ViGs: A Grid Simulation and Monitoring Tool for ATLAS Workflows

Aaron T. Thor[1], Gergely V. Záruba[1], David Levine[1], Kaushik De[2], Torre J. Wenaus[3]

[1]Department of Computer Science and Engineering, [2]Department of Physics
University of Texas at Arlington

[3]Brookhaven National Lab

Abstract – *With the recent success in transmitting the first beam through Large Hadron Collider (LHC), generation of vast amount of data from experiments would soon follow in the near future. The data generated that will need to be processed will be enormous, averaging 15 petabytes per year which will be analyzed and processed by one- to two-hundred-thousand jobs per day[1]. These jobs must be scheduled, processed and managed on computers distributed over many countries worldwide. The ability to construct computer clusters on such a virtually unbounded scale will result in increased throughput, removing the barrier of a single computing architecture and operating system, while adding the ability to process jobs across different administrative boundaries, and encouraging collaborations. To date, setting up large scale grids has been mostly accomplished by setting up experimental medium-sized clusters and using trial-and-error methods to test them. However, this is not only an arduous task but is also economically inefficient. Moreover, as the performance of a grid computing architecture is closely tied with its networking infrastructure across the entire virtual organization, such trial-and-error approaches will not provide representative data. A simulation environment, on the other hand, may be ideal for this evaluation purpose as virtually all factors within a simulated VO (virtual organization) can easily be modified for evaluation. Thus we introduce"Virtual Grid Simulator"(ViGs), developed as a large scale grid environment simulator, with the goal of studying the performance, behavioral, and scalability aspects of a working grid environment, while catering to the needs for an underlying networking infrastructure.*

1. Introduction

The PanDA (Production and Distributed Analysis System) [1] is being developed by the US ATLAS (A Toroidal LHC ApparatuS) project with the goal of managing and scheduling very large workflow production and analysis of experimental results, distributed across the US. The purpose of the PanDA system is to handle the ATLAS requirements for large scale scientific experimental event production, and the analysis processing of such events across geographically distributed locations. PanDA's objective is to fulfill the throughput, scalability, robustness, minimal operations manpower, and efficient integrated data/processing management requirements of the ATLAS project. In order to ensure a smooth transition to real data generated when the Large Hadron Collider (LHC) [2] becomes fully operational, thorough and detailed testing is essential now. Although PanDA is not yet processing large amounts of real data, pending the start of ATLAS datataking later this year, very large volumes of simulated data are being processed, making the management and use of resources across distributed grid farms very real. To better study the operational and scalability aspects of a computational grid environment such as PanDA, the ViGs simulator was conceived. The use of a simulator provides better control over experimental scenarios, and the added flexibility of modeling in either compressed or expanded time frame for analysis purposes. Moreover, it is almost impossible to accurately describe a complex real-world distributed grid environment with a mathematical model. A simulator also provides a tool to attain comparable results to real outcomes without wasting resources and network bandwidth, which could otherwise be put to more productive use. In addition, the use of a simulator enables testing of virtually any resource setup configuration without physically going through the changes. ViGs is a hybrid simulation tool with an internal simulation scheduling engine and client simulator that can interact with real grid scheduling entities over the network (such as the PanDA system). Thus the real PanDA system is used in this paper, and not a mere 'black box' model of it.

The rest of the paper is organized as follows: section 2 refers to related work; section 3 discusses the design considerations and components of the ViGs simulator. Section 4 goes into the details of the conducted experiments, the simulation process, as well as observations made. Section 5 presents results obtained from ViGs simulator. Finally, section 6 concludes the paper, outlining future work for the ViGs simulator.

2. Related Work

Prior to developing the ViGs simulator, we looked at several representative grid simulators with the intent to

978-1-4244-2872-4/08/$25.00 ©2008 IEEE

adopt them for our use, as it is often more time efficient to build a simulator on top of an existing platform, rather than building one from scratch. In this section, we discuss some of our findings during this process.

In OptorSim [3], the authors present a grid simulator that has been developed to test different optimization strategies using a predefined set of performance metrics, with their simulation environment based on the UK Grid for Particle Physics (GridPP). It has the capability of incorporating optimization of replication strategies through the use of Replica Optimization Agent, into a grid environment. The main difference between [3] and ViGs is that [3] is a simulator which mimics a working grid application whereas the ViGs simulator works with real applications by replacing the entire grid environment with a simulated environment, and provides the findings one would expect in a real world situation.

In CasSim [4], an abstract model simulator is described that dynamically selects scheduling heuristics to be adopted in the grid environment. It has the advantage of having the ability of adapting to both artificial (user created offline input) and grid (actual input from a working grid environment) applications. However, the type of applications suitable for [4] was limited to high level abstract usage, and since its main objective is on performing comparisons and studies of varying scheduling heuristics methodology, it is better suited for scheduling and resource allocation studies rather than being a candidate for job consumption applications on grid systems such as the PanDA system.

The GridSim simulator in [5] started life as a toolkit for the purpose of modeling and simulating scheduling and resource management for grid applications. It has a strong ability to simulate a vast range of resource types and configurations, which is essential for a grid environment simulator. On top of that, it has a good foundation for network modeling, which is another vital part for a working grid. It also provides support for resource reservations, simultaneous job submission to the same resource, and resource modeling in both time and space-shared modes. However, it cannot interact with real scheduling systems during simulations.

GangSim [6] is a grid environment simulator which models the main components within a working grid environment, which consists of monitoring, job submission, scheduling, and usage policies. It is capable of modeling usage policies at both site and virtual organization levels, hence enabling it to study the effects of various policies under varying grid environments. The main focus of this simulator is to study the performance and effects of varying policies within a grid environment. However, the simulator needs models for all entities and thus cannot interact with an existing grid environment.

The MONARC toolset [7] is a process oriented discrete event simulator used to simulate LHC experiments. It is designed based on mathematical models and measured parameters to simulate a working computational grid environment using Java threads. Key components such as data model, data processing model, networking, job arrival patterns, and system architecture are abstracted to be modeled in the simulated environment. One key difference between this and the ViGs simulator is that ViGs is a standalone customizable grid environment simulator with the ability to consume real jobs from a working system, hence providing the valuable capability to perform stress testing of a working grid environment, seeking out real potential bottlenecks in a working environment.

VIEW [13] is a Scientific Workflow Management System (SWMS) based on Service Oriented Architecture (SOA) paradigm. It has the capability of providing a system's functionality as a Web service, hence supporting loose coupling services. On top of that, the use of Web services allows for service abstraction, reusability, discoverability, and interoperability across multiple SWFMSs. One key difference between VIEW and our proposed system is that VIEW is a purpose built user-interaction / parameter intensive scientific workflow system with a high dependency on the use of Web services. Since the PanDA system already uses substantial web interfaces on a machine, we needed a simulator with minimal web traffic generation so as not to interfere with the study results of the system.

The Emulab [14] software provides an emulated environment through clever allocation and multiplexing of physical resources to time- and space- shared emulated system. Through its tight integration with physical hardware, it supports the testing of applications on real operating systems and software without the need of any customization to the applications, while yielding actual performance results. It would have been an ideal candidate for our application if not for the scalability requirements of our studies. As a typical PanDA environment consists of somewhere between six to eight thousand nodes, we needed a system which has the capability of handling job requests from such nodes. On top of that, we needed the ability of scaling up when the need arises – especially as the number of participating nodes in PanDA production system grows.

In this paper, we seek to provide a simulation platform capable of consuming the workload of actual jobs provided by the PanDA production system environment, so as to allow the studying of performance and behavioral aspects of the system, as well as identifying any potential weakness within the system. It is necessary for the simulator to possess the ability to replace an entire functional grid environment, simulating all the resources and networking infrastructure which is used in a grid environment of today. As a result, the ViGs simulator was conceived.

3. Design considerations

In this section, we will first provide an overview of the PanDA production system, followed by a discussion of the ViGs architecture and job definition.

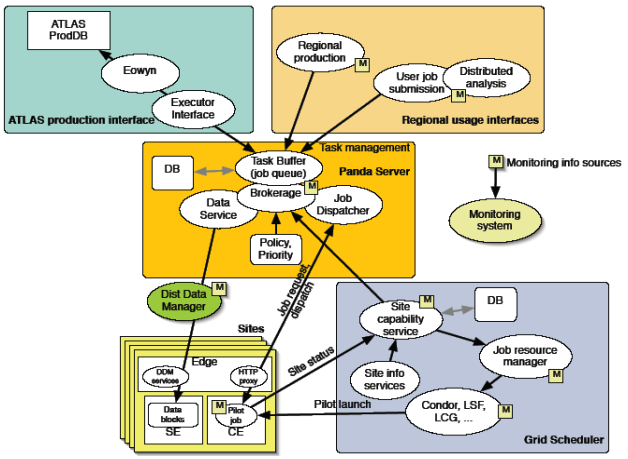

Figure 1 - PanDA architecture

3.1. PanDA Overview

A quick overview of the PanDA production system is shown in Figure 1. The PanDA production system is essentially a Many-Task Computing (MTC) [12] system consisting of numerous compute or data intensive tasks, scheduled and processed on various computing resources spanning across multiple administrative boundaries around the world. It consists of the following components: ATLAS production interface, Regional usage interface, Monitoring system, Grid scheduler, PanDA server, and Participating sites, which includes research institutions around the world. Although we tried to capture the best representative 12-hour snapshot of the PanDA performance in Figure 2, the reader is reminded that a 12-hour representation of system performance is by no means a comprehensive representation of PanDA production performance. It serves to provide the reader with some basic information on the PanDA system, such as the total number of participating nodes at any point in time, how widely distributed the experiments are conducted (e.g. Canada (CA), France (FR), United Kingdom (UK), and United States (US) etc.), which stresses the importance of proper network modeling. Due to the various natures of jobs submitted, the typical job processing time ranges from ½ hour to 48 hours. Moreover, due to the nature of job locality [9][10][11], jobs often have the tendency to arrive in bursts. Hence, a period of 12-hour with more submitted short jobs might give the impression of superior system performance, whereas other periods where bursts of long jobs submitted might give the false impression that the system performance has deteriorated considerably. In a typical PanDA operation, each node from a participating site submits what is called a pilot job request directly to the PanDA server. A pilot job can be seen as one possessing computational and data intensive characteristics of that of a MTC job, which is scheduled and processed on computing resources crossing administrative boundaries. For the sake of simplicity the inner workings of the PanDA server have been omitted. For more details on how the PanDA system works, readers are referred to [1]. Upon the completion of input datasets pre-placement process, jobs are delivered to the worker nodes prior to the initiation of jobs execution (a typical LHC job takes from ½ hour to 48 hours to complete, depending on the job type). After jobs complete, the resulting datasets has to be "staged out" successfully to be tagged as 'finished'.

Figure 2 shows a snapshot of a PanDA production job summary over a 12 hour period. From the collected data, we can see that there were a total of 7908 active nodes available to process jobs, with 14323 running jobs and 16236 finished jobs within the 12 hour period. The failure rate was 14%, which is a little higher than its typical value of 10% for the PanDA production jobs. However, readers are reminded that this is only a snapshot of the PanDA summary over a period of 12 hours, and it is by no means a complete representation of its performance in the long run. PanDA Production job performance may fluctuate due to unforeseen circumstances at times, as will be discussed in the next section. To get a better picture of the PanDA system performance, we have included plots of running, finished/failed jobs over a one-month period [8].

Production job summary, last 12 hours (Details: errors, nodes)

Cloud Information	Nodes	Jobs	Latest	Pilots (3hrs)	defined	assigned	waiting	activated	running	holding	transferring	finished	failed	tot	trf	other
Overall Production	7908	81902	04-15 05:41	10807	0 / 0	2158 / 0	42 / 0	28922 / 0	14323 / 0	2731 / 0	14996 / 3464	16236 / 0	2534	/ 19	14%	0% 13%
CA	909	9764	04-14 23:41	501	0	0	0	3692	1845	85	1008 / 0	3037	96		3%	0% 3%
DE	1219	14568	04-14 23:41	680	0	361	0	5391	2793	43	2020 / 0	3560	398		10%	0% 10%
ES	221	1980	04-14 23:40	32	0	0	0	1022	380	329	82 / 0	148	16		10%	0% 10%
FR	1802	15905	04-14 23:41	4886	0	462	0	3608	1971	1965	4121 / 1001	3514	263		7%	0% 7%
IT	201	2449	04-15 05:41	150	0	0	0	1275	87	6	458 / 63	548	75		12%	0% 12%
NL	266	7348	04-14 23:41	426	0	31	11	1425	565	11	3904 / 2368	205	1206		85%	0% 85%
UK	1716	11653	04-14 23:41	1248	0	1056	15	4796	2843	66	1179 / 31	1613	97		6%	2% 4%
US	1396	16435	04-14 23:41	2556	0	216	31	6990	3531	225	1972 / 0	3231	261		7%	1% 7%
TW	178	1800	04-14 23:40	328	0	32	0	723	308	1	252 / 1	380	103		21%	0% 21%

Figure 2 - Panda Production job summary

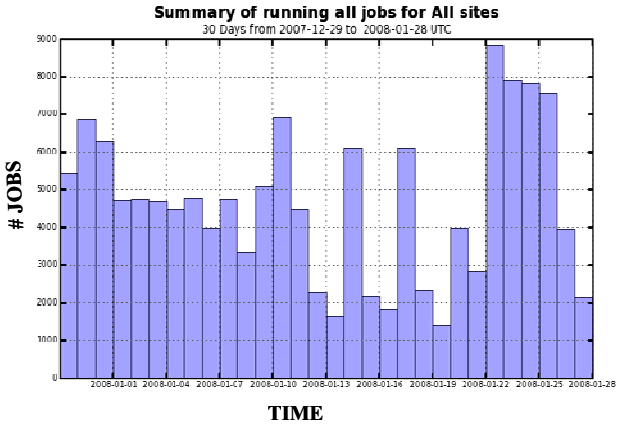

Figure 3 - Summary of PanDA running jobs over a one month period

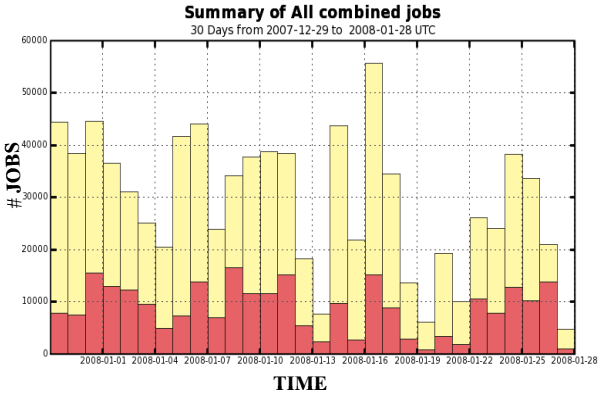

Figure 4 - Summary of Panda finished and failed jobs over a one month period

Figure 3 depicts typical running plots while Figure 4 shows plot of finished and failed jobs over the same period. From the data collected in Figure 4, the averaged failure rate is calculated to be around 30% with a monthly job completion rate of 610,350 (for that particular month). The total number of jobs received over the period sums up to approximately 872,000 jobs, which averages to around 29,000 jobs per day. One of the main explanations for the observed differences between Figure 2 and Figure 3, 4 can be attributed to the fact that since the PanDA system is still in the development and testing phase, factors such as upgrades (e.g. software and database upgrades), job outages (e.g. running out of jobs to fulfill pilot job requests), system instabilities (system, network, and power outages) etc. play an essential role in affecting the performance of the PanDA system. For instance, several snapshots of Figure 2 have been taken during the course of writing this paper, each with different characteristics. On a good day, the failure representation may go as low as 5~6%. On some rare occasions, however, the failure rate has gone as high as 47% with around 500 finished jobs.

3.2. ViGs overview

One of the most challenging tasks encountered when building a simulator is to determine if the simulation model is an accurate representation of the system being modeled. Although a detailed simulation model may often be preferred over an abstract representation of its real world counterpart, it may sometimes become the Achilles' heel of the entire experiment due to Mother Nature's rule of uncertainty. It is almost impossible to simulate every detail of a system without compromising some other details within the system. On the other hand, an overly abstracted representation might leave a researcher with insubstantial experimental results. So here we are faced with a dilemma: How much detail to include in our simulator and what to abstract? Our approach to the problem is to create a hybrid simulator which provides us with a tool to be able to manipulate and fine tune the represented resources, while maintaining minimal interference with the targeted source of interest. In addition, the hybrid simulator has to possess the ability to perform simulation over the network in order to interact with the objective system.

ViGs's heart is a discrete event simulator written in C++ which simulates the job distribution and processing on clusters of computers used in a grid environment, such as the PanDA system. However, unlike a conventional discrete event simulator, ViGs has the ability to connect to a real scheduling system, thus allowing for more precise emulation of every event during the course of simulation. ViGs is capable of working with the current version of the PanDA system with minimal modifications (mostly configuration file changes). As the main goal is to provide a mechanism for studying the performance, behavioral, and scalability aspects of a working grid environment, while catering to the needs for an underlying networking infrastructure, we focus our attention to the consumption of PanDA jobs, simulated execution within the environment, as well as monitoring the status of data movement within the simulation. In order to study the scalability and performance threshold of the PanDA system, the ViGs simulator offers the ability to generate different VO topologies, varying the distribution pattern of resources at each distributed site, and their processing capabilities, simulations in either real-time or other user specified expedited simulation conditions, as well as I/O file staging and job execution failure rates. Figure 5 shows the interaction between ViGs simulator and the outside world. The underlying network topology follows that of a typical grid environment, with hierarchical layout (gateway/site, clusters/nodes). Network link speed within individual sites may be customized, as well as the network infrastructure linking participating sites together. To simplify network traffic routing, shortest-distance / minimum-hop methodology has been adopted for determining the best path to route

traffic from one point to another. Future improvements would include incorporation of workflow models, along with randomly downed links to simulate broken links. The current version of ViGs has a simplified 'exponential backoff' built in, which exponentially increases the time taken for data to route through when too many failed data delivery has been detected (to simulate an overwhelmed node). Our goal for future ViGs versions is to include a smart traffic rerouting mechanism which takes into account when the time/cost of taking a shortest path exceeds that of a longer, less congested path.

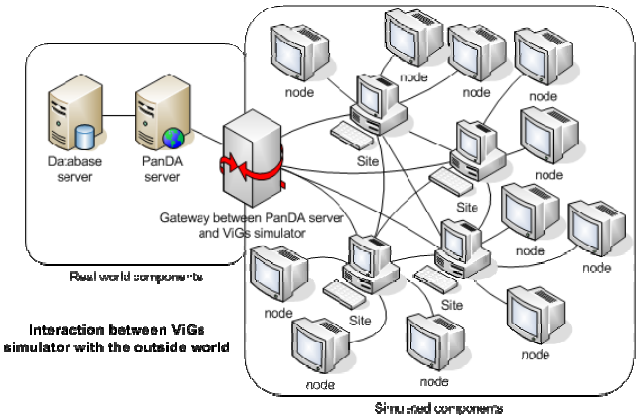

Figure 5 - - ViGs Interaction diagram

Specifications for each individual node within a site may be customized based on processor speed, available memory and storage capacity. This provides a better representation of sites consisting of machines with heterogeneous specifications. At the time of writing, we are working on including randomly downed machines to simulate downed machines, with customizable frequencies to represent machine and other hardware failures within the system. On top of that, failed jobs would have to be reassigned, along with possible input dataset pre-staging to another available location when necessary.

3.3. ViGs jobs

In ViGs, resource nodes receive job tasks based on responses from prior pilot job request submissions. As each node submits its pilot requests independent of one another, the inter-arrival time of subsequent pilot job requests as seen by the PanDA server is assumed to form a sequence of Independent and Identically Distributed (IID) requests. Hence the ViGs simulator generates pilot requests which follow that of a Poisson distribution. The job execution process follows that of a closed queuing network with each node dedicated to one job at any point in time (one process per CPU). Only upon completion of the job execution will the node submit subsequent pilot requests for new jobs. The assignment of tasks is described as $Task_{ijk}$ where:

n = number of sites, where $0 < i \leq n$

m = number of nodes at respective site, where $0 < j \leq m$

p = job number at node j, where $0 < k \leq p$

The time it takes to execute a job is defined as: Execution time = $Exe_{Task_{ij}}$ which follows a uniform distribution (a parameter to the simulation) over [1/2, 48] hours. The simulation executes until all assignable jobs available in the system are consumed. Therefore, the total simulation time is limited by the time it takes for the last processor node to complete processing the last job till completion, and is represented by:

$$\arg\max_{k} f(k) \in \left\{ k \mid \forall_{ij} : \sum_{0}^{k} Exe_{Task_{ij}} \right\}$$

To ensure we have enough jobs to feed the ViGs simulator, over 500,000 activated jobs are generated and stored in the MySQL database server prior to each simulation process.

4. Experiment

4.1. System overview

The experiment was conducted primarily using 3 dedicated servers: Database server, PanDA server, and Simulator sever respectively. All 3 servers have Intel Xeon 2.66GHz dual CPU with a total of 2GB memory, running Red Hat Linux version 2.4.21-32.ELsmp. The Database server is a dedicated MySQL server with a working copy of mysqldump obtained from the ATLAS database, i.e., with real job descriptions. It handles all database related transactions during the simulation. PanDA server is a duplicate of the running PanDA server with minimal configuration changes, but without the Distributed Data Management (DDM) data movement support (For more information on DDM, users are referred to [1]). We chose to omit the use of the DDM module since all simulations are conducted within the 3 servers, the complexity of DDM data movement was deemed not necessary at this time, and has been substituted with simulated data movement within the simulation process. Simulator server is where the ViGs simulator is running, which simulates the virtual network of computational grid, consisting of sites and processing nodes within the ATLAS system.

4.2. ViGs simulation

Before the simulation process, ViGs goes through an initialization process which sets up the simulation environment, network topology, and participating sites and nodes (each with a unique IP address used for status tracking) to simulate. We stress the importance of

including a proper network topology and design because it plays an important role in the real world, and is essential for proper operation of the simulator. In a grid environment where jobs are submitted across different countries and continents, the routes and distances traversed play a non-trivial role in the job requisition process. Imagine the scenario where two sites submit job requests at the same time: assuming everything else being constant, the site closer to the destination will usually reach first. So in cases where maximum-1 number of connections (queue) has been reached, the earlier request would be accepted as the last queued request whereas the latter slower request would be dropped, hence requiring job resubmission.

4.2.1. Initialization

ViGs begins the simulation process by first contacting the MySQL database server to determine the number of participating sites to generate, as well as any additional site information available. This is followed by reading a network configuration file which contains pre-computed location mapping of each site, as well as propagation distances amongst all sites. For simplicity, the total numbers of sites have been limited to a maximum of 10 for our current studies, each with 100 2.4GHz processor nodes and 1024MB memory. However, the ViGs simulator is capable of simulating heterogeneous sites, each with a different process, memory and storage configuration. The task of site generation may be configured to follow a random distribution function, or manually configured by the user. After the environment setup process, ViGs continues to read from its simulation configuration file to check for additional user specified simulation settings to include. A special module is used to monitor the simulated time against a clock in order to maintain accurate simulation time. In an attempt to create a more realistic scenario, all simulated timings (other than the simulation clock) have been set to operate within a standard deviation margin instead of following a hard edge-driven clocking system.

4.2.2. Simulation

After generating the underlying network topology as well as all participating sites and their respective nodes (please refer to section 3.2 for details), the simulator sends a signal to all (simulated) sites to signify the go-ahead. Upon receipt of the signal, each participating site will inform its respective nodes to begin submission of pilot job requests. Each node submits a pilot job request by spawning a child thread, which remains alive throughout the life of the job process. Threading is used in this case since each job request interacts with the real-world PanDA server, and the thread remains alive throughout the life of the job, enabling us to study the scalability and performance of both the PanDA and MySQL server under different loads.

During a job execution process, the respective node status is toggled to 'busy' until completion of the execution process along with any data stage-out process if needed, where it is toggled to 'free', allowing for subsequent pilot job request submission. At any point in time during the simulation process, the user can monitor the status of each node (site) by submitting a 'ping' request (through a web interface) to the particular node or site IP address.

4.3. Experiment Limitations

One physical constraint which we faced was the processing limitation of the Linux kernel and the Xeon processors we used to perform our simulations, which by default allow a default of only 300 threads to be generated (default 'ulimit –s 10240'). We attempted several different system resource setting configurations, and although we were successful in our attempts to alter the system resource allocation to achieve upwards of 1000 threads, we were discouraged by the fact that it dramatically affected system performance and stability due to over working of the dual core processors through swapping. As a result, we had to settle for running the simulations with 600 concurrent threads since our experience has showed us that our Xeon processor, along with its available memory, was only able to support around 600 (ulimit –s 5120) threads to run the simulations reliably. As a result, one of our goals for our future work is to support parallel instances of the ViGs simulator running on distributed machines.

5. Simulation Results

5.1. Results analysis

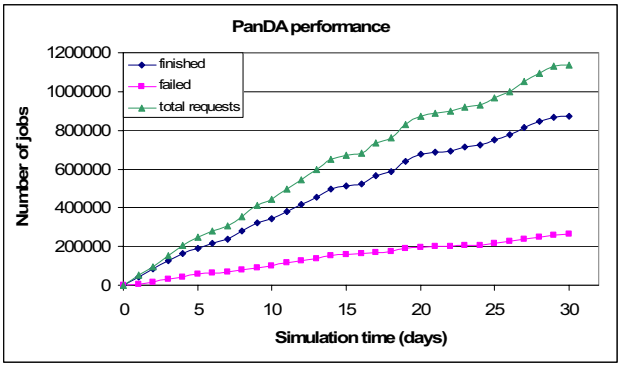

Figure 6 – PanDA server one-month plot

Figure 6 shows a typical PanDA server performance plot over a period of one month. From the collected data, we can observe that the PanDA server was able to consume an averaged total number of 1,134,650 jobs

while producing completed jobs of approximately 872,000 within the one month time frame. The error rate for this period was computed to be 23.15%, which is relatively higher to what was observed from Figure 2 (at 14%), as well as our aggregated simulation results at 18.16%. One of the reasons for this slightly higher failure rate can be attributed to the intermittent networking problems which have been taking place at Brookhaven National Lab (BNL), where the current PanDA server is located. The slightly higher failure rates should gradually subside as the networking problems are resolved.

During the experimentation process, a total of thirty simulated 1-month experiments were conducted, each with a different seed feeding the ViGs simulator. Figure 7 shows the aggregated plot of our simulation results. When comparing the experimental findings, we noticed a consistent correlation between the two results: with the ViGs simulator running on a machine at full capacity, it was only able to consume about 80% of the pilot jobs when compared with the PanDA system. This may be attributed to the following reasons: On a typical day, the PanDA Production system has a number of active participating nodes ranging from six to eight thousand, many with two, four, or even eight core processors performing the job execution tasks. On the other hand, our ViGs simulator testing was performed on a single dedicated dual-core 2.4GHz machine with limited processing capabilities, limiting our processing threads to a maximum of 600. With the high resource utilization nature of this simulation, attempts to squeeze out more threads often resulted in instability to the system due to overloading of the hardware. We believe that this phenomenon can be alleviated if we can create multiple instances of the ViGs simulator running in distributed mode. Figure 8 depicts the graphical comparisons of PanDA, PanDA at 80%, and ViGs simulator plots. More about this is discussed in section 6.2

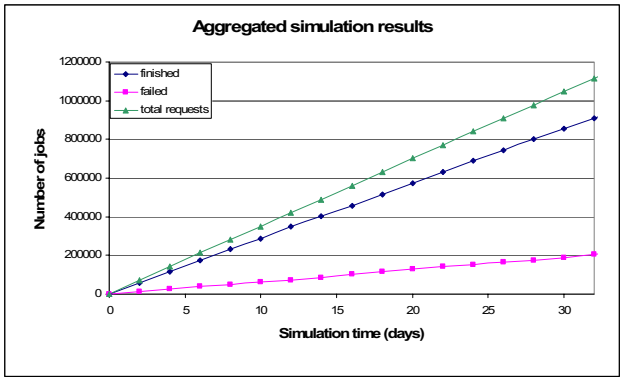

Figure 7 - ViGs simulation results plot

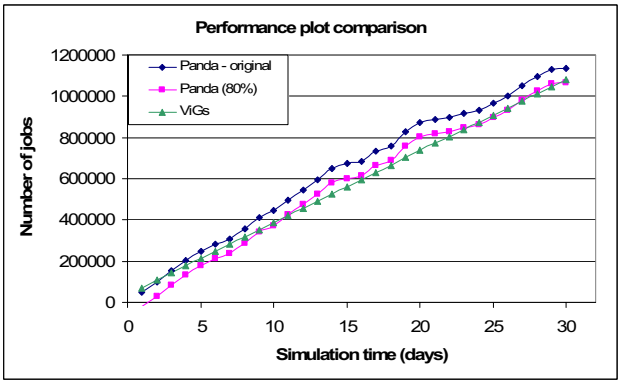

Figure 8 - PanDA vs ViGs plot

5.2. *Observations*

In this section, we make an attempt to find out how representative do the results obtained from the ViGs simulator compare with the real-world production PanDA system. We perform our comparison by forming a confidence interval for the difference between production PanDA and ViGs simulation results. The Paired-t Confidence Interval approach [10] since it provides a good measure for comparing differences between the expected results of two systems by pairing them together: For our case, real world performance results from the PanDA system and the simulation results obtained from the ViGs simulator.

We let X_j be the average of the observations in the j^{th} set of production PanDA data, and Y_j be the average of the observations in the j^{th} set of ViGs simulator resulting data, and $\mu_x = E(X_j)$ and $\mu_y = E(Y_j)$. Since we obtained a sample size of thirty results from our study, we let

$$Z_j = X_j - Y_j \Big\} \text{ for } j = 1..30 \qquad [1]$$

and

$$\left.\begin{aligned} \mu_x &= E(X_j) \\ \mu_y &= E(Y_j) \end{aligned}\right\} \text{ for } j = 1..30 \qquad [2]$$

With $E(Z_j) = \zeta$ as the value used to construct the confidence interval, we have:

$$E(Z_j) = \zeta = \mu_x - \mu_y \qquad [3]$$

By having $[l(\alpha), u(\alpha)]$ as the corresponding lower and upper confidence interval endpoints respectively, and using formula:

$$\left.\overline{Z}(n) = \frac{\sum_{j=1}^{n} Z_j}{n}\right\} \text{ for } j = 1..n, \text{ where } n = 30 \qquad [4]$$

and

$$\widehat{Var}\left[\bar{Z}(n)\right] = \frac{\sum_{j=1}^{n}\left[Z_j - \bar{Z}(n)\right]^2}{n(n-1)} \qquad [5]$$

, we can form an approximated 100 (1-α) percent confidence interval with:

$$\bar{Z}(n) \pm t_{n-1,\,1-\alpha/2}\sqrt{\widehat{Var}\left[\bar{Z}(n)\right]} \qquad [6]$$

Here, we attempt to compare the results obtained from the ViGs simulated model with the PanDA system by constructing a 95 percent confidence interval for ζ using the paired-t approach, so as to determine if the model is a good representation of the system:

From [4]:
$$\bar{Z}(n) = \frac{\sum_{j=1}^{n} Z_j}{n} = \frac{\sum_{j=1}^{30} Z_j}{30} = 946.15$$

From [5]:
$$\widehat{Var}\left[\bar{Z}(n)\right] = \frac{\sum_{j=1}^{n}\left[Z_j - \bar{Z}(n)\right]^2}{n(n-1)} = \frac{\sum_{j=1}^{30}\left[Z_j - \bar{Z}(n)\right]^2}{30(29)} = 52{,}766{,}750.99$$

From [6]:
$$\bar{Z}(n) \pm t_{n-1,\,1-\alpha/2}\sqrt{\widehat{Var}\left[\bar{Z}(n)\right]},$$

we get:

$$\bar{Z}(30) \pm t_{30-1,\,1-\alpha/2}\sqrt{\widehat{Var}\left[\bar{Z}(30)\right]},$$

resulting in:

$$-946.15 \pm 2.045 * \sqrt{52{,}766{,}750.99}$$

where

$$\left[l(\alpha), u(\alpha)\right] = \left[-15{,}801.1785,\ 13{,}908.8785\right]$$

Since $\bar{Z}(n)$ falls within interval $\left[l(\alpha)\ ,\ u(\alpha)\right]$, with $0 \in \left[l(\alpha),\ u(\alpha)\right]$, it shows that the ViGs simulation results is a valid representation of the PanDA system with approximately 95 percent confidence [10].

6. Conclusions and Future Work

6.1. Conclusion

In this paper, we have briefly discussed the motivations to develop the ViGs simulator for studying and testing the PanDA production system. We have also given an overview of ViGs and discussed some of its inner workings. Through running simulations, we have verified the simulator, and comparing it to the actual PanDA production system. We have also discussed findings from the simulation studies for both real-time and expedited rate simulations with regards to the PanDA production system. ViGs is showing great promise as a platform for testing and studying numerous aspects of the PanDA Production system, while eliminating the need to allocate resources for testing issues such as scalability and other performance issues. Furthermore the availability of ViGs enables development and fine tuning of the PanDA system even after the production system will start crunching on real data from the LHC.

6.2. Future work

In order for ViGs to become a full fledged test suite for PanDA, we have a list of improvement work planned out. First of all, as of the time of writing, the task of job assignment is performed by the built-in brokerage entity provided by the PanDA production system. As one of our future goals, we would like to develop our own scheduling and resource management module to compare against the current brokerage module in PanDA. We are also in the process of testing the limits and capabilities of the ViGs simulator's expedited simulation, tweaking with the configuration files and studying the results of the simulation, in hope of finding the optimal settings for ViGs. We are also working on a distributed version of the ViGs simulator which is capable of running the same simulation distributed over different machines, hence increasing the simulation capabilities of the current ViGs simulator.

References

[1] https://twiki.cern.ch/twiki/bin/view/Atlas/Panda

[2] http://public.web.cern.ch/public/content/Chapters/AboutCERN/CERNFuture/WhatLHC/WhatLHC-en.html

[3] D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, C. Nicholson, K. Stockinger, F. Zini, "UK Grid Simulation with OptorSim", UK e-Science All Hands

Meeting, Nottingham, UK, September, 2003 - hep.ac.uk
[4] E. Xia, I. Jurisica, J. Waterhouse, “CasSim: a Top-level-simulator for Grid Scheduling and Applications”, Proceedings of the 2006 conference of the center for Advanced Studies on Collaborative research, Toronto, Ontario, Canada
[5] R. Buyya and M. Murshed, “GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing”, Concurrency and Computation: Practice and Experience (CCPE), Volume 14, Issue 13-15, Pages: 1175-1220, Wiley Press, USA, November - December 2002
[6] C. Dumitrescu, I. Foster, “GangSim: A Simulator for Grid Scheduling Studies”,IEEE/CCGrid 2005, Cardiff, UK
[7] I. Legrand, H. Newman, “The MONARC toolset for simulating large network-distributed processing systems”, In proceedings of Winter Simulation Conference 2000
[8] http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query?dash=Plots
[9] R. Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling," Wiley-Interscience, New York, NY, April 1991.
[10] Averill M. Law, W. David Kelton, David W. Kelton, “Simulation Modeling and Analysis”, McGraw-Hill Science Engineering
[11] E. Medernach,” Job arrival analysis for a cluster in a Grid environment”, 1st Open International conference on Modeling & Simulation June 12th – 15th 2005 – ISIMA / Blaise Pascal University – France
[12] I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, B. Clifford. "Toward Loosely Coupled Programming on Petascale Systems", ACM/IEEE International Conference for High Performance, Networking, Storage and Analysis (SC08), 2008
[13] C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, F. Fotouhi “Service-Oriented Architecture for VIEW: a Visual Scientific Workflow Management System", in IEEE SCC 2008
[14] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, J. Lepreau. “Large-scale Virtualization in the Emulab Network Testbed”, Proceedings of the 2008 USENIX Annual Technical Conference, Boston, MA, June 2008.