Leveraging Windows Workflow Foundation for Scientific Workflows in Wind Tunnel Applications

A.Paventhan, Kenji Takeda and Simon J. Cox
Microsoft Institute for High Performance Computing
School of Engineering Sciences
University of Southampton
Highfield, Southampton, SO17 1BJ, UK
Email: {povs, ktakeda, sjc}@soton.ac.uk

Denis A. Nicole
School of Electronics and Computer Science,
University of Southampton,
Highfield, Southampton, SO17 1BJ, UK
Email: dan@ecs.soton.ac.uk

Abstract

Scientific and engineering experiments often produce large volumes of data that must be processed and visualised in near-realtime. An example of this, described in this paper, is microphone array processing of data from wind tunnels for aeroacoustic measurements. The overall turnaround time from data acquisition and movement, to data processing and visualization is often inhibited by factors such as manual data movement, system interoperability issues, manual resource discovery for job scheduling, and disparate physical locality between the experiment and scientist or engineer post-event. Workflow frameworks and runtimes can enable rapid composition and execution of complex scientific workflows. In this paper we explore two approaches based on Windows Workflow Foundation, a component of Microsoft WinFX.

In our first approach, we present a framework for users to compose sequential workflows and access Globus grid services seamlessly using a .NET-based Commodity Grid Toolkit (MyCoG.NET). We demonstrate how application specific activity sets can be developed and extended by users. In our second approach we highlight how it can be advantageous to keep databases as central to the complete workflow enactment. These two approaches are demonstrated in the context of a wind tunnel Grid system being developed to help experimental aerodynamicists orchestrate such workflows.

1. Introduction

Scientists and engineers conducting experiments often perform a sequence of tasks in a workflow pattern similar to that of a business process. The individual steps in business workflow systems are typically control flow driven, whereas scientific workflows may also be data and event driven. In a simplistic scenario, the steps in an experimental workflow might include data acquisition, data movement, pre-processing, processing and visualization. The data acquisition instruments, storage systems and compute resources are often distributed within and across organizational boundaries. The user may have to deal manually with the complexity of the resource discovery, data movement and job scheduling, impacting the overall turnaround time and reducing time to insight. Customized application specific workflows can help reduce the time taken for a complete workflow by automating data flow driven activities, supplementing or replacing manual user-driven steps.

Current Grid computing [16] solutions allow resource sharing across organizational boundaries. But, there are many issues still to address while implementing application-specific scientific workflow on grids. In order to provide users with a workflow composition framework, the workflow development and execution environment apart from composition, monitoring and scheduling, should support customization of activities. As can be seen from our discussion in the next section, existing workflow solutions are either developed keeping the target application domain in mind [11, 12] or their functional extension to suit a particular application domain requires more work [10, 18, 20].

The Windows Workflow Foundation (WWF) part of Microsoft WinFX [5] is an extensible framework for developing workflow solutions. It has predefined sets of activities (If-Else, While, Parallel, InvokeWebService and so on) and allows for user-defined custom activities by object inheritance from base classes. Since it is integrated with robust development environment (Microsoft .NET Framework [15]) supporting multiple languages, user can compose, run, debug, version, share workflows apart from developing their application code under the same environment.

In this paper we present two approaches to building customized scientific workflows based on WinFX Windows Workflow Foundation. The application example used is a wind tunnel grid framework that includes experiment specific activities enabling users to compose customized grid workflow quickly. A typical wind tunnel experimental workflow would involve a sequence activities as shown in Figure.1.

In our first approach, we extend the base WinFX workflow activities to deliver a set of application-specific wind tunnel grid workflow activities. By making use of our earlier work, the multi-language Commodity Grid Kit (MyCoG.NET) [9], we can seamlessly access Globus grid services as required, as well as other Web Services.

In our second approach, we present a database-centric architecture for wind tunnel experimental workflow that hosts both data and processing. SQL Server 2005 stored procedures execute processing code in any supported CLR languages, such as C# or FORTRAN.

Until now, the role of database management systems (DBMS) in the scientific domain has largely been relegated to that of a passive store for querying metadata and results. With DBMS capabilities evolving rapidly, such as the recent advances including high-level language stored procedures [6, 14, 33], native support for XML [19, 24, 27], XML Web Services [4, 17, 21] and transactional messaging [2, 31], it is pertinent to re-examine the role of DBMS in scientific workflow.

Many scientific processing codes are amenable to parallel processing at a coarse- or fine-grained level. Some of the wind tunnel experimental processing algorithms of interest can be run in a task-farmed manner, with each process requiring relatively short runtime (measured in seconds or minutes). Typical cases involve an overall large volume of data to be processed. In such cases it is better to move the processing to data, rather than vice-versa. Hence, in our second approach the processing and data are together hosted on a DBMS cluster.

The rest of the paper is organized as follows. Section 2 compares related works. In Section 3, we outline wind tunnel experiment requirements and show why workflow customization is important. Section 4 covers a brief overview of Windows Workflow Foundation. In Section 5 and 6, we present our approaches to wind tunnel experimental workflow. In Section 6, conclusions and future work are presented.

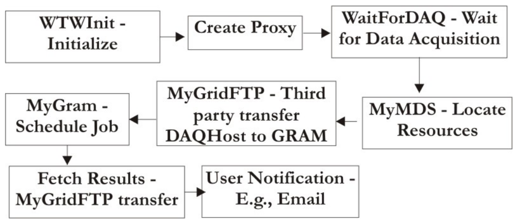

WTWInit - Initialize
Create Proxy
WaitForDAQ - Wait for Data Acquisition
MyMDS - Locate Resources
MyGridFTP - Third party transfer DAQHost to GRAM
MyGram - Schedule Job
Fetch Results - MyGridFTP transfer
User Notification - E.g., Email

Figure 1. A Simple Wind Tunnel Experimental Workflow

2. Related Work

The set of requirements on scientific workflow systems from different scientific disciplines, in terms of data size, formats, real-time requirements and computational complexities, bring different sets of challenges. In this section, we discuss the influences of related works on our project and highlight the particular wind tunnel experimental requirements that lead us to our work.

The GriPhyN [12] project addresses the workflow requirements of physics experiments. When the user requests a data object, an abstract workflow DAG that would generate the desired data object is constructed. The abstract workflow is then converted to concrete workflow represented as Condor DAGman files [1] and submitted to the Condor-G scheduler. In case of wind tunnel experiments, the workflow is triggered by data acquisition and the raw data transfer requires more customization.

In the KEPLER system [20], the workflow components are known as *actors* and their communications happen through interfaces called ports. The component interactions and their order of execution are controlled by an object known as a *director*. There are workflow component extensions supporting Web service invocation and Grid service access. KEPLER addresses Grid and web services access, but, the integration of data acquisition hardware and experiment specific data transfer are paramount in our work.

Grid-DB [18] is a data-centric grid workflow system. It provides a declarative language for the user to register code and data with the system. Further, set of programs can be modeled into an abstract workflow. Grid-DB submits the programs to Condor pool for execution. In our work, we aim at providing user with intuitive interface for workflow design and ability to extend the functionality of the activi-

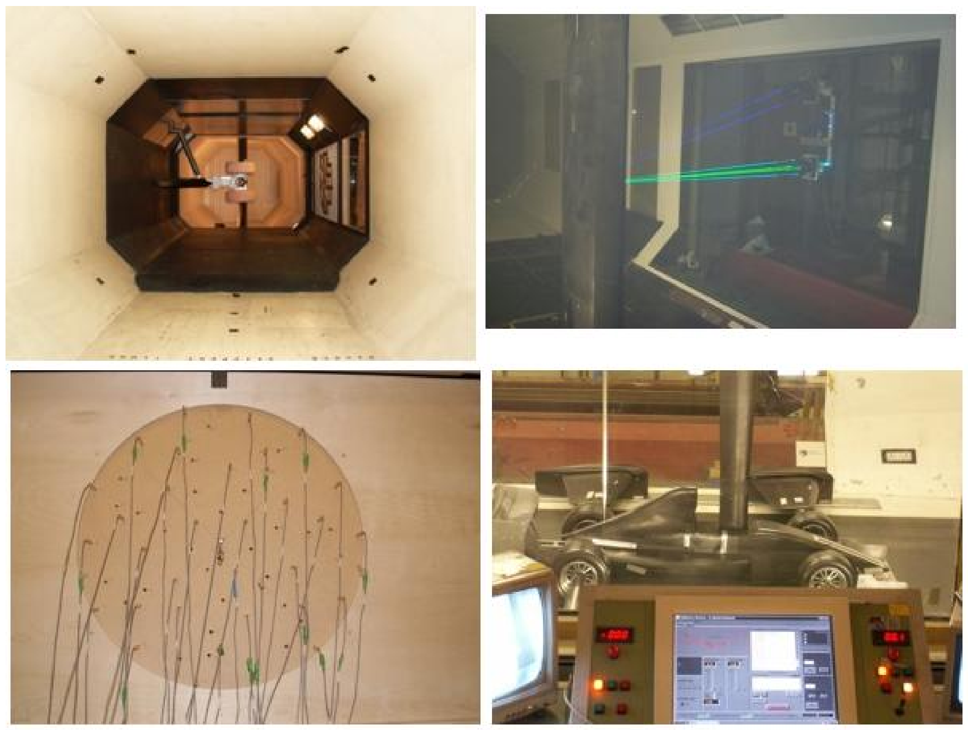


Figure 2. Wind Tunnel Experiments (Clockwise from top left: Aircraft landing gear test, LDA flow measurements, Racing car model test, Wall mounted microphone array)

ties by object inheritance.

DiscoveryNet [11] addresses the need for knowledge discovery process in lifesciences. The components and workflows in DiscoveryNet are composed as Web and Grid services by sharing across teams.

The wind tunnel experiments run on some proprietary systems and their integration into scientific workflow requires customized solution. Also, the data transfer and processing requirements are experiment specific. The wind tunnel workflow framework is aimed at providing users with ready-to-use experiment specific workflow activities, hiding the underlying complexities and at the same time providing advanced users with an option for customization, if required. Over time the processes and systems available in the wind tunnel change significantly, so the ability to rapidly develop and customise workflows is crucial.

3. Wind Tunnel Experiment Requirements

Wind tunnels are widely used to design, test and verify aerodynamics of aircraft, cars, yachts, and buildings, amongst others. A variety of testing techniques are used in a given wind tunnel complex, depending on the particular application and flow regime. Measurement of aerodynamically generated noise is also an important application of wind tunnel testing and has its own special set of requirements that are discussed below.

The wind tunnel facilities at University of Southampton [3] house a variety of specialized experimental hardware and software for academic and industrial research. They are used in a wide variety of projects including fundamental aerodynamics, aerodynamics of racing cars and road vehicles, rotorcraft aerodynamics, aeroacoustics, aeronautics, wind engineering and industrial aerodynamics. Some of the experiments include Laser Doppler Anemometry (LDA), Phased Microphone Array Systems and Particle Image Velocimetry (PIV) (see Figure.2). In all these experiments, the data acquisition event is generally followed by application-specific and user-defined processing steps. In many experiments, the data movement operations to the processing computer are manual due to interoperability issues between hardware, software and the acquisition systems. The automated solution would require the following steps: 1. Experiment-specific data verification to ensure whether the acquisition was indeed successful 2. Experiment-specific annotation of metadata for auto-upload and processing 3. Raw data movement operations (based on metadata) and 4. User-defined processing steps.

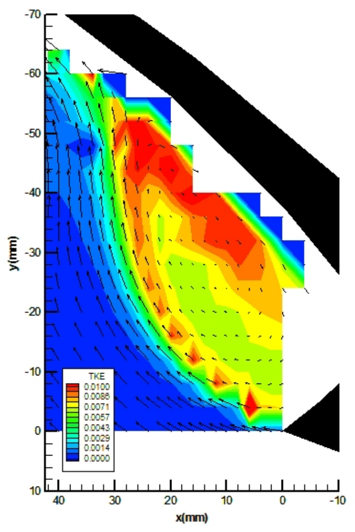


Figure 3. Typical LDA results showing turbulent kinetic energy contours around the leading edge slat of a multi-element wing wind tunnel test [30]

The data generated during acquisition vary in terms of the number of data items, file size and format, depending on the wind tunnel experiment and user parameters. The processing requirements are also user and experiment specific. The broad requirement is that the experimental workflow should be customizable during acquisition, data movement and processing. In this section, we list some of the requirements with reference to three wind tunnel experiments.

3.1. Laser Doppler Anemometry (LDA)

LDA systems use non-intrusive point-measurement techniques to accurately measure fluid velocity in highly turbulent or reversing flow. The measurement results are important steps in fine-tuning product designs to improve aerodynamic efficiency, quality and safety. The accuracy of this technique is invaluable for measuring turbulence levels to understand flow physics at a detailed level.

For each experimental configuration, a calibration step must be performed and a transformation matrix must be derived. The transformation matrix is used to translate the raw data in laser coordinate system to the tunnel coordinate system during processing. LDA data acquisition software collects selected number of samples (typically thousands) at user programmed traverse positions of up to three velocity components (This value is also equal to number of Burst Spectrum Analysers (BSA)). The collected data are stored in separate raw data files for each traverse position. The raw data filenames have a suffix 0, 1 or 2 to indicate the velocity component (u,v & w, in the laser coordinate system). The file extension represents the traverse position. There are essentially $n \times p$ raw data files for one experiment, *n (n=3)* is the number of velocity component and *p* is the number of traverse positions. The user parameters for acquisition are stored in a separate flat file.

The upload activity of the workflow requires verification of the raw data files prior to the uploading to processing node. Since the number of velocity components and traverse positions are known from the parameter file, all the raw data files along with metadata (transformation matrix, user parameters) can be uploaded without any user intervention.

The LDA processing steps are: 1. Data conversion - encoded samples are converted to physical units. 2. Coincidence processing - transformed velocity components are computed from measured velocity components using transformation matrix. 3. Moment processing 4. Spectrum processing 5. Correlation processing

3.2. Particle Image Velocimetry (PIV)

Particle Image Velocimetry is a non-intrusive, field-based technique to measure fluid velocity. In contrast to the LDA system, which measures velocities at a single point in space at multiple times, the PIV system simultaneously measures velocities in the field of view of the sensor (a digital camera) at a single instant in time. 2-D PIV systems use a single camera, while 3-D PIV systems employ two CCD cameras, one on the left and one on the right, to produce two 2-dimensional vector maps showing the instantaneous flow field as seen from each of the cameras. Using the calibration function obtained during camera setup, the true 3-D particle displacement can be calculated.

The requirements for PIV are: 1. An upload component to transfer image frames from acquisition system to processing cluster. 2. Timely processing response by means of high-performance implementation. This requires parallel implementation of cross-correlation computation between image frames.

3.3. Microphone Arrays

The microphone array technique is used to measure noise of aircraft components (slats, landing-gears, flaps, etc) to help aerospace engineers improve the aircraft design and to reduce the overall airframe noise. Microphone arrays consist of multiple, O(100), microphones that must be simultaneously sampled. The phase shift between channels is then used to derive acoustic source information.

The aeroacoustic researchers at University of Southampton use a National Instrument's NI4472-based data acquisition system designed for acoustic and vibration applications. This is able to sample multiple channels at up to 96kHz (48kHz anti-aliased), while remaining tightly synchrosnised in time. The system controller is driven by an NI's proprietary Labview system running Windows XP. The I/O slots of the controller can be populated with specialized data acquisition cards each supporting a number of channels. For example, a system with 7 cards and 8 channels would support an array of 56 microphones for the measurement.

A typical data acquisition event on a high channel count system would generate a large volume of data, running into hundreds of megabytes per second [22]. In order to achieve realtime processing of the time-series data, efficient data storage and data transfer techniques must be employed. The raw data comprises blocks of samples received from individual microphones at a user-specified sampling rate.

The microphone phased array processing involves a series of steps known as beamforming to compute cross-spectral matrix of the size $M \times M$, where M is the number of microphones. The steps include: 1. Data calibration using Microphone sensitivity data 2. Using a Hamming windowing function, blocks of data are transformed into frequency domain by Fast Fourier Transform (FFT) 3. Block averaging cross spectral components 4. Background noise removal.

The metadata (number of microphones, microphone sensitivity data, sampling rate, block size etc) must be used for customized data upload. The user processing step also requires customization so that different algorithms can be developed and used as the state-of-the-art advances.

As can be seen from the three experiments discussed above, experiment specific upload activity with the ability for user customization is required. Similarly, default experiment specific processing steps which can be modified to suit user requirements are essential.

4. Windows Workflow Foundation

Microsoft Windows Workflow Foundation is an extensible framework and is part of the upcoming Microsoft's next generation development Framework, WinFX [5]. The workflow in Windows Workflow Foundation is composed from a set of *activities*, compiled to a .NET assembly. It can be executed under the Common Language Runtime (CLR) in a variety of container processes.

4.1. Workflow Model and Composition

There are two models supported [7]: 1. Sequential workflow model - comprising activities that execute in a predictable sequential path, and 2. State machine model - a flow driven by events triggering state transitions. In both these models the basic element of the workflow is called an *activity*. Some of the Windows Workflow Foundation's activity types include: control-flow (While, IfElse, Delay), exception (throw, exception-handler and BPEL compensations), data handling (Update, Select), transactions (and compensations for long-lived "transactions" that cannot be directly unwound) and Communication (InvokeWebService, InvokeMethod).

A workflow consists of metadata for the workflow definition and the accompanying .NET classes that form the code file. The workflow can be composed using a visual workflow designer; this has a drag-and-drop interface and can be hosted in Visual Studio or a user-application. Alternatively, users can declaratively write in XOML, an XML dialect for writing workflows. The workflow can also be completely coded in CLR languages. A workflow must be compiled with *wfc* workflow compiler before it can be run.

All the Windows Workflow Foundation *activities* are derived from *System.Workflow.ComponentModel.Acitivity* base class. The Windows Workflow Foundation extensible development model enables creation of domain-specific *activities* which can then be used to compose workflows that are useful and understandable by domain scientists.

4.2. Workflow Runtime, Scheduling and Hosting

The workflow runtime layer is at the core of Windows Workflow Foundation and is responsible for execution, tracking, state management, scheduling and policies. The workflow engine runs inside a *hosting process* provided by the workflow application. The hosting layer is responsible for communication, persistence, tracking, transaction, timing and threading. It is possible to dynamically update the running workflows on the fly.

With this flexible approach to workflow hosting and extensible framework for workflow activities, most of the functionality of the state-of-the-art scientific workflow systems [34] can be hosted on top of Windows Workflow Foundation. In the following sections we illustrate how specific workflows to wind tunnel aerodynamic and aeroacoustic testing can be constructed using Windows Workflow Foundation.

5. Sequential workflows exploiting Globus Grid Services

Our approach to implementing a wind tunnel grid workflow based on Windows Workflow Foundation is shown in Figure.4.

The user has access to four different sets of activities from which to compose an experimental work-

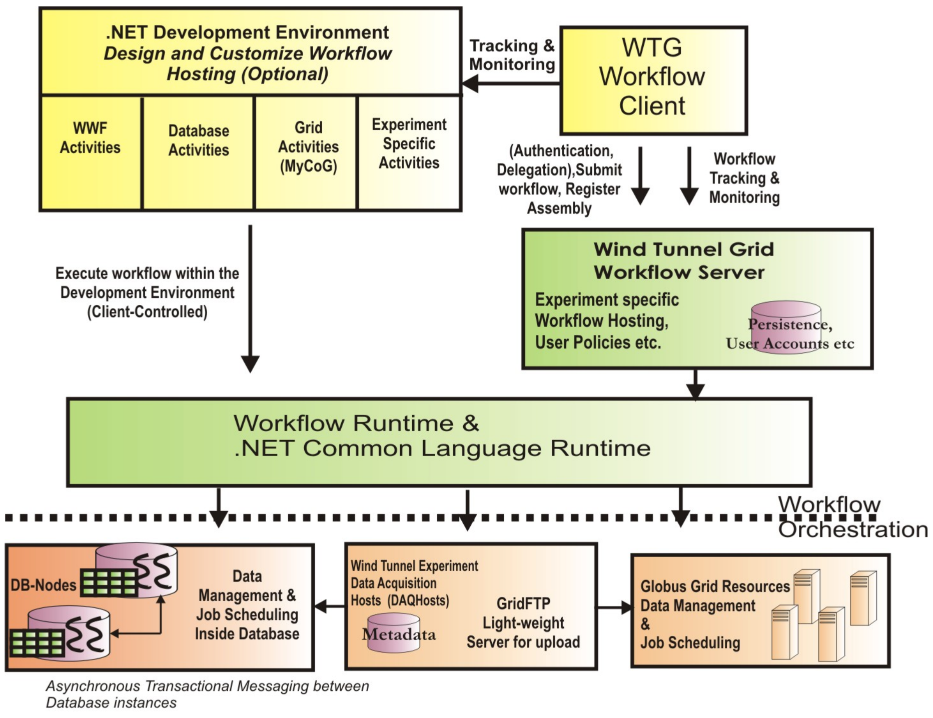

.NET Development Environment
Design and Customize Workflow Hosting (Optional)
WWF Activities
Database Activities
Grid Activities (MyCoG)
Experiment Specific Activities
Tracking & Monitoring
WTG Workflow Client
(Authentication, Delegation),Submit workflow, Register Assembly
Workflow Tracking & Monitoring
Execute workflow within the Development Environment (Client-Controlled)
Wind Tunnel Grid Workflow Server
Experiment specific Workflow Hosting, User Policies etc.
Persistence, User Accounts etc
Workflow Runtime & .NET Common Language Runtime
Workflow Orchestration
DB-Nodes
Data Management & Job Scheduling Inside Database
Wind Tunnel Experiment Data Acquisition Hosts (DAQHosts)
Metadata
GridFTP Light-weight Server for upload
Globus Grid Resources Data Management & Job Scheduling
Asynchronous Transactional Messaging between Database instances

Figure 4. Wind Tunnel Experimental Workflow Architecture

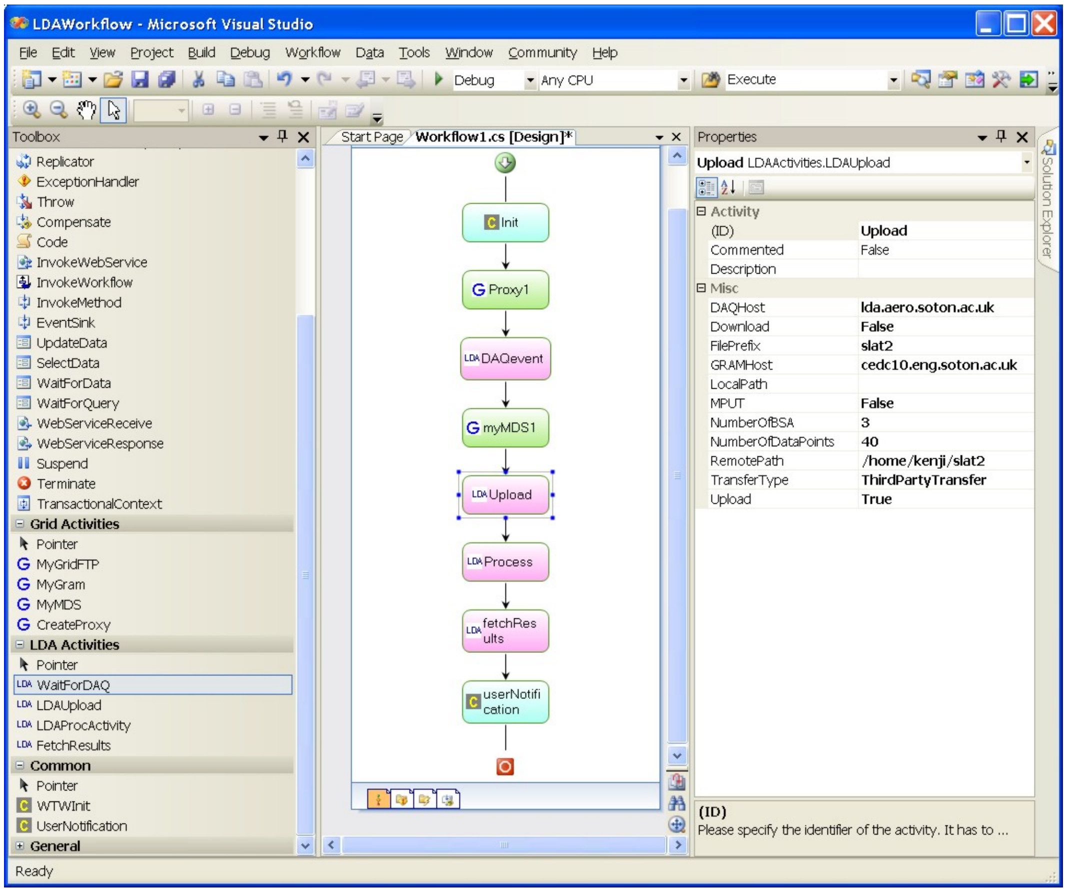
flow: 1. Windows Workflow Foundation Activities 2. MyCoG.NET-based Grid activities to access Globus services 3. Experiment-specific activities for upload, processing, results etc. and 4. Database activities (which will be discussed in the next section). The user can design the workflow from these activity sets depending on his or her requirements.

There are two possible options to host the workflow: 1. Client-controlled hosting, and 2. By submitting to the wind tunnel grid workflow server for hosting. In client controlled hosting, the workflow runtime runs as part of the *host process* running on the user's PC. In this case, the user must leave the *host process* running until the workflow finishes. Underlying the hosting process is the WinFX workflow runtime and .NET Common Language Runtime. The workflow can be monitored from wind tunnel grid workflow client while it is running.

In the second case, the user deploys their workflow for hosting to the wind tunnel grid workflow server after successful Grid Security Infrastructure (GSI) [32] authentication and delegation of user credentials. The wind tunnel grid workflow server maintains user account information. A separate *host process* is instantiated for the user's workflow and the runtime is started. This allows the user to disconnect after submission and monitor the workflow periodically from a wind tunnel grid workflow client.

The generic wind tunnel workflow activities are: *WTWInit* - initializes wind tunnel workflow server hosting process for the user; this is the first activity in any wind tunnel experimental workflow. *UserNotification* - Customized user notification on the state of the workflow (workflow completion or failure).

Figure.5 shows a sequential LDA workflow designed using customized wind tunnel grid workflow activities. The WaitForDAQ is an event driven activity customized for the LDA experiment. On completion of the data acquisition, this activity verifies the raw data files for completeness and workflow transitions to next activity. The MyGridFTP, MyGram and MyMDS activity uses MyCoG.NET Commodity Toolkit to access Globus resources. The implementation details of MyCoG.NET are discussed in [9]. These Grid service access activities are further customized for individual experiments. For example, as shown in Figure.5, the *LDAUpload* activity derived from MyGridFTP has specific input properties for the experiment (data acquisition hostname, number of data points, number of burst spectrum analysers). Some properties are initialized at workflow design time with default values and others received as input from the *host process*. The experiment specific properties would enable, for example, automatic uploading of raw data files from data acquisition host to a Gram server, as can be seen from Section.3.1. The *FetchResults* is an activity derived from MyGridFTP to transfer results from Gram host to Windows Workflow server and to the user's desktop.



Upload LDAActivities.LDAUpload

Activity	
(ID)	Upload
Commented	False
Description	
Misc	
DAQHost	lda.aero.soton.ac.uk
Download	False
FilePrefix	slat2
GRAMHost	cedc10.eng.soton.ac.uk
LocalPath	
MPUT	False
NumberOfBSA	3
NumberOfDataPoints	40
RemotePath	/home/kenji/slat2
TransferType	ThirdPartyTransfer
Upload	True

(ID)
Please specify the identifier of the activity. It has to ...

Figure 5. LDA workflow

In our earlier implementation [9] of LDA workflow, each stage of the workflow is triggered by the user via portal interface. By bringing Globus grid service access to the Windows Workflow Foundation using MyCoG, the user would be able to design, execute and monitor wind tunnel experimental workflow on Globus resources.

OGSA-DAI [8] web services allow data from different sources (relational, XML and files) to be queried, updated, transformed and delivered. As the processed result sets and metadata of wind tunnel experiments are stored in relational format, OGSA-DAI based Grid database service interfaces can be developed conforming to Global Grid Forum (GGF) standards, so they could be included as custom WWF activities.

6. Database workflows

In this section, as part of our ongoing work, we present a database-centric approach to wind tunnel experimental workflow. The strategy here is to run the data parallel code on a database cluster (bottom left of Figure.4) that hosts both experimental data and user algorithms. The customized *database activity* set will allow the user to compose workflow based on this approach.

6.1. Experimental Data Management

Most of the scientific applications use flat files for storing configuration files, raw data and processed results. In wind tunnel experimental setups, there are multiple experiments each having different sets of configuration parameters for each run, with many users producing experimental data on a daily basis. The complexity of managing the data sets poses the single largest problem for the user. The flat file approach leads to version conflict and data management is mostly adhoc. Large-scale data management is thus better served by maintaining experimental data inside database systems where possible. Different scientific applications in fields including High Energy Physics [29], Earth Sciences [23] and Geosciences [26] have already been demonstrated using the database-centric approach in Grid environments.

The advantages of database management systems in real-world scientific application have been demonstrated in the Sloan Digital Sky Survey project. With efficient indexing, join and parallel query operations, a twenty times speedup was achieved as compared to a file-based implementation [25]. Complete workflow modelling using declarative style languages to integrate workflow inside databases has also been studied recently [28].

A typical Microphone Array experiment produces hundreds of megabytes of data per second. Considering the storage constraints on the acquisition system, the experimental data needs to be uploaded in near-realtime. The user is interested in an early processed results so as to know how the experiment is progressing and whether any course corrections are required. Further, the significant cost associated with the resources (hardware, software, manpower, etc) at large wind tunnel facilities and the user's limited time allotment to run the experiments make the *near-realtime* processing step all the more essential. The application-specific timely response can be achieved by employing distributed and parallel processing components.

Off-the-shelf cheap commodity processors, storage and high speed local area interconnect have made parallel processing approaches feasible in the database domain [13]. Data partitioning enhances query performance; but the scientific processing requires parallelism in computations. Many popular database systems now support high-level language stored procedures and user-defined functions. This allow us to host computation inside databases. SQL Server 2005 hosts .NET Common Language Runtime (CLR) [6]; the Java Virtual Machine (JVM) and .NET CLR are supported by Oracle [33] and by IBM DB2 [14]. In addition, the asynchronous queue based messaging framework (SQL Service Broker [31] or Oracle streams [2]) enables transactional message exchanges between data parallel tasks running inside database instances.

6.2. Microsoft SQL Server

The implementation approach we discuss in the next section is based on SQL Service Broker [31] and CLR Integration [6] in SQL Server 2005. We believe it can be implemented over most commercial or open source database systems.

Service Broker provides asynchronous, reliable and transactional messaging support. Service Broker objects include Queues, Dialogs, Message Types, Contracts and Services. These objects can be created using regular CREATE, ALTER and DROP Data Definition Language (DDL) commands. The messages from the transmit queue can be transferred to the receive queue inside a transaction; they are thus reliable. The receive queue could be in the same or a different instance of the database or even on a remote machine. The messages can also be routed through an intermediary machine.

The .NET Common Language Runtime (CLR) is integrated into the recent release of Microsoft SQL Server 2005. It enables users to write stored procedures, triggers, functions in any of the CLR languages. These high-level languages are better in terms of performance and expression of arithmetic computation, string manipulation, logic etc. than Structured Query Language (SQL). SQL is, however, better when it comes to set oriented queries (SELECT/UPDATE/INSERT/DELETE etc). An application can take advantage of combining high-level language code for procedural logic and SQL code for queries.

6.3. Architecture

The Database centric workflow architecture is also shown in Figure.4 with a major difference being scheduling of job onto the database cluster. The wind tunnel workflow server, DAQHost and DB-Nodes all run SQL Server 2005 instances and they communicate via SQL Service Broker.

The major components in this architecture are:

1. User assembly - An application-specific class library implementing processing algorithms and residing in DB-Nodes. Users can implement new classes by inheritance and by overriding the processing logic. The user can compile a customized assembly and register it through database workflow activity.

2. Database Workflow Activities:

   - RegisterAssembly, ReinstallAssembly, RemoveAssembly: These activities have explicit properties like ApplicationType (LDA, PIV, Microphone), version etc. and implicit properties like "user who is invoking", as the entire workflow runs with delegated user credentials stored on the wind tunnel workflow server. The assembly activities will internally translate into SQL DDL statements (CREATE ASSEMBLY, ALTER ASSEMBLY, DROP ASSEMBLY). Once the assembly is registered, the public interfaces (public class, static functions, data) are available to WTG-Database stored procedures.
   - DBUpload: This is a Service Broker activity running in DAQHost to upload raw data and configuration parameters to DB-Nodes. User can derive for customized upload.
   - DBProcess: This activity can start subsequent to successful DBUpload. User can specify three configurations: *a) Exact* - Data distributed by sending out user specified number of messages to DB-Nodes, *b) Flexible* - As many messages depending on the available CPUs and data size

and *c) Single* - No data distribution. The first two configurations require the part results from distributed nodes to be merged.

3. DB-Nodes: The DB-Nodes run the SQL Server 2005 instance. They also run *host processes* to handle database workflow activity. During registration of a user assembly, a unique namespace is associated (based on username, application and version), so that compute message requests can be dispatched to the correct CLR function.

4. WTG Database: The main database containing tables to store user information, wind tunnel application information, results, user assembly, messages, queues. The following statements illustrate the usage of SQL Server DDL extensions to support Service Broker applications and CLR integration.

   ```
   CREATE QUEUE ServiceQueue;

   CREATE MESSAGE TYPE CSMCompute VALIDATION = WELL_FORMED_XML;

   CREATE ASSEMBLY CSMAssembly FROM 'path/to/assembly' WITH PERMISSION_SET = EXTERNAL_ACCESS;

   CREATE PROC CSMProc AS EXTERNAL NAME CSMAssembly.[WindTunnelGrid.Microphone.CSMService].CSMProc
   ```

   The queue is associated with a stored procedure and when a message like 'CSMCompute' arrives, Service Broker activates the stored procedure and despatches the message to the worker threads. The worker thread is responsible for invoking the correct version of the user code. Simple messages are defined in XML and messages which carry data and results are defined as opaque binary (de-serialized by the receiver).

5. WTG-Workflow Client: User can run this client from anywhere on the network to access wind tunnel workflow services. Users are able to login, register their application assembly, check the status of all their currently running workflows, query/download results of completed run for local visualization, etc.

6. Wind tunnel Workflow Server: Hosts the workflow and provides database storage for user accounts and results. This server also authenticates a user on their first entry into the system and accepts user's delegated X.509 credentials for job scheduling (using a lightweight GSI server side component for authentication and authorization).

As an initial test, we studied the performance of microphone array processing algorithm by parallelizing it and running within SQL Server 2005 configured on a Dual Pentium III 1 GHz Windows Server 2003 machine. The raw data for the test is from 56 microphones with 100 blocks each consisting of 1024 samples. It can be considered as a 3-dimensional matrix of size $100 \times 56 \times 1024$. The data is split among two threads each working on a equal sized samples ($50 \times 56 \times 1024$). The processing timings (Table 1) show linear speedups can be achieved by splitting the number of blocks among available CPUs. The two thread version takes almost only half the time of single thread version when working on half of the data in parallel. Also, the performance of stored procedure version (running under SQL Server 2005 runtime) is comparable to the console version (running outside database). The serialisation and de-serialisation timing refers to the execution time to convert an in-memory .NET object representing configuration parameters plus raw data to and from a byte stream suitable for network transfer. This overhead is minimal and is required only when inter-node transfer is involved.

Table 1. Beamforming Timings

Implementation type	Number of threads	Number of blocks	Serialisation Deserialisation	Beamforming
.NET Console Application	1	100	11.9 sec	95.80 sec
.NET Console Application	2	50	12.07 sec	47.69 sec
.NET CLR Stored Procedure	1	100	14.95 sec	98.52 sec
.NET CLR Stored Procedure	2	50	13.76 sec	50.79 sec

100 blocks consists of 102400 samples and 50 blocks consists of 51200 samples. Block size = 1024

7. Conclusions and Future Work

In this paper we have discussed the implementation of real-world scientific workflows using Windows Workflow Foundation in wind tunnel applications. We have presented two approaches for users to build customized application workflows.

By using a .NET Commodity Grid toolkit (MyCoG.NET) we have demonstrated interoperability between Windows Workflow Foundation and Globus grid services, including certificate-based authentication, proxy support, GridFTP file transfer and GRAM job submission.

For certain applications it is desirable to move the computation to the data. Where this data can be stored in a DBMS, we have shown that it is possible to run data parallel processing within SQL Server 2005, leveraging CLR integration and Service Broker.

Due to its extensibility, the addition of provenance tracking, semantic definition and representation, and derivation of metadata from results, are possible. While the example applications described here are based around experimental science, the applicability to computational simulation is also apparent.

As more grid resources move to Web Service interfaces, the use of Windows Workflow Foundation as a generic framework for composing scientific workflows is likely to become more common.

Acknowledgment

The authors would like to thank Microsoft for its ongoing support.

References

[1] Condor DAGman. http://www.cs.wisc.edu/condor/dagman/.

[2] Sharing Information with Oracle Streams. An Oracle White paper, May 2005.

[3] Southampton Wind Tunnels, University of Southampton. http://www.windtunnel.soton.ac.uk/.

[4] Using Native XML Web Services in SQL Server 2005. http://msdn2.microsoft.com.

[5] WinFX Dev. Center. http://msdn.microsoft.com/winfx/.

[6] A. Acheson, M. Bendixen, and et.al. Hosting the .NET Runtime in Microsoft SQL Server. In *ACM SIGMOD Conference*, pages 860–865, 2004.

[7] P. Andrew, J. Conard, S. Woodgate, J. Flanders, G. Hatoun, I. Hilerio, P. Indurkar, D. Pilarinos, and J. Willis. *Presenting Windows Workflow Foundation, Beta Edition*. Sams, September 2005.

[8] M. Antonioletti, M. P. Atkinson, R. Baxter, and et.al. The design and implementation of Grid database services in OGSA-DAI. *Concurrency and Computation - Practice & Experience*, 17(2-4):357–376, 2005.

[9] A.Paventhan and K. Takeda. MyCoG.NET:Towards a multi-language CoG Toolkit. In *3rd International Workshop on Middleware for Grid Computing (MGC 05), Co-located with ACM/IFIP/USENIX 6th International Middleware Conference*, November 2005.

[10] D. Breuer, D. Erwin, D. Mallmann, R. Menday, M. Romberg, V. Sander, B. Schuller, and P. Wieder. Scientific Computing with UNICORE. In *Proceedings of the NIC Symposium*, February 2004.

[11] V. Curcin, M. Ghanem, Y. Guo, A. Rowe, W. He, H. Pei, L. Qiang, and Y. Li. IT Service Infrastructure for Integrative Systems Biology. In *IEEE International Conference on Services Computing,(SCC'04)*, September 2004.

[12] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman. *"Workflow Management in GriPhyN" in Grid Resource Management J. Nabrzyski, J. Schopf, and J. Weglarz editors*. Kluwer, 2003.

[13] D. DeWitt and J. Gray. Parallel Database Systems: The Future of High Performance Database Processing. *Communications of the ACM*, 36(6):85–98, 1992.

[14] G. Evans. A detailed look at DB2 Stinger .NET CLR Routines, IBM developerWorks article. June 2004.

[15] D. Q. M. Fay. An Architecture for Distributed Applications on the Internet: Overview of Microsoft's .NET Platform. In *17th International Parallel and Distributed Processing Symposium (IPDPS)*, 2003.

[16] I. Foster and C. Kesselman. *The GRID 2: Blueprint for a New Computing Infrastructure*. Morgan-Kaufmann, second edition, 2003.

[17] G. Hutchison. DB2 Web Services: The Big Picture, IBM developerWorks article. August 2002.

[18] D. T. Liu. The design of GridDB: A Data-Centric Overlay for the Scientific Grid. In *Proceedings of the 30th VLDB Conference*, pages 600–611, 2004.

[19] Z. H. Liu, M. Krishnaprasad, and V. Arora. Native Xquery processing in Oracle XMLDB. In *ACM SIGMOD Conference*, pages 828–833, 2005.

[20] B. Ludscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the KEPLER system. *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows*, to appear, 2005.

[21] K. Mensah. Virtualize Your Oracle Database with Web Services, Oracle Technical Article. November 2005.

[22] T. J. Muller(Ed.). *AEROACOUSTIC MEASUREMENTS*. Springer-Verlag, 2002.

[23] S. Narayanan, T. Kurc, U. Catalyurek, and J. Saltz. Database Support for Data-driven Scientific Applications in the Grid. *Parallel Processing Letters*, 13(2):245–271, 2002.

[24] M. Nicola and B. V. der Linden. Native XML Support in DB2 Universal Database. In *Proceedings of the 31st VLDB Conference*, pages 1164–1174, 2005.

[25] M. A. Nieto-Santisteban, J. Gray, A. S. Szalay, J. Annis, A. R. Thakar, and W. J. O'Mullane. When Database Systems Meet the Grid. In *Proceedings of the 2005 CIDR Conference*, January 2005.

[26] S. Pallickara, B. Plale, S. Jensen, and Y. Sun. Structure, sharing and preservation of scientific experiment data. *IEEE 3rd International workshop on Challenges of Large Application in Distributed Environments (CLADE)*, July 2005.

[27] M. Rys. XML and Relational Database Management Systems: Inside Microsoft SQL Server 2005. In *ACM SIGMOD Conference*, pages 958–962, 2005.

[28] S. Shankar, A. Kini, D. J. DeWitt, and J. Naughton. Integrating databases and workflow systems. *ACM SIGMOD Record*, 34(3):5–11, September 2005.

[29] H. Stockinger. Distributed Database Management and the Data Grid. In *Proceedings of the Eighteenth IEEE Symposium on Mass Storage Systems and Technologies*, pages 1–12, 2001.

[30] K. Takeda, G. B. Ashcroft, X. Zhang, and P. A. Nelson. Unsteady aerodynamics of slat cove flow in a high-lift device configuration. *AIAA Paper 2001-0706, AIAA Aerosciences Meeting*, 2001.

[31] R. Walter. *The Rational Guide To SQL Server 2005 Service Broker*. Rational Press, 2005.

[32] V. Welch. Grid Security Infrastructure Message Specification. February 2004.

[33] M. A. Williams. Using .NET Stored Procedures in Oracle, Oracle Technical Article. November 2005.

[34] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *ACM SIGMOD Record*, 34(3):44–49, September 2005.