ASKALON

Abhishek Naik[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: ahnaik@indiana.edu

project-002, March 28, 2017

ASKALON is a Grid application development and computing environment which aims to provide a Grid to the application developers in an invisible format [1]. This will simplify the development and execution of various workflow applications on the Grid. This will not only allow a transparent Grid access but also will allow the high-level composition of workflow applications. ASKALON basically makes use of five services: Resource Manager, Scheduler, Execution Engine, Performance Analysis and Performance Prediction [2].

Keywords: Cloud, I524

https://github.com/cloudmesh/sp17-i524/blob/master/paper2/S17-IR-2022/report.pdf

This review document is provided for you to achieve your best. We have listed a number of obvious opportunities for improvement. When improving it, please keep this copy untouched and instead focus on improving report.tex. The review does not include all possible improvement suggestions and if you sea comment you may want to check if this comment applies elsewhere in the document.

1. INTRODUCTION

A Grid is basically a combination of hardware and software that provides reliable and pervasive computation abilities.

I see your point, but unless it is cited from some known authoritive person, try to avoid making analogies about major concepts.

Grid based development methodologies are used for application development. These Grid-based development methodologies emphasize providing the application developer with a non-transparent grid. ASKALON provides such invisible grid to the application developers.

you lost me here: non-transparent means developers see it. invisible grid means developers do not see it. which one is true for ASLALON?

While using ASKALON, the Grid work flow applications are made using Unified Modeling Language (UML) based services [1]. Besides this, it also enables integration of the workflows that have been created programmatically using languages such as XML. ASKALON typically requires some middleware for this.

seems like you can implement ASKALON with either UML or XML with some middleware. can be more clearer...

2. DESCRIPTION

When using ASKALON, the users need to create Grid workflow applications at a higher level of abstraction using the Abstract Grid Workflow Language (AGWL) which is based on XML. This representation of the workflow is later passed on to the middleware layer for scheduling and exection. Figure 1 shows the typical architecture of ASKALON.

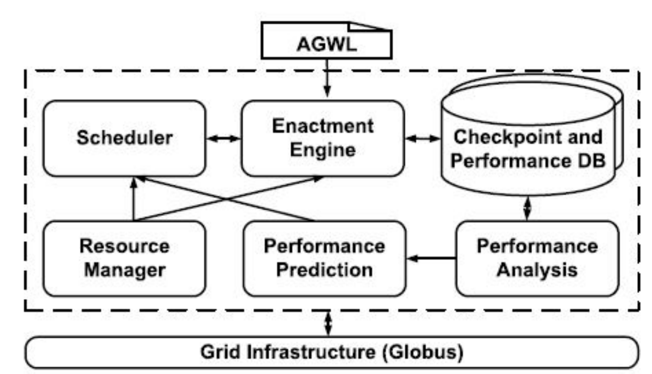

Fig. 1. ASKALON Architecture [1]

As shown in the figure, the major service components are the Resource Manager which is used for management of the various resources, the Scheduler that is used for scheduling the various workflow applications onto the Grid, the Execution engine which provides reliable and fault tolerant execution, the Performance analyser that analyses the performance and detects bottlenecks and the Performance predictor that estimates the execution times. The following paragraphs provide a brief description about these services:

2.1. Resource Manager

The Resource Manager service is responsible for the management of the resources like the procurement, allocation, reservation and automatic deployment. It usually works hand in hand with the AGWL. In addition to this management of resources, the main task of the Resource manager is to abstract the users from low-level Grid middleware technology.

2.2. Scheduler

The Scheduler service is responsible for the scheduling (or mapping) of various workflow applications onto the Grid using optimization algorithms and heuristics based on graphs. In addition to this, it monitors the dynamism of the Grid infrastructures thorough an execution contract and adjusts the optimised static schedules accordingly. The Scheduler thus provides Quality of Service (QoS) dynamically. It usually uses one out of the three algorithms - Heterogeneous Earliest Finish Time (HEFT), Genetic or Myopic. HEFT algorithm schedules the various workflows by creating an ordered lists of tasks and then mapping the tasks onto the resources in the most efficient way. Genetic algorithms are inspired by Darwin's theory of evolution and work using its the princples of evolution while the Myopic algorithms are basically just-in-time approaches, since they make greedy decisions that would be best in the current scenario.

2.3. Execution Engine

The Execution Engine service also known as the Enactment Engine service performs tasks such as checkpointing, retry, restart, migration and replication. It thus aims to provide reliable and fault tolerant execution of the workflows.

2.4. Performance Analyser

The Performance Analyser service provides support to the automatic instrumentation and bottleneck detection within the Grid workflow executions. For example, excessive synchronization, load imbalance or non-scalability of the resources might result in a bottleneck and it is the responsibility of the Performance Analyser to detect and report it.

2.5. Performance Prediction

The Performance Prediction services predicts the performance, i.e., it emphasises on the execution time of the workflow activities. It uses the training phase and statistical methods for this.

3. WORKFLOW GENERATION

ASKALON basically offers two interfaces: graphical model based on UML and a programmatic XML based model [1]. The main aim of both these interfaces is to generate large-scale workflows in a compact as well as intuitive form:

3.1. UML-modeling based

ASKALON allows creation of workflows via a modeling service similar to UML diagrams. This service combines Activity Diagrams and works in a hierarchical fashion. This service can be implemented in a platform independent way using the Model-View Controller (MVC) paradigm. This service can then be shown to contain three parts: a Graphical User Interface (GUI), model traverser and model checker. This GUI in turn comprises of the menu, toolbar, drawing space, model tree and the properties of the elements. The drawing space can contain several diagrams. The model traverser, as the name suggests, provides a way to move throughout the model visiting each element and accessing its properties. The model checker on the other hand is responsible for the correctness of the model.

3.2. XML-based Abstract Grid Workflow Language

The Abstract Grid Workflow Language enables the combination of various atomic units of work called as activities. These activities are interconnected through control-flow and data-flow dependencies. Activities can in turn be of two levels: activity types and activity deployments. An activity type describes the semantics or functions of an activity, whereas the activity deployment points to a deployed Web Service or executable. AGWL is not bounded to any implementation technologies such as the Web Services. Also, the AGWL representation of a typical workflow can be generated in two ways: automatically via the UML representation or manually by the user. In both the cases, AGWL serves as the input to the ASKALON runtime middleware services.

4. COMPARISON WITH OTHER COUNTERPARTS

Few experiments have been carried out using the seven Grid clusters of the Austrian Grid [3] and a group of 116 CPUs. Figure 2 represents performance of the individual clusters, wherein each cluster aggregates the execution time of all the workflows executed on a single CPU. As can be inferred from the figure, the fastest cluster is around thrice faster than the slowest one.

DAGMan [4] is a scheduler that is used for Condor jobs that have been organized in the form of a Directed Acyclic Graph (DAG). Scheduling doesn't use any specialized optimization techniques and is done simply using matchmaking. Fault tolerance is done using rescue DAG that is automatically generated whenever some activity fails. As against this, the ASKALON checkpointing also takes care of the fact when the Execution Engine itself fails. Thus, the checkpointing provided by ASKALON is more robust compared to that by other counterparts like DAGMan.

ASKALON differs in many respects compared to other projects like Gridbus [5] and UNICORE [6]. In ASKALON, the AGWL allows a scalable specification of many parallel activities by using compact parallel loops. The Enactment Engine also enables handling of very large data collections that are generated by large-scale controls and data-flows. The HEFT and genetic search algorithms that the ASKALON scheduler implements, are not used by the other projects, like the ones mentioned above. The Enactment Engine also provides checkpointing of workflows at two leves for restoring and resuming, in case the engine itself fails. In ASKALON, a lot of emphasis is laid on the clear separation between the Scheduler and the Resource Manager. It thus proposes a novel architecture in terms of physical and logical resources and thus provides brokerage, reservations and activity type to deployment mappings.

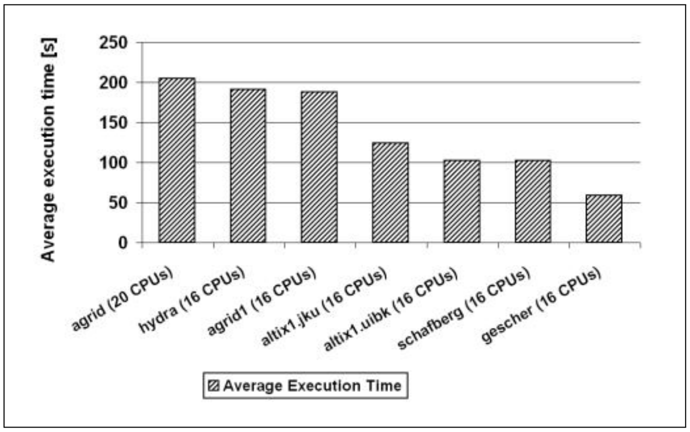

Fig. 2. Performance of Austrian Grid clusters [3]

5. ASKALON IN BIG DATA PROJECTS

ASKALON has been used in the design of Meteorological Stimulations in the Cloud [7]. In order to deploy the application on a distributed Grid infrastructure, the simulation code was split into a workflow called the RainCloud, which was represented using ASKALON. Figure 3 denotes the graphical representation of this workflow.

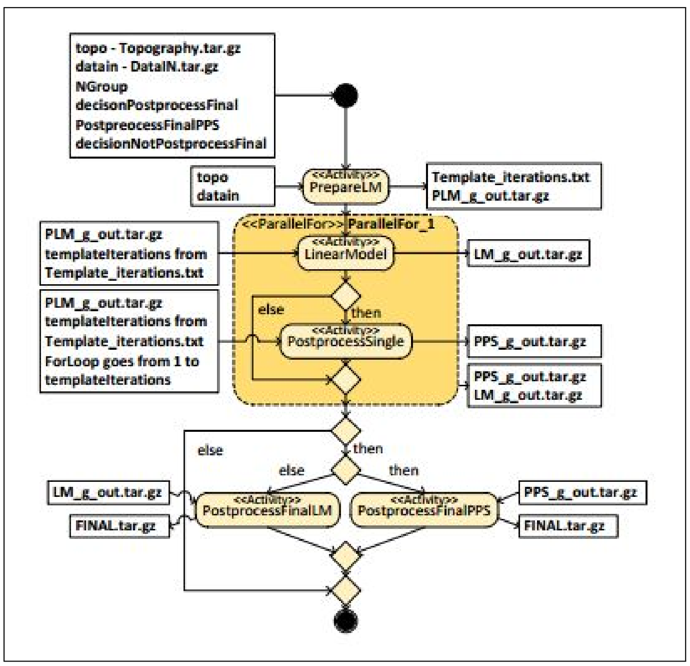

Fig. 3. Graphical representation of the Meteorological workflow [7]

This workflow was flexible in the sense that it could be easy extended to suite the needs of some other projects. For example, this workflow setup was extended for an investigation of precipitation/snow accumulation on the Kongsvegen glacier on Svalbard, Norway, as a part of the SvalClac project [7].

6. CONCLUSION

ASKALON supports workflow integration using UML and also provides an XML based programming interface. This effectively abstracts the end user from the low level middleware technologies. The Resource Manager handles the logical and physical resources and the workflow activities to provide features such as authorization, Grid resource discovery, selection, allocation and interaction. Scheduler makes use of some algorithms like HEFT or other genetic algorithms which deliver high performance. It highly benefits from the Performance Prediction service which in turn depend upon the training phase and the statistical methods used. The Execution Engine handles data dependencies and also working on high volume data - something that is highly useful in Big Data related applications and projects. The Performance Analyser analysis the performance benchmarks. Typically, the overhead of ASKALON middleware services which consist of the Resource Manager, Scheduler and Performance prediction are usually constant, thereby requiring less execution time holistically.

Thus, to conclude, we focused on ASKALON as a Grid application development environment. We also saw the various architectural components of ASKALON as well as the comparisons amongst the different Grid clusters that were used. We also saw some Big Data use cases wherein ASKALON was used and the flexibility with which the workflow setup using ASKALON was extended to support different projects.

some clarifications needed for the term "middleware". There is this middleware layer in charge of scheduling and executing ASKALON applications. Also, there are so called low-level grid middleware technologies. Are they the same middleware? Other than this, well written.

REFERENCES

[1] I. Taylor, E. Deelman, D. Gannon, and M. Shields, *Workflows for e-Science*. Springer, 2006, pp. 462–471.
[2] F. Pop, J. Kolodziej, and B. DiMartino, Eds., *Resource Management for Big Data Platforms*. Springer Nature, 2016, pp. 49–50.
[3] "Austrian-grid-project," Web Page, accessed: 2017-3-20. [Online]. Available: http://austriangrid.at
[4] "Dagman-manager," Web Page, accessed: 2017-3-19. [Online]. Available: http://www.cs.wisc.edu/condor/dagman
[5] R. Buyya and S. Venugopal, "Gridbus technologies," in *The Gridbus toolkit for Service Oriented Grid and Utility Computing*, 2004. [Online]. Available: http://www.cloudbus.org/papers/gridbus2004.pdf
[6] "Unicore technologies," Web Page, accessed: 2017-3-20. [Online]. Available: http://www.springer.com/gp/computer-science/lncs
[7] G. Morar, F. Schuller, S. Ostermann, R. Prodan, and G. Mayr, "Meteorological simulations in the cloud with the askalon environment," in *Euro-Par 2012 Parallel Processing Workshops*, vol. 7640, 2012. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-36949-0_9?CFID=916124568&CFTOKEN=29254584