Actor Provenance Capture With Ganglia

Ian Wootten, Shrija Rajbhandari, Omer Rana, J.S.Pahwa
School of Computer Science,
Cardiff University,
UK
Email: {I.M.Wootten, S.Rajbhandari, O.F.Rana, J.S.Pahwa}@cs.cf.ac.uk

Abstract

Provenance is generally defined as the documentation of a process that leads to some result, and has long been recognised as being fundamental to the development of problem solving mechanisms within Grid environments. The knowledge of how a particular result has been derived is just as important as the result itself within many e-Science experiments. We present the concept of "actor" provenance, which provides detailed information concerning the state of an actor at a particular time. We also demonstrate how actor provenance differs from "interaction" provenance between actors. We describe how actor provenance may be represented and recorded using monitoring tools such as ganglia. This is explained using a number of use cases in a Bio-Diversity application.

1 Introduction

The use of Grid based technologies is increasingly commonplace as a mechanism to support problem solving in large-scale scientific computations. The benefits of using Grid computing infrastructure has proven to be useful in reaching increases of speed in achieving a particular result (generally through the use of high throughput computation), decreases in costs (by making use of resources made available by others) and addressing problems outside of scope of local homogeneous systems. Many varied application domains are making use of such technology, such as the search for black holes and gravitational waves [5], species distribution modelling [13] and aircraft maintenance [3]. Service oriented computing has made it possible to share resources where direct knowledge of implementation and access is hidden from users. In the TeraGrid project [1] for example, a cluster of machines may be viewed as a single compute service, hiding the details of the underlying network and operating system from the user.

Due to the distributed nature of many such experiments, results will often involve the use of complex workflow techniques, which require the composition of services located outside the control of a single administrative domain or scientist. The ability to track where a particular partial result has originated from is therefore highly important, as often the validity of the end result will depend on the accuracy of partial results produced along the workflow. In many existing workflow efforts, the focus is generally on the answer generated as an outcome of the workflow – i.e. what has been produced as an end result. It is likely that scientists will simply accept this outcome. There are situations, however, where a scientist may wish to probe results generated by intermediate services that have been used in a workflow session. The ability to trace documentation associated with a workflow session is therefore important to be able to fully understand the generated outcome. Such documentation is said to constitute "Provenance" of a result. Many applications have identified bespoke requirements for both provenance-related data and the architecture whereby it may be recorded [11].

The rest of this paper is organised as follows: Section 2 begins by defining provenance and presenting the existing work that has taken place in this area. Section 3 outlines the application scenario, with use cases obtained from the Bio-Diversity application to demonstrate actor provenance applicability. We describe how actor provenance may be recorded in a service oriented computing environment in section 4, using monitoring tools such as ganglia. In section 5 we define actor provenance and outline how it may be categorised, presenting how it may be represented using the Common Information Model (CIM) schema defined as part of the DMTF [2] consortium and the Global Grid Forum. In section 6 we perform an evaluation of the solution with reference to our use cases, and section 7 presents our conclusions.

[1] http://www.teragrid.org/

[2] http://www.dmtf.org/

2 Background

Provenance is the documentation of the process which leads to a particular result [14]. Such documentation associated with a process may be of two types: (i) documentation about interactions between actors, (ii) documentation about the internal state of each actor in the process. We consider an **actor** as any client or service involved in any single interaction during workflow invocation. It is also useful to note that a workflow graph often provides an abstraction of the process – i.e. a static description of services involved, not necessarily the actual services that were used in a particular execution instance. Groth [9] defines actor provenance as documentation that can only be provided by a particular actor pertaining to the process that led to the data. We make a clear distinction between interaction and actor provenance:

Interaction Provenance - The documentation of process concerning interaction between actors in a service oriented architecture. This data concerns the actual information which has passed between actors. Using such information, it is possible to reconstruct the overall result from any given workflow. Interaction Provenance may be recorded by the workflow enactment engine (by copying all messages exchanged between actors in a workflow to a repository), or it may be explicitly recorded by the actors.

Actor Provenance - That data which an actor and only that actor can submit regarding an interaction. Such information concerns the actor state at a particular time during an interaction. This may encompass the internal flow of data within an actor, or actor performance metrics. Actor Provenance can only be recorded by the actor, and not by the workflow enactment engine. An actor must therefore explicitly decide to make available such information to third parties. Actor Provenance would generally include the following:

Static data associated with a particular actor: such as service identity, name, owner, version, capability, etc. Such information is similar to that published by an actor in a registry service in a service oriented architecture.

Dynamic data associated with a particular actor: such as service execution time, uptime, availability, memory usage, etc. Such information needs to be recorded by the environment hosting the actor, and may be made available on demand. The accuracy of such dynamic data is dependent on the types of measurement tools being used.

Actor Provenance data may be used to either evaluate the behaviour of an actor over the past, or to predict the likely future behaviour of a actor. Coupled with interaction provenance, such data may be used to evaluate whether a particular actor is the cause of an inaccurate result.

2.1 Current approaches

Work upon provenance is currently being used in a variety of research domains. Within the Chimera virtual data system [7], provenance documentation is generated in a catalog format by explicitly representing derivation procedures used in a process along with derived items and methods to define and query data. The schema which represents this process documentation is pre-defined and does not allow for arbitrary data. Database composition has also been assessed to evaluate how data items may have contributed to particular query results [4], [6] but generally only focus on the data warehousing context.

More specifically, *actor provenance* is being used in areas to evaluate the state of actors in various research projects. In the myGrid project [3] for instance, provenance data is generated from bioinformatics experiments and classed into two major forms, *the derivation path* by which the results were generated from the input data and *annotations* associated with a particular object or collection of objects [8]. Such annotations may include elements of actor provenance such as version data for workflows and resources [15].

Triana [4] is an open source problem solving environment which allows users to form complex workflows from prewritten services, as well as allowing services to be defined by the user, or obtain services from external sources [10]. A similar method of annotating workflows, whereby data objects specify information to be recorded at each point in the workflow process has been created. In Triana this is recorded as meta-data associated with each service in the workflow, although the exact format of this meta-data is not specified.

While all these approaches are perfectly applicable to their individual domain, they are not transferable between them, due to the nature of the data intended to be captured. Although Chimera specifies a particular schema for provenance documentation, it does not allow for arbitrary definitions which are often necessary by workflow users. Triana goes the farthest to providing a generic capture method for meta-data relevant to any domain-specific workflow, but only provides this for the processes and objects involved in the local workflow and does not specify the format of this data. Our work differs by proposing a domain independent actor provenance capture method, which records the information upon the source actor involved itself. Using this method we are able to capture relevant actor provenance within any service oriented architecture.

[3] http://www.mygrid.org

[4] http://www.trianacode.org/

3 BioDiversityWorld

BioDiversityWorld (BDW) is a research project at Cardiff University that is exploring how a problem solving environment (PSE) can be designed and developed for biodiversity informatics in Grid environments [13]. Its aim is to provide scientists with tools with which they can readily access resources that were originally designed for use in isolation, composing these resources into complex workflows, and to ease the complexity of creating and introducing resources into the system.

Currently, work focuses on three main scenarios: biodiversity richness analysis and conservation evaluation (obtaining richness statistics and locality information for species); bioclimatic modeling and climate change (predicting the possibility that a species may become endangered or hazardous pests as a result of climate change); and phylogenetic analysis and biogeography (using phylogenies to interpret other biodiversity data distribution and morphology).

In all the scenarios, an investigator performs analysis on selected particular groups of organism using the BDW system. The data set used for these scenarios is sundews (Droseraceae), Leguminosae data (flowering plants of pea and bean family) retrieved from International Legume Database and Information Service (ILDIS) and a group of insects from Natural History Museum in London.

3.1 Bioclimatic Modelling

The main BDW scenario (bioclimatic modelling) shown in figure 1 begins with the generation of a taxonomy for the particular species' of interest. This is then queried against the Global Biodiversity Information Facility (GBIF) to obtain the locality information for the species. In parallel with this climate layers, containing estimations of such attributes as temperature and rainfall are obtained and selected to produce a 'climate envelope'. This is then used with a specific selected Open Modeller (OM) algorithm by interpolating the climatic data at the points of locality of the specimens, thereby producing a bioclimatic model. This model is then able to be projected upon a map of the world in order to make predictions of the anticipated effects of climate change upon biodiversity.

3.2 Use Cases

We present a number of use cases which will be used as a basis for the use of actor provenance. In each case we outline what Provenance information is needed.

Use Case 1: Execution Bottleneck - *A bioinformatition B downloads some locality information for a particular species from the GBIF database and runs the bioclimatic modeling experiment upon it. A later run of the same process yields an overall execution time which is far greater than the earlier run. B determines which of the processes involved caused the extension in execution time.*

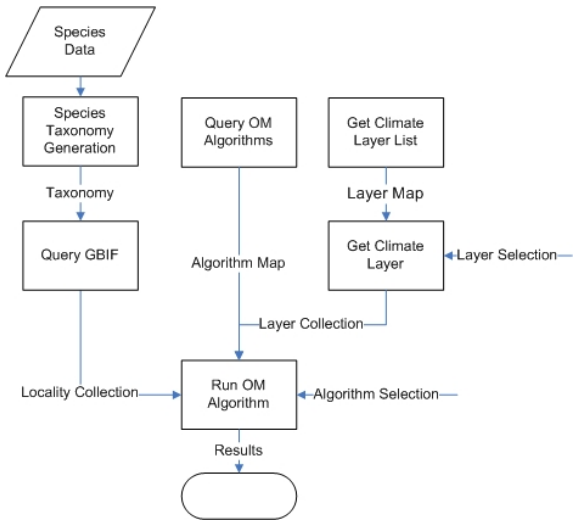

Figure 1. Bioclimatic Modelling within BDW

Through inspection of the execution times stored within actor provenance, B is able to determine which service/s caused the increased time for this particular process run.

Use Case 2: Result Accuracy - *A bioinformatition B downloads some locality information for a particular species from the GBIF database and runs the bioclimatic modeling experiment upon it. B determines whether the resultant projection image is one which is accurate and can be relied upon.*

As each actor provenance contains the accuracy to which an actor produces a given result, B is able to determine what the overall accuracy of the projection image shall be depending on how the bioclimatic process is constructed. In a similar manner B is also able to calculate the overall reliability that can be placed in the process which produced a particular projection image, through inspection of all the actors reliability records who are involved in the process.

Use Case 3: Service Throughput: *A bioinformatician B wants to run a large number of bioclimatic modeling experiments within a particular time frame. This is due to the GBIF data source only being available over that time frame to B. It is therefore necessary to choose a service that undertakes this modeling with a particular minimum throughput.*

Using prior throughput values, B is able to select a service that matches a certain throughput profile. This assumes that multiple bioclimatic modeling services are available for use.

4 The Actor Provenance Architecture

Our proposed method of Actor Provenance capture consists of a number of interacting components shown in figure 2. Data sources and analytic tools in the environment consist mainly of "legacy" resources which cannot be easily recreated. Resources may not have been originally intended for use together or exposure in a Grid or service oriented architecture. In order to standardize communication, they are wrapped by a Web Service which is tailored to that particular resources' inputs and exposed by a standard set of methods. These methods are invoked through use of the Triana workflow enactment and composition tool. Triana sees only the remote wrapper which features operations to poll the service for actor data (such as availability), as well as to invoke it. It only knows of input and response types from a service, due to pre-defined descriptions of such services held locally.

In order to capture actor provenance, provenance handlers are deployed upon the actor which we wish to monitor. Such handlers send copies of all messages which are sent to them to a provenance store for recording, as well as the destination for which they were originally intended. Acknowledgement messages are then sent by a provenance store to the sender to inform them of receipt. In this way, no changes are required to the application in question, enactment is performed as if provenance handlers did not exist. As a provenance handler is called, it checks the current message type for a request or response message, indicating the start or finish of a particular invocation. When found, registered monitoring tools (discussed in section 4.1) are polled for provision of actor state information to record provenance at this instance. As the store is primarily designed for interaction messages, actor state information is sent to a provenance store as additional provenance messages. In this way, we are able to capture both the information passed between actors (interaction provenance) as well as that actor state information (actor provenance) to be captured at a particular store.

Both the wrapper and enactment engine communicate via local and remote provenance handlers to record any information appropriate for the current invocation from both the service and clients perspective. The handler is also used to associate a particular actor with monitoring data that is recorded about the actor. This information is recorded to both remote and local stores, to ensure all involved parties have local copies of the data submitted to the remote provenance store. Actors within multiple domains may all submit to the same remote provenance store, or may have individual stores depending on the security constraints imposed by that particular domain.

The enactment engine may be invoking resources held within the same domain (using the same provenance handlers), or upon multiple domains (using multiple provenance handlers). It is important to note that not all the resources from which actor provenance data is to captured are within the control of a single enactment engine. Deployment of remote provenance handlers therefore may not be possible due to imposed restrictions on the remote resources. We therefore record as much information about an invocation as is possible on both the client and service, to obtain the clearest description of the state of either actor at a particular time.

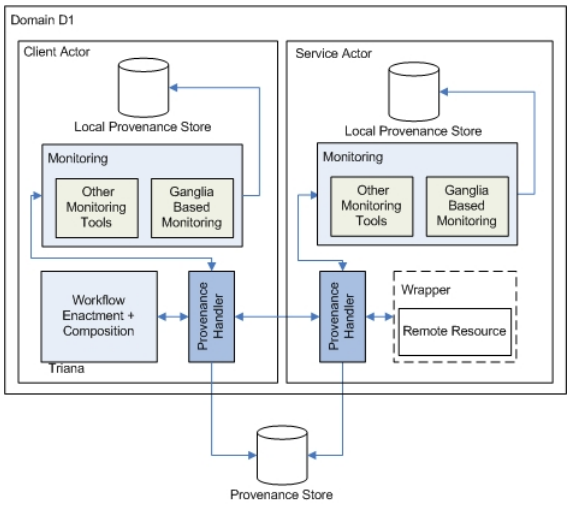

Figure 2. Actor Provenance Architecture

4.1 Monitoring Sources

Ganglia is currently being deployed for use within BDW to monitor application resource usage [2]. Ganglia operates through the use of a number of monitoring and meta daemons. It facilitates the monitoring and sharing of common state variables upon a number of nodes. Firstly, the **Ganglia Monitoring Daemon** (gmond) monitors changes in node state, announces changes to others/listens to changes from other nodes via use of a multicast or unicast channel, and responds to query requests. The **Ganglia Meta Daemon** (gmetad) provides federation of this state information across clusters using a tree of point-to-point connections.

Ganglia also allows us to define our own metrics with the ganglia metric tool (gmetric), to expand on those core metrics available by default. This can be done through use of the metrics currently available, or tell gmetric to send metrics obtained from other sources to listening gmonds. In this way, it is possible to record any state variable or metric derived from these variables and have them automatically added to the metrics returned in a query.

4.2 Provenance Store Communication

The actor provenance data to be recorded is communicated to a remote provenance store through use of the Provenance Recording Protocol (PReP) [9]. The PReP protocol has been designed as a method of recording the interaction between actors within a service oriented architecture. Interaction between a recording actor and a provenance store takes place in a number of stages. Message formats for determining which provenance stores should be used for a particular recording, invoking services, recording provenance and terminating a recording session are defined within PReP to facilitate such functionality. An additional provenance message also allows actor state to be recorded during the provenance recording phase. The result of invocation, as well as those derived metrics are wrapped in a message envelope and sent along with an identification parameter, to the appropriate provenance store using the additional provenance message. Confirmation that this particular message has been received is made by the provenance store using additional provenance acknowledgement message which contains the identity of the exchange.

5 Actor Provenance

In this section we describe actor provenance, listing the various types which may be recorded, along with their uses. It is possible to categorise the way in which actor provenance is generated into two distinct groups:

Recorded is the provenance information (or raw data) which is recorded about a service. This information can also be broken down further into *static* (that information which will not change throughout the life of the service, i.e. its name, location) and *dynamic* information (which will alter across the life of a service, i.e. elapsed time).

Derived are those metrics which can be derived from the recorded information. This includes execution time of a service, its reliability and availability, for instance. This information would only be recorded from monitoring data acquired for a particular service.

5.1 Actor Provenance Taxonomy

The actor provenance architecture provides the ability to record instances of service invocations at run time including information such as start time/end time of invocations, memory usage information, information to indicate invocation failure/success etc. Using the raw recorded information it is possible to calculate the actor provenance attributes that are termed as *derived actor provenance* and represent all metric types through a taxonomy). Recorded metrics are shown in *italics*, derived metrics are shown in **bold** and metrics available from Ganglia are shown in `teletype`.

A number of static metrics may be specified. In the case of our taxonomy, these include elements such as the *name*, *location* and *IP Address* of the actor recording the information. We may obtain this information through the operating system on which the actor resides and the workflow user. These static variables are used to identify the actor which produced some provenance data. Due to the large amounts of data which an actor will submit to a particular provenance store we also define a dynamic *ID stamp* and *Message Type*. The ID stamp is an identification number generated to indicate the interaction which this provenance handler invocation was a part of. This would be created by the provenance handler and used for the duration of a workflow enactment. The message type identifies if the message is the original client request or the subsequent response. These variables allow distinction between multiple runs of the same workflow involving the same actors.

Other dynamic metrics include *precision* and *completion status*. Precision defines the accuracy to which an actor can generate a result. This may be specified as the number of decimal places to which it specifies its result. It is therefore possible to determine the overall accuracy of a previously executed workflow, comprised of services all specifying their precision values. The completion status of a service returns a boolean specifying whether or not the service completed the invocation and returned a result or failed. These may be used to determine which actor contributed to an incorrect result in a workflow session.

Workload: Workload is measured from a service perspective and defined as the number of unfulfilled requests within a particular time frame. As we are able to determine the current *number of requests* and *responses* which a particular service may be dealing with at a point in time, we can also calculate the difference between these, giving the number of unfulfilled requests. If actor provenance reveals that a particular service is likely to have a high workload, it is likely to be a less favorable choice for a user. High workload could also indicate greater preference for a given service by other actors.

Reliability: The Mean Time Between Failure (MTBF) of a service, measured based upon the success rate of completing a request sent to a service. The actor provenance system could calculate success rate by utilising the recorded *completion information* of a service invocation over a period of time. If a service request has been successfully processed and the results are sent to the service requestor, then this is a successful completion of the request from the service provider's perspective. The information about whether the results reached the concerned requestor and the accuracy of the results could only be confirmed from the provenance handlers submitted provenance of that invocation to

the provenance store. It is likely that recently recorded actor provenance data would provide a more accurate reliability measure for a particular service at that time.

Availability: Availability is represented as a percentage of uptime of an actor in an observation period. Likelihood of the service availability depends on the recorded actor provenance data such as; (i) Service invocation information that indicates that the service has been used, (ii) time between failures of this service, such as the Mean Time To Failure (MTTF),(iii) Time taken to repair the service when it fails, defined as the Mean Time To Response (MTTR).

For example, if a service is requested frequently for a given period of time, the set of provenance data recorded at that time would be greater than at a time when the service is requested less frequently. In this context we need to address any effects frequency might have in measuring availability of a service at a given point in time. Within BDW, it is possible to monitor availability of wrapped resources through use of the poll operation defined for each wrapper, which returns an `int` indicating the availability of that resource.

Serviceability: This can also be referred to as "accessibility" or Mean Time To Response (MTTR), and is a measure indicating to what extent an available service is capable of accepting a user request. It is possible that a service is available but not accessible at a certain point in time, which makes it desirable to include it in our taxonomy. It can been seen that if a service has had low availability and a low accessibility, it may be more suitable to select an alternative. A service might decide to stop accepting user requests when its system latency increases, thereby affecting its performance. Using the actor provenance data, measured request frequency and performance during a given period in time would help to calculate the percentage of serviceability.

`Throughput`: Throughput is measured from a service perspective and is the number of successful invocations over the services lifetime. The system uptime is exposed once again through ganglia. Recording this would allow user to measure the throughput of a service.

System Performance: This is measured based on the actor provenance data collected over a period of time. It includes:

Latency: Time between the arrival of a service request and the request being served. This could be generated from the recorded *start time* and *end time* for a particular service invocation.

`Bandwidth`: The average number of bits submitted/accessed by a service per unit time.

`Memory usage`: current memory usage by the server at a particular time. Increase in throughput could be reflected by the server's increased memory usage pattern.

`CPU usage`: current CPU usage of the server at a particular time. A decrease in serviceability could be reflected by an increase in both the server's CPU and memory usage pattern.

Each of these variables is available to be monitored within ganglia, or is able to be derived from the data which ganglia provides. Recording this information within actor provenance allows users to determine if the host system meets the performance requirements for their application workflow.

5.2 Deriving Metric Data

The ganglia monitoring system provides a number of metrics which are well suited to recording the state of performance of a particular system. These consist of parameters to monitor the current state of the CPU, disk usage, memory and network (a full list of default monitored parameters is available at [2]). By making use of these metrics we are able to record actor provenance information which are then submitted via PReP to a particular provenance store. Due to the explicit monitoring of memory and CPU usage by ganglia, we are able to directly obtain a number of the performance metrics described in section 5.1 (such as memory and CPU usage). We now show how it is also possible to calculate other desirable derived data from the core metrics which ganglia provide. This experiment was conducted upon a single system featuring a 1Gz Pentium III Processor, 512Mb RAM and a 40Gb hard disk. The system ran using Ubuntu Linux with both Ganglia monitoring and meta daemons configured upon the same node so that it monitors itself. RRDTool [12] was used to generate the visual output.

$$Bandwidth = \frac{N\bar{X}}{R} \text{ (Normalised)} \quad (1)$$

Where:
N = average input rate to the network (bits per second)
$\bar{X}$ = average number of bits per packet (bits)
R = channel transmission rate (bits per second)

By using the monitored *bytes_in* and *pkts_in* ganglia metrics as shown in figure 3 and 4, we are able to determine the average size of packets being transmitted to our monitored actor. In this scenario, our given point would be on entry to the node. From this information, and the maximum network transmission unit (mtu) also available from ganglia, we are able to calculate the bandwidth as described in equation 1.

Asserted values for bandwidth will be calculated in this way each time a provenance handler requests it from a registered ganglia monitoring source. In addition to ganglia sources we are able to define other sources of monitoring information. These could encompass a whole other monitoring system to obtain a number of metrics, or a simple script to return a single parameter. Such a registry allows us to have different configurations across multiple domains and potentially alternative definitions for metrics.

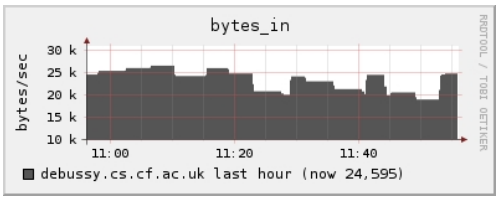

Figure 3. Number of bytes in per second

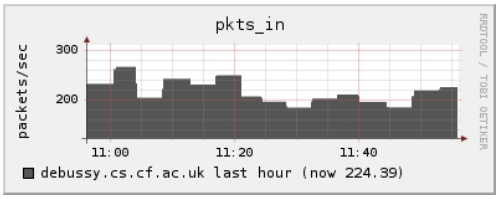

Figure 4. Number of packets in per second

5.3 Representing Actor Provenance

The Common Information Model (CIM) schema provides a 'Metrics Model' [1] for collection and management of dynamic metric information. This includes both a metric to gather 'Unit of Work' (UoW) information and also a 'Base Metric Concept'. This allows modeling of any type of raw or aggregated metrics, as well as that raw data related to response times as with the UoW.

Through use of the **CIM_BaseMetricDefinition** we are able to define the aspects of a particular metric we wish to record. For example, a template for the metric with the meta-data of the information we wish to record. **CIM_BaseMetricValue** would be an instance of such a definition, containing the values for that particular metric.

Using the schema it is possible to represent the metrics defined earlier in section 5.1. We are also able to define a SubUoWDef relationship to represent those UoW's which are comprised of others, increasing the granularity of recorded metric data. In the example which follows, granularity is of a very low level to merely illustrate how the metrics may be represented.

5.4 An Example

A 'UnitOfWorkDefinition' defines those units of work (i.e. invocations) associated with a logical element (abstractions used to configure and coordinate the physical or software environment, i.e. systems themselves). It represents the definitional components of the unit of work, and not the unit itself. 'UoWMetricDefintion' defines a metric that could be associated with a 'UnitOfWork'. It describes the metadata about the metric (value and behavior). In the case of actor provenance, this could be those metrics we wish to record about an invocation.

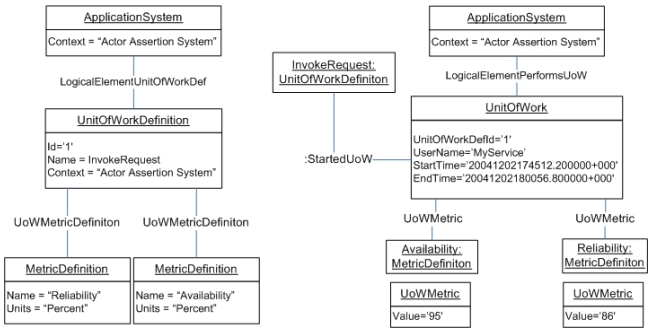

Figure 5. Class Instance Diagram

Figure 5 shows an instance of the metrics and the invocation request with which they are associated. In the example, values for the defined reliability and availability metrics have been recorded at 86% and 95% respectively. Both of these apply to the InvokeRequest UoW definition.

6 Evaluation

In this section we evaluate how and what inferences may be made about the differences in metric values found valuable for the use cases specified in 3.2 through use of actor provenance captured by ganglia.

Use Case 1: Execution Bottleneck - In order to determine overall execution time of a given workflow, the time at which provenance handlers are called is captured. As actor provenance is only recorded as service request and responses are made, we are able to determine the execution time of a particular actor invocation. If the recording of actor provenance reveals that the execution of the same workflow incurs differing overall execution times, we can use actor provenance to determine why this may be. If for instance, ganglia's recorded actor provenance reveals that for a particular actor, CPU and memory utilisation is far greater in a second invocation of a workflow, it may be that the system is still dealing with other processes, or multiple requests during the time the workflow was invoked. Such an insight could allow the bioinformatition to locate the problematic service and possibly better allocate resources in order to remedy this.

Use Case 2: Result Accuracy - As service actors specify their own accuracy, the combined accuracy of an invocation of the bioclimatic workflow is able to be calculated from the arrangement of actors in the workflow. Dependant on the level of accuracy which the bioinformatition B specifies as the threshold for the projection image, the image may be accepted as accurate or rejected as inaccurate. Through inspection of the provenance information ganglia is able to provide, the bioinformatition will be able to assess why ac-

curacy of a given projection image differs in two workflow runs. It may for instance reveal that disk space upon the actor was at a minimum, forcing a lower level of accuracy to be undertaken to enable the projection image to be stored.

Use Case 3: Service Throughput - We are able to calculate a particular services' throughput by utilising its completion information and ganglias' uptime variable. Calculating throughput in this way allows the bioinformatition to build a throughput profile over a time period and determine which of the modeling services has the lowest level of throughput appropriate for their use. Through further inspection of other ganglia performance metrics, it will be possible for the bioinformatition to decide whether just a low throughput value is enough to decide upon actor usage. There could for instance be a low amount of throughput due to the actors very low bandwidth.

7 Conclusions

Support for provenance in Grid computing is essential to allow a scientist to determine how a particular result has been derived, especially when the use of third party data or computation resources is involved. We consider such provenance information to be of two types: interaction provenance and actor provenance. We focus here on actor provenance, define it, and relate it to a monitoring tool widely deployed on many existing grid systems. Within problem solving and grid environments, situations where a finer understanding of why a particular service has failed, or why a particular service returns results with a very different response time can be analysed using our metrics. We present a domain independent architecture for actor provenance capture through a number of sources, along with specifying a mechanism by which it can be represented. Particular use cases from the BDW project are used to demonstrate how actor provenance information can be used in practise.

Acknowledgments

This research is funded in part by EPSRC PASOA project GR/S67623/01. Our thanks go to Dr R.J.White and Dr A.C.Jones from the BioDiversityWorld project at Cardiff University for provision of use cases specified in 3.2.

References

[1] Common information model (cim) metrics model, version 2.7, June 2003. Available at: http://www.dmtf.org/standards/documents/CIM/.

[2] Ganglia monitoring system, 2005. Available at: http://ganglia.sourceforge.net/.

[3] J. T. Austin, J and M. Jessop. Diagnostics and prognostics on the grid: the distributed aircraft maintenance environment project (dame). 2003.

[4] P. Buneman, S. Khanna, and W.-C. Tan. Why and where: A characterization of data provenance. *Lecture Notes in Computer Science*, 1973:316–??, 2001.

[5] D. Churches, B. Sathyaprakash, M. Shields, I. Wang, and I. Taylor. A Parallel Implementation of the Inspiral Search Algorithm using Triana. In S. J. Cox, editor, *Proceedings of UK e-Science All Hands Meeting*, pages 869–872. EPSRC, CD-Rom only, September 2003.

[6] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. In *The VLDB Journal*, pages 471–480, 2001.

[7] I. T. Foster, J.-S. Voeckler, M. Wilde, and Y. Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *SSDBM '02: Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, pages 37–46, Washington, DC, USA, 2002. IEEE Computer Society.

[8] M. Greenwood, C. Goble, R. Stevens, J. Zhao, M. Addis, D. Marvin, L. Moreau, and T. Oinn. Provenance of e-science experiments - experience from bioinformatics. In *Proceedings of the UK OST e-Science second All Hands Meeting 2003 (AHM'03)*, pages 223–226, Nottingham, UK, Sept. 2003.

[9] P. Groth, M. Luck, and L. Moreau. A protocol for recording provenance in service-oriented grids. In *Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS'04)*, Grenoble, France, Dec. 2004.

[10] S. Majithia, M. S. Shields, I. J. Taylor, and I. Wang. Triana: A Graphical Web Service Composition and Execution Toolkit. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 514–524. IEEE Computer Society, 2004.

[11] S. Miles, P. Groth, M. Branco, and L. Moreau. The requirements of recording and using provenance in e-science experiments. Technical report, University of Southampton, 2005.

[12] T. Oetiker. Rrdtool: Logging and graphing, 2005. Available at: http://people.ee.ethz.ch/ oetiker/webtools/rrdtool/.

[13] R. White, F. Bisby, N. Caithness, P. B. T. Sutton, P. Williams, A. Culham, A. J. M. Scoble, W. Gray, N. Fiddian, N. Pittas, X. Xu, O. Bromley, and P. Valdez. The biodiversityworld environment as an extensible virtual laboratory. In *UK e-Science All Hands Meeting, Nottingham, UK, EPSRC*, pages 341–344, 2003.

[14] F. Xu, A. Biller, L. Chen, V. Tan, P. Groth, S. Miles, J. Ibbotson, and L. Moreau. A proof of concept: Provenance in a service oriented architecture. Technical report, 2005.

[15] J. Zhao, C. Goble, M. Greenwood, and C. W. amd Robert Stevens. Annotating, Linking and Browsing Provenance Logs for e-Science, 2003. in conjunction with 2nd International Semantic Web Conference. online proceedings CEUR Vol 83.