

Workflow Management Systems

Marco Slot (1397117), Paul van Zoolingen (1284657)

Division of Mathematics and Computer Science
Vrije Universiteit, The Netherlands
{marco,pfvzooli}@few.vu.nl

Abstract- Workflow Management Systems (WMS) automate the scheduling, monitoring and execution of workflow applications. In this paper we discuss the challenges of managing workflow applications and different solutions that have been developed for workflow management systems. We will also discuss how workflow management systems are used in terms of workflow modelling.

1 Introduction

Early grid applications primarily focused on utilizing the massive parallelism facilitated by the grid for a single task. With the advance of e-Science the focus is shifting towards more complex, scientific applications that use various distributed resources, such as computational units, databases and scientific instruments, organized in a workflow.

More so than with parameter-sweep applications, managing workflow applications poses a number of complex issues. Scheduling becomes a major issue as different components with different requirements need to be scheduled on different resources at different times efficiently. Data has to be moved between components, different components have to be monitored and we wish to achieve some degree of fault tolerance. Workflow Management Systems are specifically built to address the automated management and execution of workflow applications.

In this paper we discuss the challenges of managing workflow applications and different solutions that have been developed for workflow management systems. Specifically we will look at solutions to modelling, scheduling,

execution and monitoring workflows and we will discuss fault tolerance for workflow applications.

2 Workflow modelling

Workflow management systems aid the programmer and the user in building and running workflow applications. To do so the user or programmer needs to provide the structure of the workflow and possibly additional execution information.

2.1 *Workflow structure*

The minimal information a workflow management system needs is a directed graph which represents the dependencies between nodes. A node can represent any active part of the workflow, it could for example be a single job, pre-staging a file, or an entire sub-workflow. The semantics of the graph is that a child may not start until all of its parents are done. The dependency graphs are referred to as DAG (Directed Acyclic Graph) when acyclic and non-DAG otherwise [1].

The basic graph already provides the control structures to run in sequence and in parallel. A workflow structure may also contain additional control structures such as a conditionals and loops.

2.2 *Setting constraints*

Depending on the workflow management system a number of constraints on the workflow structure can be set. The constraints can be very concrete, such as a specific resource, or more abstract, such as hardware requirements or QoS constraints.

Constraints are primarily meant for the scheduler which will have to take into account specific resource requirements and, for example, maximum execution times. A constraint can also be a specific resource on which a task needs to be performed, in this case the scheduler will not need to do any optimization but simply has to use that resource.

QoS constraints that are associated with workflows are time, cost, fidelity, reliability and security [5]. Constraints can be set for an entire workflow or a single task. In the current state-of-the-art the possibilities of setting QoS constraints are somewhat limited.

2.3 *Graphical workflow modelling*

To make workflow modelling intuitive for users (e.g. e-scientist) it is generally done using a graphical interface. The dependency graph is drawn by dragging and dropping components that can be defined using configuration panels. Some components are predefined. These could be various scientific analysis tools, or lower level actions such as upload/download a file.

2.4 *Language-based workflow modelling*

A workflow can also be defined using a, generally XML-based, workflow modelling language. Although mostly inconvenient to write it is a useful way to exchange workflows. At this time no standard workflow language exists. A workflow language that is being used extensively in business software is BPEL, Business Process Execution Language

[6]. The intended use of BPEL is to describe business processes in a programmatic way. It has powerful primitives such as message passing, service invocation and XML transformations. Part of the definition is model a workflow.

Figure 1 shows a sample BPEL workflow. Basic control structures such as sequence (sequence), parallel (flow), condition (switch) are available, but it also has more advanced structures such as while, try-catch and variables. It is in many respects similar to a programming language.

```
<sequence>
  <invoke ... />
  <flow>
    <invoke ... />
    <invoke ... />
  </flow>
  <switch>
    <case condition="xpath_exp">
      <invoke ... />
    </case>
    <case
      condition="other_xpath_exp">
      <invoke ... />
    </case>
  </switch>
</sequence>
```

Figure 1. Example BPEL Workflow

3 Workflow scheduling

Once a workflow model has been defined it can be given to a workflow execution engine which will then have to schedule the workflow.

If we wish to make efficient use of resources we need to plan ahead. For example, if we have a long job and a short job that can run in parallel, we do not want to long job to be scheduled on a slow resource while the short job is being scheduled on a fast resource.

3.1 *Scheduling strategy*

A workflow schedule can be optimized using different decision strategies. We do not necessarily want find the fastest way to schedule our workflow. In a market

environment where a cost is associated with the use of resources we could be more interested in the cheapest schedule. When we are dealing with sensitive data we may not care about performance either, rather we would be interested in finding the most trust-worthy resources (i.e. resources from trust-worthy organizations). Trust does not necessarily imply security, it may also cover reliability and fidelity.

Which strategy can be used depends entirely on the design of the Workflow Management System. Most workflow systems use a performance-driven strategy.

3.2 Performance estimation

To be able create a schedule optimized for performance we first need to have an estimation of the duration of individual parts of the workflow on various resources. The duration depends heavily on the hardware, software, resource utilization, network throughput, etc. There are various models that can be used to estimate the performance of an application on a specific resource given the a rough estimation, historical results (see 5.), simulation results **Fout! Verwijzingsbron niet gevonden.**, or analysis of the program itself. Information services can be used to give the final estimation or to provide information which the scheduler can use to make an estimation itself.

3.3 Creating the schedule

Optimal scheduling on is an NP-complete problem and therefore schedulers use heuristics or other optimization methods such as genetic algorithms [7]. The schedule can be created at the start of the application or even before. The advantage is that all the information is available when creating the schedule and a good schedule can be

found. However, as the grid environment changes rapidly the information used is outdated. An alternative is therefore to plan ahead for only part of the workflow or even to schedule just-in-time. Which method works best depends on many factors (e.g. the environment, the application) and is hard to predict.

3.4 Scheduling architecture

Scheduling complex workflows imposes a scalability problem. Workflow applications may use many different resources on different locations from across the globe and as such a central scheduler may cause large delays and the risk of failure increases.

Three different scheduling architectures are being used by workflow systems of which the centralized scheduler is the most common. Despite its scalability issues the advantage of the centralized scheduler is that it has information on all parts of workflow and can find the best schedule. A decentralized scheduling architecture [3] has much better scalability, but can only plan ahead for a small part of the workflow. As a compromise more systems are starting to use a hierarchical scheduling architecture [4]. Although this architecture can not protect against failure of the root, it can improve performance and still achieve a reasonable schedule because there is strict central control over which scheduler does what.

4 Workflow execution

For the execution of Grid workflow applications, different kind of files have to be moved or copied to multiple places in the Grid. Input files have to be staged to remote locations and the output files have to be returned later on. In workflow systems this data movement can be managed in an user-directed and an

automatic manner. In the user-directed way, the user can specify the data movement in the workflow himself and ignore the automatic mechanisms. Automatic data movement comes in three flavours; centralized, mediated and peer-to-peer. In the centralized approach the data movement between resources is transferred through a central point. Although this is easy to implement, it is not scaleable. In the mediated approach, a special data management service is used which keeps track of all the data in the system. This service can be queried by processes and used as a mediator to get the needed data. This approach is more scaleable and data can be retrieved for later use. Peer-to-peer is a principle where the data is moved between processes without a third-party service. This removes a possible bottleneck and can be used for large-scale intermediate data transfer. The downside is that is hard to implement and less suitable for times when intermediate data has to be monitored.

5 Workflow monitoring

A workflow management system does not execute the tasks itself, but just coordinates the execution of them by the grid resources, it needs information to guide the workflow and to bring the tasks to the suitable resources [14]. There are three dimensions of information retrieval: static information, historical information and dynamic information.

Static information refers to information that does not vary with time. It may include infrastructure-related, configuration-related, QoS-related, access-related and user-related information. Static information is used by Grid workflow management systems to pre-select resources during the initiation of the workflow execution.

Since during execution, environment variables and the current status of the workflow management systems may change, the Grid workflow management system also needs to identify dynamic information such as resource accessibility, workload, and performance during execution time. This information could also task execution information and market related information such as dynamic resource price.

Historical information is obtained from previous events that have occurred such as performance history and execution history of Grid resources and application components. Workflow management systems can analyze historical information to predict the future behaviors of resources and application components on a given set of resources. Historical information can also be used to improve the reliability of future workflow execution. Several information services are available for accessing static and dynamic information about Grid resources, such as Network Weather Service (NWS) [13] and Monitoring and Discovery System (MDS) [11].

6 Fault tolerance

In a Grid environment, workflow execution failure can occur for various reasons. They could for example result from a lack of homogeneity in the execution environment configuration, services being unavailable, or overloaded systems. Grid workflow management systems should be able to identify and handle failures and support reliable execution in the presence of concurrency and failures. Workflow failure handling techniques into two different levels; the task-level and workflow-level.

Task-level techniques mask the effects of the execution failure of tasks in the workflow. Task-level techniques have

been extensively studied in parallel and distributed systems and can be divided into retry, alternate resource, checkpoint/restart and replication.

The retry technique simply tries to execute the same task on the same resource after failure [12]. The alternate resource technique submits a failed task to another resource [12]. The checkpoint/restart technique moves failed tasks transparently to other resources, so that the task can continue its execution from the point of failure [9]. The replication technique runs the same task simultaneously on different Grid resources to make sure that the task execution at least one of the replicas does not fail [8,10].

Workflow-level fault tolerance techniques manipulate the workflow structure to deal with faulty conditions. Just like with the task-level techniques, there are multiple workflow-level techniques, alternate task, redundancy, user-defined exception handling and rescue workflow. The first three approaches assume there is more than one implementation for a certain computation with different execution characteristics.

The alternate task technique executes another implementation of a certain task if the previous one failed, while the redundancy technique executes multiple alternative tasks simultaneously. The user-defined exception handling allows the users to specify a special treatment for a certain failure of a task in workflow. The rescue workflow technique system [9] ignores the failed tasks and continues to execute the remainder of the workflow until no more forward progress can be made. Then, a rescue workflow description, which indicates failed nodes with statistical information, is generated for later submission.

7 Conclusion

We have discussed the modelling of workflows and the most important problems addressed by workflow management systems in scheduling, monitoring, execution and fault tolerance.

A Workflow Management System needs a workflow model which is generally created using a graphical interface. To find an efficient execution schedule a WMS relies heavily on heuristics applied to performance metrics obtained from information services. To deal with scalability issues in workflow scheduling distributed scheduling architectures have been developed. Workflow Management Systems also need to provide a method for moving data between components, e.g. peer-to-peer or centralized. To achieve a degree of fault tolerance a WMS may simply restart or move individual components, or use a workflow-level technique such as a rescue workflow.

Although each of the problems presented in this paper have been addressed in some way most solutions are very basic and are more often derived from necessity than from proper research.

Most research in scheduling has so far focussed on optimizing performance even though that may very often be of lesser importance. Especially market-driven scheduling is a wide open topic which has been given little attention. Alternative scheduling strategies may also open up viable research questions, scheduling based on QoS requirements is largely unaddressed.

Monitoring has so far been limited by the underlying architectures and information services which are oriented at individual nodes and tasks. Workflow Management Systems try to obtain

information on their workflow execution by querying all individual nodes. No information services that can provide, for example, the network throughput between nodes in a partial workflow currently exist.

Fault tolerance solutions are mostly similar to those used in parameter-sweep applications providing task-level fault tolerance. A lot of work is still to be done to provide higher level fault tolerance, especially facilities to create distributed snapshots are desirable.

8 Bibliography

- [1] J. Yiu, R. Buyya,
"A Taxonomy of Workflow Management Systems for Grid Computing"
Jnl. Of Grid Computing vol. 3, pp. 171-200, Springer 2005
- [2] G. Zheng, T. Wilmarth, P. Jagadishprasad and L.V. Kalé,
"Simulation-based Performance Prediction for Large Parallel Machines",
Int. Jnl. of Parallel Programming, vol. 33, No. 2-3, pp. 183-207, Springer 2005.
- [3] I. Taylor, M. Shields, I. Wang,
"Resource Management of Triana P2P Services",
Grid Resource Management, Kluwer 2003.
- [4] J. Cao, S.A. Jarvis, S. Saini, G.R. Nudd,
"GridFlow: Workflow Management for Grid Computing",
3rd Intl. Sym. on Cluster Computing and the Grid, IEEE CS 2003.
- [5] J. Cardoso, J. Miller, A. Sheth, J. Arnold,
"Modeling Quality of Service for Workflows and Web Service Processes",
Web Semantics Journal.: Science, Services and Agents on the WWW, Vol. 1, No. 3, pp. 281-308, Elsevier 2004.
- [6] "Business Process Execution Language for Web Services version 1.1"
http://www.ibm.com/developerworks/library/specification/ws-bpel/
- [7] R. Prodan, T. Fahringer,
"Dynamic Scheduling of Scientific Workflow Applications on the Grid: A Case Study",
Symp. on Applied Computing, ACM 2005
- [8] J.H. Abawajy, "Fault-Tolerant Scheduling Policy for Grid Computing Systems", *18th International Parallel and Distributed Processing Symposium (IPDPS'04), Santa Fe, New Mexico, pp. 238-244, April 26-30, 2004.*
- [9] DAGMan Application,
http://www.cs.wisc.edu/condormanual/v6.4/2_11DAGman_Applicaitons.html (December 2004).
- [10] S. Hwang and C. Kesselman, "Grid Workflow: A Flexible Failure Handling Framework for the Grid", *12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03), Seattle, Washington, June 22-24, 2003.*
- [11] S. Fitzgerald, I. Foster, C. Kesselman, G. Von Laszewski, W. Smith and S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Computations", *6th IEEE Symposium on High-Performance Distributed Computing, Portland, OR, IEEE CS, Los Alamitos, pp. 365-375, August 1997.*
- [12] Taverna User Manual.
http://taverna.sourceforge.net/manual/docs.word.html (December 2004).
- [13] R. Wolski, N.T. Spring and J. Hayes,
"The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing" *Future Generation Computer Systems, Vol. 15, No. 5-6, pp. 757-768, 1999.*
- [14] R. Yahyapour, P. Wieder, A. Pugliese, D. Talia and J. Hahm, "Grid Scheduling Use Cases", *White Paper, Global Grid Forum, 19 July, 2004.*