rapid scaling to deal with a variable demand level.

- Super-Scalability
  It is difficult to judge the demand of an application, so there is an inherent need to make applications scalable. In addition to the application scaling the underlying resources need to scale. As with the burst capability, cloud computing offers near-instant scalability (quicker than purchasing physical machines [8]) allowing an application to scale beyond what is easily achievable with in-house data centres, as shown in Figure 3. In addition, as an application workload declines or plateaus, cloud computing permits the scaling back of resources; currently this is very difficult to accomplish with physical hardware.
- Algorithm development
  Procuring hardware on-demand ensures that the most appropriate hardware is utilised throughout algorithm development and validation. Where a test

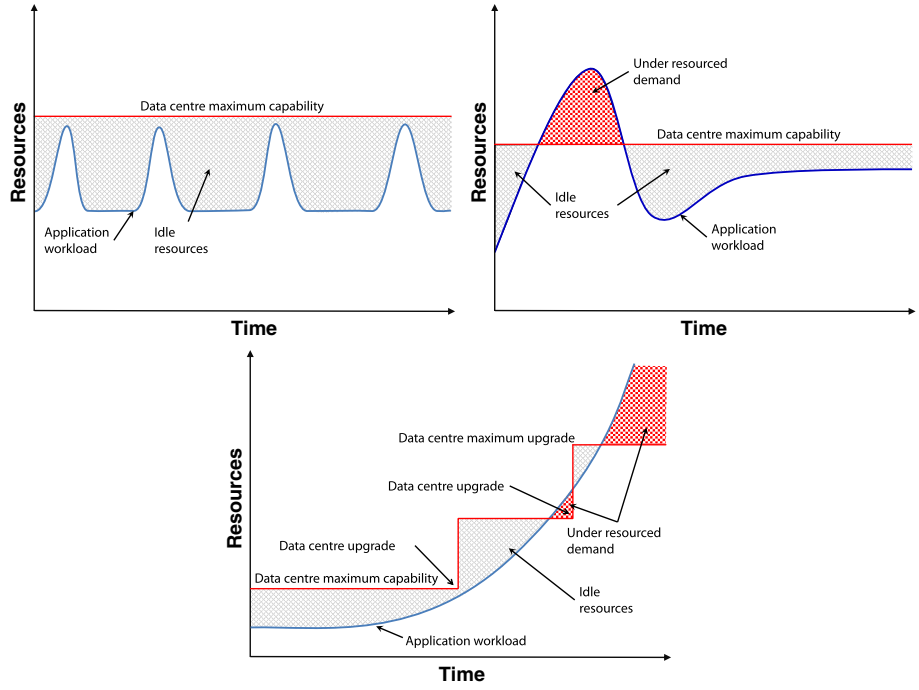


**Figure 3 Available resource and demand curves for several data centre scenarios.** Under utilisation of compute resources for predictable demand (top left), unpredictable demand with insufficient resources available (top right) and scaling of an application and the data centre that hosts it, with alternating periods of excess and insufficient available resources (lower) [4].

or validation dataset becomes sufficiently large or requires large computational resources a cloud-based architecture can reduce the overall algorithm development time. Cloud-based architectures also encourage a modular design which is ideal for comparing different algorithms as they can be run side-by-side and output data can easily be compared [9].

### Applicability of cloud computing to space situational awareness

In the previous section, we highlighted four key benefits of using a cloud-based architecture in general. Now we will discuss in more detail how the SSA challenge specifically fits into these categories.

- Data dissemination
  ESA currently depends on the space catalogue provided by the US Department of Defence (DoD), through its Space Surveillance Network (SSN), for an awareness of space objects. The DoD catalogue contains orbital data for all objects $> 10\text{cm}$ and for some objects $> 5\,\text{cm}$ (approximately 20000 objects). An independent, European catalogue, derived from measurements using European sensor systems, is likely to provide similar capability. However, as new detection hardware is incorporated in the future, the increase in sensitivity will result in a several-fold increase in the number of catalogued objects. The catalogue will also increase in size as space launches are sustained and fragmentation events continue (albeit at a reduced rate as a result of mitigation measures). In particular, collisions between large, intact objects are likely to generate several thousand fragments larger than 10 cm and tens of thousands of fragments larger than 1 cm. For example, the Iridium 33-Cosmos 2251 collision in February 2009 added 1900 objects to the catalogue whilst more than 2000 debris of the order 10 cm or larger were identified by the US SSN in the year following the intentional destruction of the Fengyun 1-C satellite in January 2007 [10]. Whilst a significant number of conjunctions between space objects involve intact spacecraft, nearly half of all conjunctions occurring in August 2009 involved debris from these recent major fragmentation events (Figure 4), illustrating the importance of the timely detection and characterisation of break-ups [2]. A cloud-based storage solution could offer an excellent way to store this increasingly large amount of data. The advantage of storing in a cloud-based resource is the ability to share data between trusted partners and to co-locate data and compute. For example, this could enable satellite operators to securely share precise orbital data and to understand possible conjunctions whilst each person pays for their own storage and compute requirements.
- Burst capability
  Every object in the debris catalogue requires processing (e.g. for conjunction analysis) and as the catalogue grows, the demand for computational power increases. New launches increase the catalogue size in a predictable manner but conjunctions can unpredictably add thousands of new objects, then as the debris orbits decay, the number of entries reduces. A cloud-based architecture would facilitate the rapid procurement of processing power to process the debris orbital data and the characterisation of the conjunction event in a timely fashion. This is a fundamental component of the SST segment. As debris in the catalogue decays out of orbit, excess computational resources can be released, thus not incurring a cost. The burst capability of a cloud-based architecture offers rapid expansion and reduction of computational resources making it ideal for scenarios such as SSA.
- Super-Scalability
  The current debris catalogue size is limited by the ability to track distant or small objects. As detection methods improve we can expect to track a wider range of debris. This will vastly increase the debris catalogue. Currently the catalogue contains approximately 20000 objects but there are millions of objects that could be tracked [1]. This ability to purchase additional compute power in a flexible way means that a cloud-based infrastructure can be scaled to provide a continuity of awareness as the population of space objects and the SST measurement hardware evolve over time.

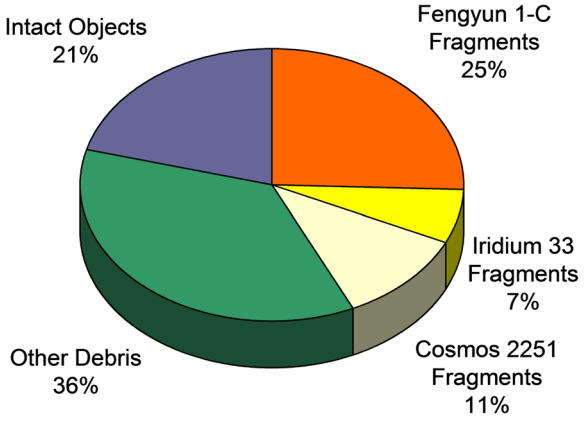


**Figure 4 Contribution of recent breakup fragments to close approaches.** (data: Centre for Space Standards and Innovation, generated at 13:33 on 15 August 2009).

- Algorithm development
  The development, verification and tuning of complex SST algorithms can be accelerated using using cloud-based technologies. *e.g.* running two different propagators to predict orbits side by side and then comparing the output or comparing collision probability assessment algorithms. Developing such algorithms is an active area of research (see, *e.g.* [11-13]).

The requirement to monitor compliance with space debris mitigation guidelines and the increasingly strong focus on the remediation of the near-Earth environment within the space community will require considerable support in the form of an awareness of overall space traffic. In this way, SSA has a key role to play. The ability of a cloud-based architecture to combine this service-oriented infrastructure with support for research and algorithm development offers a way to generate future space debris and space traffic solutions in a manner that is consistent, reliable and allows for full international collaboration. The development of active debris removal (ADR) technologies and the design of ADR missions fit within this paradigm. As such, we have selected the development of an algorithm for optimising ADR mission delta-*v*s as an illustration of the utility of a cloud-based computing approach.

#### A cloud-based architecture solution to the SSA case study

Recent computer modelling studies have suggested that the LEO debris population may be stabilised at current levels through the removal of five large, intact objects per year [14]. Whilst this approach can only be successful if the objects that are targeted would otherwise contribute to future collision activity, it does provide a more cost-effective approach to remediation than the removal, en masse, of all debris objects. However, this leads to a requirement that future collisions are forecast to a sufficient accuracy. In addition, to limit the generation of more debris and to reduce costs further, it is likely that an ADR mission will aim to remove more than one debris object. Consequently, mission requirements include orbital transfers between targets in addition to manoeuvres in close proximity to these uncontrolled objects.

In the light of these requirements, a key concern in the design of an ADR mission will arise from the choice of propulsion system. The choice will be determined, in part, by the energy required to remove debris targets from orbit and to transfer to subsequent targets. The required energy also provides an additional constraint on the selection of removal targets, as it is also linked to mission cost, such that the determination of the route between target destinations becomes an important optimisation task in ADR mission design. This optimisation problem, known as the travelling purchaser problem (TPP), forms the basis of the demonstration of a cloud-based computing approach. Figure 5 shows the cloud-based architecture for the example ADR mission. The architecture is implemented on Microsoft Windows Azure and each numbered block in the figure is a worker type which can be launched as multiple instances if required.

In our example, an ADR mission with a chemical propulsion system performs a rendezvous manoeuvre to attach a solid rocket motor to a target object, which subsequently fires under remote command to de-orbit the target. The ADR vehicle then uses its primary chemical propulsion to transfer to the next target. Removal targets are identified and ranked using a fast, pair-wise collision algorithm based on the Cube approach employed by the LEO-to-GEO Environment Debris Model (LEGEND) [15] and applied to all objects in the US SSN catalogue. The approach determines the collision probability for each object using Monte Carlo (MC) simulation, whereby the number of MC samples effectively determines the amount of compute time required within the cloud-based solution.

An overview of the ADR workflow architecture is shown in Figure 5. The entire workflow is a single instance of the ADR analysis for a given point in time and it is possible to run multiple workflows in parallel; they do not require inter-process communications. In this paper we only run a single ADR instance, which is comprised of seven different worker types. The workers are synchronised using cloud based queues to identify which unit of work requires processing, and all the data is stored in cloud storage. The storage and queues are designed to be super scalable and are part of the cloud fabric. A cloud based architecture affords us the ability to vary the number of worker instances dynamically, thus we can easily add more hardware to speed up parallel tasks. This is not the case for all workers as some are single instance, shown as a single worker in Figure 5. For example the 1$^{st}$ worker is a data importer that monitors a particular location for new TLE data and therefore only requires one worker. If there were multiple TLE sources it would be possible to run multiple workers to import the data.

The 2$^{nd}$ worker in the ADR architecture is the main propagator and collision detector, which consumes a full TLE catalogue and runs an MC simulation to calculate the probability of a collision between each piece of debris. The bespoke numerical orbital propagator features Earth gravity harmonics up to order 20, solar radiation pressure, luni-solar and atmospheric drag perturbations (using the NRLMSIS-00 atmospheric model) [2]. The propagation and collision algorithms are implemented as a single worker within the cloud-based architecture so that multiple instances (multiple MC samples) can be created for each debris pair. Propagation and collision detection are

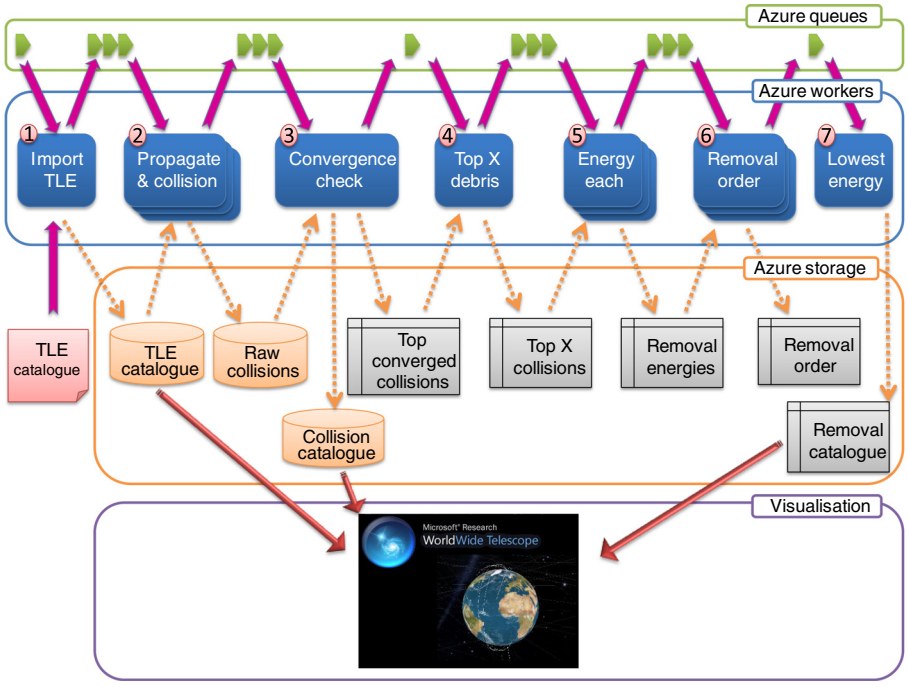


**Figure 5 Windows Azure active debris removal architecture.** Worker types and sample output data for visualisation. Each worker can have multiple instances demonstrating the scalability and burst capability of a cloud-based architecture.

the main computational workload and are used to identify the debris which has the potential to cause the most disruption in the future. Many instances of this worker are run in parallel, each outputting an ordered list of debris and probability of collision.

The 3rd worker reads these lists of collision probabilities and checks to see when the order of debris in the collision list has stabilised, at which point the MC simulation has converged with the most problematic debris at the top of the list. The 4th worker is a single instance worker that identifies the top ranked objects according to collision probability (and other physical characteristics), normally around 10 pieces of debris. This table of debris is consumed by the 5th multiple instance worker, which computes the delta-$v$ required by each solid rocket motor to de-orbit a selected target object, the optimum route between target objects, and the delta-$v$ required to transfer between these objects. In our preliminary implementation, the TPP is solved using a 'brute force' approach whereby the delta-$v$s required for every route permutation are calculated by the 6th worker, and we assume Hohmann transfers are employed. The 7th and final worker outputs a list of problematic debris as well as the removal order which requires the lowest energy.

A Hohmann transfer is a transition between two coplanar circular orbits of different altitudes, first described by Walter Hohmann in 1925 [16]. The manoeuvre is accomplished by firing a spacecraft's engine to accelerate it from the first circular orbit into an elliptical orbit, chosen to touch both the initial and destination circular orbits. At the intercept between the transfer orbit and the destination orbit, the engine is fired again to accelerate the spacecraft into a circular orbit. To transfer to a larger circular orbit the acceleration is applied along the spacecraft's current direction of travel; to transfer to a smaller orbit, it is in the opposite direction.

The ADR architecture shown in Figure 5 generates data which is stored in cloud-storage. Accessing the raw data from cloud-storage is trivial and we utilise World Wide Telescope (WWT) [17] to visualise the input, output and intermediate files. WWT has a rich API which supports importing data via a REST interface or from Excel, and is used to visualise data directly from Windows Azure as shown in Figure 6.

In Figure 5 the workers depicted with a single block are single instance workers, where as those with multiple blocks are parallel workers. The propagation and collision worker consumes the largest computational resources but is highly parallel, however the convergence checker cannot start until the propagator has completed. Currently, convergence is checked after the propagator has run for a set number of times, but future implementations will run a convergence checker which can terminate the propagation once converged (to save computational resources).


**Figure 6 World Wide Telescope displaying a full TLE catalogue.**

This limits the minimum computational wall clock time: if each propagation and collision MC simulation was run with its own worker, the minimum time to complete a full run is bounded below by the time taken by a single propagation (this could be accelerated by using more powerful hardware). The 'energy each' and 'removal order' workers have to wait for the top debris list, but can then process the entries in parallel. Using cloud storage and queues reduces the communication bottlenecks and failure overheads as they are transactional and fault-tolerant.

### Cloud observations

The SSA example discussed provides an insight into the generic capabilities of a cloud-based architecture. These are applicable and transferable to many disciplines and are worthy of discussion. For example the generic worker pattern shown in Figure 2 is a pattern commonly applied to a cloud-based architecture.

Cloud based applications may be scaled by either or both of two methods: scaling up by procuring a more powerful computational resource, and scaling out by procuring more instances of a computational resource, each of which offer some distinct advantages.

Scaling up is the most common method to improve performance, but is restricted by the capabilities of the most powerful hardware; the evolution of hardware performance should also be considered. Migrating existing solutions to more powerful hardware is a well understood problem and is particularly applicable where the task cannot easily be decomposed into smaller units of work. In the SSA example, each worker performs a unit of work that would be difficult to decompose, and satisfactory performance is within the capability of existing hardware.

In order to benefit by scaling out, an understanding of the computation is required as the algorithm has to be decomposed to take advantage of parallel operations. Scale-out often requires more development effort than migrating to a scale-up method. Dividing a task across multiple computational resources incurs an overhead, thus limiting the theoretical improvement in performance as more resources are added. In the SSA example, complex units of work consume tasks from a queue, which makes scaling-out easier since the number of workers consuming tasks from a queue can be varied with the length of the queue.

Using Microsoft Windows Azure was advantageous in this example as this is a PaaS, negating the need to maintain, patch and update the underlying OS. The environment also supports queues, various types of storage, including an SQL server and even includes a data market place to monetise datasets. One key advance which emerged during this work is a cloud-based high performance compute (HPC) cluster. Although not incorporated into this example architecture, HPC is a very powerful asset that ensures legacy MPI applications can seamlessly migrate into a cloud-based architecture.

Throughout this example it became obvious that cloud providers are offering powerful, cost effective infrastructures, but harnessing the power and migrating existing applications is often painful and just out of reach of most application scientists. Cloud providers are still evolving their offerings, and as migration scenarios and remote debugging capabilities improve we can expect to see scientists consuming more cloud resources.

Using workers as an individual unit of computation and feeding them individual tasks using a cloud based queue works well as it is easy to scale-out. Each item in the queue incurs a monetary cost as well as a retrieval time, thus when the computation for each task is short and the queue large, it is preferable to retrieve multiple tasks from the queue in one go, or better still, for each message in the queue to contain multiple tasks.

In this work we have demonstrated a cloud capability, but further work is required to optimise the workflow. For example, the number of worker instances is set at the start, and the workers do not terminate if there is a shortage of work. Likewise, as the queue for a particular worker increases in size, the number of instances does not automatically increase. It is possible to increase worker instances manually, but some work is required for taking them off-line.

Much of this work was carried out using the development environment for Windows Azure, which includes an emulator that can be run on a single development machine. This is a very powerful tool as we were able to test each worker and the entire workflow using a sample TLE dataset. Once we were satisfied with the results, simply deploying the workers on Azure resulted in a working system which could process complete TLE catalogues.

Further work is required to see how scaling-up can benefit the workflow; Microsoft Windows Azure workers come in different sizes and are billed proportionally. Buying larger, more powerful instances does not always improve the performance at the same rate as the instance cost. This is partly dependent upon the type of task – for example, whether it is computationally or IO intensive. It is no longer sufficient to look at overall performance, but rather performance per monetary cost.

## Discussion

The space surveillance and tracking segment of ESA's space situational awareness (SSA) system will provide vital security for space assets due to the increased awareness of the risks posed by space debris. The requirements of the SSA system will grow as the population of space objects — and the threat they pose — increases into the future. In this work, we have shown the relevance of a cloud-based architecture to SSA. In particular, the cloud-based architecture is able to manage unpredictable computational demands, in response to a break-up event, in addition to the predictable requirements associated with the regular processing of a space object catalogue. The solution can grow to include more physical computational and storage resources, thereby scaling with the demands of a catalogue of space objects which is rapidly increasing in size due both to conjunctions which introduce new debris, and the introduction of new measurement hardware which can provide information on increasingly smaller debris.

The cloud-based solution provides additional advantages, including the ability to share data with trusted partners simply, rapidly and securely. The partners, at their option, could then fund additional compute resources located close to the data to perform further analysis. The data marketplace provided by Windows Azure is also potentially advantageous, in that it extends the concept of readily and securely sharing data to include the option for the data owner to monetise the data set, the income from which could fund additional analysis, for example.

Further, we have illustrated the applicability of the cloud-based architecture to the development of algorithms that support the long-term sustainable use of outer space. The modular architecture pattern that a cloud-based solution promotes is ideal for this purpose, since a new algorithm could be implemented as a new worker type, and could be run in parallel with existing algorithms on the same data. The compute resources required to try out a novel algorithm and compare its results to those from an established code could be rented just for the time that they are required, making this an economical way to proceed.

In conclusion, not only have we shown how a cloud-based architecture using Microsoft Windows Azure can be successfully applied to an active debris removal mission design task, we have also developed a modular architecture which will be used in the future to support other SSA activities. The modular, cloud-based nature of this solution gives it some unique advantages over alternative architectures due to the rapid availability of huge computational and data storage resources; due to the simplicity that it brings to securely sharing raw or processed data; and due to the ease with which it facilitates the side-by-side comparison of alternative algorithms.

**Competing interests**
The authors declare that they have no competing interests.

**Authors' contributions**
SJJ, NSOB, AW and EEH have all contributed to the Clouds in Space project, taking part in migrating existing algorithms into Azure workers, designing the architecture and debugging the solution. HGL was responsible for the underlying SSA algorithms, contributing towards the paper, and proofreading. SJJ and NSOB were responsible for drafting the paper. SJC was the principal investigator on the project; he contributed to the cloud architecting, the paper and proofreading. All authors read and approved the final manuscript.

**Acknowledgements**
We are grateful to EPSRC and JISC for funding this work through EP/I034009/1. We thank Microsoft for providing access and resource on Windows Azure.

Received: 18 October 2012 Accepted: 8 January 2013
Published: 22 January 2013

**References**

1. NASA (2009) Orbital Debris Frequently Asked Questions. NASA Orbital Debris Program Office http://orbitaldebris.jsc.nasa.gov/faqs.html
2. Lewis HG, Swinerd G, Newland R (2011) The space debris environment: future evolution. Aeronautical J 115: 241–247
3. Liou JC, Johnson NL (2006) PLANETARY SCIENCE: Risks in Space from Orbiting Debris. Science 311(5759): 340–341
4. Johnston SJ, Cox SJ, Takeda K (2011) Grid and Cloud Database Management. In: Aloisio G Fiore S (eds). Springer, chap. 9: Scientific computation and data management using Microsoft Azure, pp 169–192. http://www.springer.com/computer/database+management+%26+information+retrieval/book/978-3-642-20044-1
5. Hey T, Tansley S, Tolle K (eds) (2009) The Fourth Paradigm: Data-Intensive Scientific Discovery. In: Jim Gray's Fourth Paradigm and the Construction of the Scientific Record. Microsoft Research, 1 edition October, 177
6. Peenikal S (2009) Mashups and the enterprise. Strategic white paper [Mphasis white paper]. http://www.mphasis.com/pdfs/Mashups_and_the_Enterprise.pdf
7. Hinchcliffe D (2008) An Executive Guide to Mashups in the Enterprise. Executive white paper [Jackbe.com Accessed 23 Aug 2010]. http://mdc.jackbe.com/downloads/Jackbe_Mashups_in_the_Enterprise_White_Paper.pdf
8. Lin G, Fu D, Zhu J, Dasmalchi G (2009) Cloud Computing: IT as a Service. IT Professional 11(2): 10–13
9. O'Brien NS, Johnston SJ, Hart EE, KDjidjeli, Cox SJ (2011) Exploiting cloud computing for algorithm development. CyberC 1: 336–342. doi:10.1109/CyberC.2011.60
10. Johnson NL, Stansbery E, Liou J, Horstman M, Stokely C, Whitlock D (2008) The characteristics and consequences of the break-up of the Fengyun-1C spacecraft. Acta Astronautica 63(1-4): 128–135
11. Yi-yong L, Huai-rong S, Zhi L (2009) Faster algorithm of debris cloud orbital character from spacecraft collision breakup. Adv Space Res 43(10): 1527–1531
12. Lewis H, Swinerd G, Newland R, Saunders A (2009) The fast debris evolution model. Adv Space Res 44(5): 568–578
13. Alarcón-Rodríguez J, Martínez-Fadrique F, Klinkrad H (2004) Development of a collision risk assessment tool. Adv Space Res 34(5): 1120–1124
14. Liou JC (2011) An active debris removal parametric study for LEO environment remediation. Adv Space Res 47(11): 1865–1876
15. Liou JC, Hall DT, Krisko PH, Opiela JN (2004) LEGEND - a three-dimensional LEO-to-GEO debris evolutionary model. Adv Space Res 34(5): 981–986. [Space Debris]
16. Hohmann W (1960) Die Erreichbarkeit der Himmelskörper. Munich and Berlin: R. Oldenbourg 1925. [NASA Technical Translation F-44: The attainability of heavenly bodies, http://archive.org/details/nasa_techdoc_19980230631]
17. Fatland DR, JFay, Fay D (2011) Geoscience Visualization with World Wide Telescope. Microsoft Research Technical Report

doi:10.1186/2192-113X-2-2
**Cite this article as:** Johnston *et al.*: **Clouds in Space: Scientific Computing using Windows Azure.** *Journal of Cloud Computing: Advances, Systems and Applications* 2013 **2**:2.