PANDA MONITORING – A SYSTEM TO MONITOR HIGH PERFORMANCE COMPUTING FOR THE ATLAS EXPERIMENT DESIGN, DEVELOPMENT, IMPLEMENTATION AND DEPLOYMENT

By

PREM A THILAGAR

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment of the requirements for the
Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON
December 2007

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to my supervising professor, Dr. David Levine for his never ending encouragement and valuable guidance throughout my research. I am grateful to him for giving me the opportunity to work under him all this while, in him I saw the inspiration to be cool and composed. I am also grateful to Dr. Jaehoon Yu from the Physics department for giving me the opportunity to be a part of this great program between University of Texas, Arlington and Brookhaven National Labs, New York. I am also grateful to the Panda project's technical leader, Dr. Torre Wenaus, Brookhaven National Labs. He has constantly given his suggestions in improving my work during my tenure there.

I would also like to extend my thanks to my committee member Dr. Gergely Zaruba for his valuable time. I would like to extend my thanks to all my team members who were also a part of the program and were at Brookhaven National labs with me. I would like to extend my thanks to Barnett Chiu and Sudhamsh Reddy who were there to help me out in my research work. I would like to thank the whole Panda team at the Brookhaven National Lab. My heartfelt thanks go to my father Dr.S.Thilagar my mother Mrs.S.Sivarani and my friends, this work could not have been accomplished without their support.

November 26, 2007

ABSTRACT

PANDA MONITORING – A SYSTEM TO MONITOR HIGH PERFORMANCE COMPUTING FOR THE ATLAS EXPERIMENT DESIGN, DEVELOPMENT, IMPLEMENTATION AND DEPLOYMENT

Publication No: ______

Prem A Thilagar, M.S.

The University of Texas at Arlington, 2007

Supervising Professor: Dr. David Levine

Grid resources are gaining wide importance in the wake of experiments like ATLAS, they aid in a bigger goal which is to understand the complexities of nature. The need for a good monitoring system is realized as grid resources are being implemented widely. This work is an analysis of the existing monitoring system of Panda which is a grid middleware for the ATLAS experiment running at CERN. The thesis aims at identifying the key bottlenecks of the current monitor and speaks about the implementation of a new monitor for the same. The new monitor which is being designed with scalability and maintainability in mind shows how it will perfectly fit in for the changing needs of the panda. As panda is moving to the next generation as a generic grid middleware for other experiments running in the OSG sites in the United States it needs a new monitor which can efficiently scale and fit the changing needs. The new monitor is designed in Ruby on Rails and has numerous advantages over the existing one. The thesis deals with the design, development and implementation of this new monitor.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION TO MONITORING

1.1 Introduction

The quest for discovery is the driving force behind every new expedition by mankind. It is the same drive that motivates to explore and find components that can aid in the quest. The scientific approach to understanding a phenomenon has always resulted in the need for components that can aid in the bigger goal. The ATLAS experiment is one such quest which began 20 years ago. High performance computing has existed even before experiments like ATLAS started using computing resources at an extensive level. High energy physics experiments always have a reputation for generating huge amounts of data; the need for a component that can manage all the data and analyze it for valuable information was realized. This led to grids being deployed extensively, also other areas of science where there was a need for huge data processing adopted the same. Grids found their way to gene analysis and even assisted in studies related to extra terrestrial intelligence. The widespread popularity of the grids eventually led to a point where resource was directly linked to processing power and time; hence its precise and optimal operation became an essential need. As grids started sufficing the needs of most experiments being conducted it became necessary there be a system that could asses the performance of grids. Monitoring became a prime area of importance especially at a point where grids are being installed throughout the world.

1.2 History

Computers have made significant breakthrough since their commercial inception in 1951[1] , from being systems that were specifically used for projects related to defense and projects of strategic importance, computers have come a long way in making a significant impact in every field they have been used. The significant breakthrough in the popularity of computers and the Internet today can be attributed to the beginning of ARPANET [2] (Advanced Research Projects Agency Network) in 1969. ARPANET changed the view about computers and many academic institutions got together and started using them in a wide scale for the numerous benefits it offered including fast and effective data sharing.

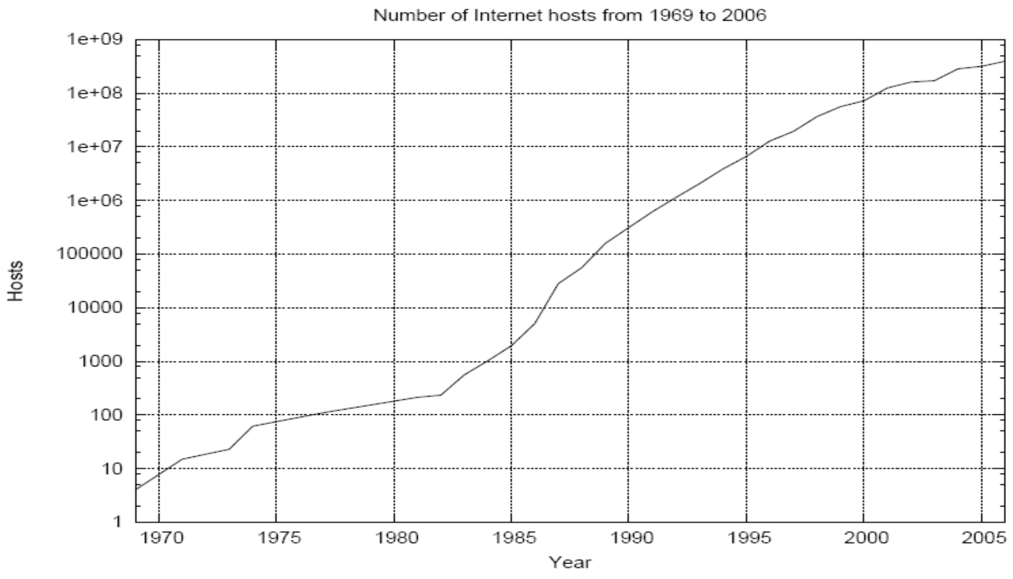


Figure 1.1: The Growth of Computer Hosts since Inception of Arpanet [3]

The formulation of ARPANET is an important milestone since it sparked off the idea of today’s Internet and also contributed significantly to innovations and improvisations in many related areas in the computer sciences.

Once the collective power of computers integrated together was illustrated it sparked off research in many areas that could make computers better such as databases and networks. As the advancements became widespread the need for systems that could assess and provide information about the components under study became important. Since then monitoring systems have played an important role in the development and assessment of a component that might be associated and working with a computer.

1.3 Monitoring

Monitor can be defined as a device or a piece of software that helps in inspecting and analyzing key performance constraints of the components of a computing system. Monitors can exist as an integral part of the computer or as an associated component that works in tandem. Monitoring in a computer can be classified mainly into two types [4],

- Hardware monitoring
- Software Monitoring

Monitoring in computing systems is not limited to the software alone but also extends to hardware. Hardware monitors are inconvenient for measuring and monitoring application programs [4], this is mainly due to the limitations like hardware monitoring is decoupled and exists separately for each device. The most advanced real-time hardware monitors are used in in-circuit emulators.

Software monitoring is more of an application-oriented perspective. In most cases software and hardware monitors work cohesively to facilitate the collection and prompt processing of the data, for example the calculation of the temperature of the processor and telling the user the maximum acceptable temperature. Additionally, software monitoring will explore and provide information about things that cannot be directly understood with the information from the hardware monitors alone.

Monitoring is a component which has been aiding many of areas in computer science. The most common of it were network, database and security. Monitoring in these areas has been a vital component in times when technological advancement was rudimentary [2]. Components like memory and other hardware were pretty expensive a few years ago. This cost factor established the idea that software had to be made in a manner that will enable optimized usage of the hardware; also the importance of using the established hardware to the fullest was realized. Creating necessary monitoring software to keep track of how efficiently things are being handled was the solution to make sure hardware resources are put to full use.

In today's world with the huge breakthroughs and advancements made in the semiconductor industry and computer hardware made available at cheaper rates, software monitoring still holds a prominent place. It has been adapted to the changing needs of the users. The development of the hardware industry has revolutionized the way people use computers. Data usage has scaled from a few kilobytes in the first generation of computers to gigabytes in the most recent ones [3]. The network which had capabilities of a few kilobytes/sec has also scaled to levels where it can handle huge data with ease.

The need for an efficient monitoring system can also be felt here, since the scale at which data is being dealt has exponentially increased and it has to be ensured that all these improvements in technology are being utilized properly.

1.4 Monitoring Areas

The typical areas in computing where monitoring has proven its importance are networking and databases [3]. Apart from these, monitoring applications also have aided security systems and intrusion detection systems to establish the effective maintenance of information integrity. The most common example of a monitoring system on a desktop can be cited as an intrusion detection firewall, which warns the user of any external network activity. Monitoring applications have also been very useful in the areas of distributed computing [3]. As huge amount of hardware resources is involved in distributed computing, effective and prompt information collection about how the resources are performing is indispensable.

The various features of a system which are, classical examples for monitoring are discussed below.

The CPU time monitoring primarily involves calculation of the total time an application utilizes to complete a certain task. This is commonly measured in the number of CPU clocks used by the respective process running [3]. It is also a most common benchmark which will have a direct impact on the performance of the system and thus looked up in most systems.

Disk monitoring is a feature where we can monitor the free and used space in the given system. Advanced monitoring systems, in disk monitoring also allow monitoring of the segments of the memory which are contiguous and those that are fragmented.

Network Bandwidth monitoring is one where we are allowed to monitor traffic that is going out or coming into the internal network. Network Monitoring, can be done at two places; extensive monitoring can be done both at the user side and at the ISP (Internet Service Provider) side. The topics of interest for the user would include the types of applications that are sending and receiving messages from his desktop while the provider will typically be more interested in the types of packet losses, the routing of packets across the network and the congestion at the network nodes to name a few. This kind of information could prove quite useful when the network is facing high inflow of network traffic at one particular node; a congestion detected early could be handled effectively by diverting the incoming traffic of a particular router, through some other nearby router which could be helpful to avoid high amounts of data and packet losses [4].

Database Monitoring has also proven to be an effective means to make computing systems more efficient. In a world where database servers handle a few thousand requests a second, it is very important to make sure we do not overload the server; this ensures the smooth serving of requests. Ideally there are a few servers (the datacenters) which handle the requests from the users and service them accordingly. All these datacenters will have the exact copy of the data and will have the capability to serve user requests in the same capacity. In certain cases, it might happen that most of the user requests might be routed to one particular server, resulting on a server overload. An effective monitoring system with a good load balancing scheme will mitigate the problems caused by overloading, most of the times this proves very effective in mitigating server outages and reduce the server downtimes effectively [3].

Web based monitoring by web sites for user information is a new paradigm of monitoring. User information has turned out to be a prime area of interest since this has begun to generate revenue based on the number of people who visit the site. There are quite a few monitoring tools, which allow the site administrators to gather information about the demographics of the users who are interested in the information posted in their web site. This information is later used to target the user with a specific item which might be of interest to him/her. This kind of information is considered vital to many of companies in the e-commerce world today, and they continuously keep monitoring the items that are of interest to the users and present the user with a wide variety of similar or associated products.

Monitoring as discussed in the examples above ranges from assessing how good a hardware is performing which could for example be keeping track of the CPU cycles [2] to more high level data such as the web pages a particular person is interested in. Each of these monitoring aspects has its own importance in specific areas associated to them.

1.5 Grid Monitoring

Monitoring, as in all the other areas, also plays a key role in computing grids. Grid-based monitoring becomes a very important feature since the resources for any grid are distributed over a wide area and all of them have to be efficiently monitored to see if they are working in the right fashion, moreover, if a problem occurs at one particular cluster, it is highly likely the other clusters may suffer the same problem. If a proper monitoring system is in place the outage when it happens at one cluster could be identified and the other systems could be adjusted in a manner that they are able to overcome the problem, thus saving valuable processing time [15].

1.5.1. Grid Experiments

Grid computing started gaining popularity with the beginning of new experiments. All the experiments mentioned below are increasingly using the services from grids in order to actually understand better their respective areas of research.

The famous ones among them being:

AMANDA [26] - (Antarctic Muon And Neutrino Detector Array), it is a telescope buried under the South Pole and helps in finding important information about the universe around us. Neutrinos have a really interesting characteristic for astronomers - they traverse long distances without being deflected, scattered or absorbed by interstellar magnetic fields, starlight or dust. The experiment studies Neutrinos.

LHC [7] -The Large Hadron Collider at CERN, Switzerland has been initiating a number of experiments, these would start rolling on a full scale once the LHC comes into production late 2008.A few of the experiments being hosted there are ATLAS[7],CMS[7], ALICE [7], LHCP [7] and TOTEM [7]. These experiments are all concentrated on various specific areas of particle physics and to understand the functional constructs of matter and the forces between them.

PET - Positron Emission Tomography, PET is a new technology for medical imaging using positron scatter. Small animals like mice and rats have always been used to test new medication because their genetic code is close to the human genetic code. PET allows studying all the effects of new medications without the need of dissecting animals-thus reducing the number of animals being used for study.

1.6 Introduction to Grid Computing

Grid computing can be viewed at a high level as one where computing becomes pervasive and the user or the client applications gain access to resources such as processors, storage, data etc, with little or no knowledge of where those resources are located or what the underlying technologies, hardware and operating system is. [3]

A grid can also been seen as a concept by which a cluster of computers are connected over the Internet, where each individual computer is purchased individually and combined together with the use of middleware and other software can produce similar computing resources as a many-CPU supercomputer, and at a lower cost [6].

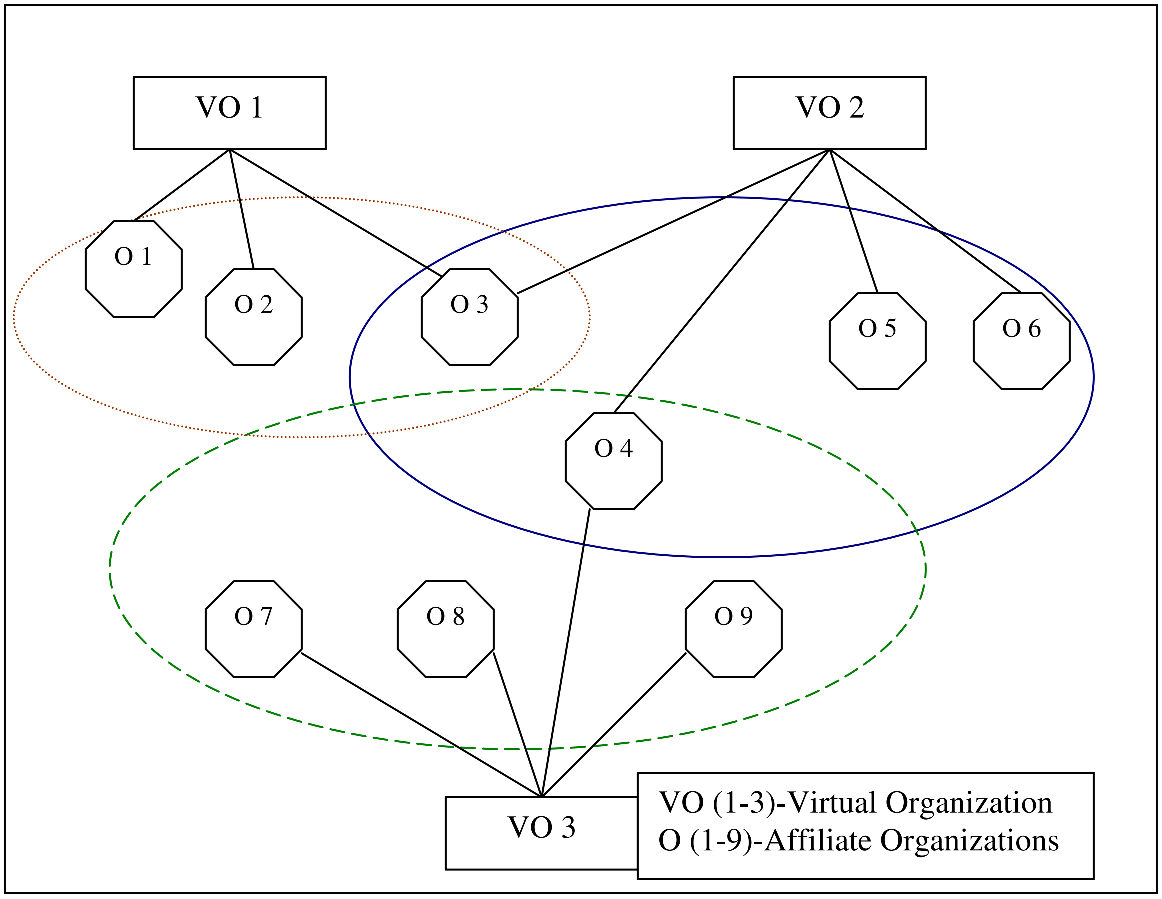

VO 1
VO 2
O 1
O 2
O 3
O 4
O 5
O 6
O 7
O 8
O 9
VO 3
VO (1-3)-Virtual Organization
O (1-9)-Affiliate Organizations

Figure 1.2: Structure of Virtual Organizations and associated Organization

Grids can functionally be classified into three types [6], Computational grids which are focused primarily on computationally intensive tasks, Data grids where the focus is on the controlled sharing and management of large amounts of data resources and Equipment grids, which have a primary piece of equipment, and where the surrounding grid is used to control the equipment remotely and aid in analyzing the data produced from the equipment [5].

The figure depicts the setup of VO’s with the affiliate organizations, There are 3 Vo’s shown and organizations 1,2,3 are associated with VO 1,also organization 3 is a part of VO 2 and we have organizations 3,4,5,6 where 4 is also a part of VO 3.

The resources in a grid are shared across various organizations by means of associating themselves with a Virtual Organization (VO). In grids, sharing of available resources is the main goal and hence there need to be rules formulated on what kind of resources need to be shared, who is allowed to share and under what conditions the sharing occurs [6]. VO takes this responsibility for coordination of these activities and makes sure certain rules are adhered to by each of the organization that is associated with the particular Virtual Organization. VO's are typically formed with respect to a particular experiment that makes use of the resources. For example, the ATLAS experiment has its own VO under the same name and all the organizations that are contributing resources towards the experiment must comply with certain rules set by the ATLAS VO. The organizations could be individuals or organizations that hold the resources necessary for sufficing the computing needs of the experiment. UTA, for example is an organization which adheres to the rules set by the ATLAS VO and is a part of the ATLAS experiment. One particular organization may be affiliated with many VO's and may participate in more than one experiment.

1.7 Types of Grids

Grids can be categorized into various types such as:

- Computational grids
- Scavenging grids
- Data Grids

Computational grids define an infrastructure which will solve complex computational problems and will be able to handle operations which are highly CPU intensive; these are often used for very large problems needing huge amounts of CPU and memory resources [6].

Scavenging grid is most commonly used with large numbers of desktop machines. It is responsible for using all the resources have signed up to be a part of the system and use their services for resource intensive tasks [6]. Typically in scavenging grids the whole system is not overloaded with intensive tasks instead the tasks are split across a huge network, and jobs given to systems are quite small that the user will not even realize they are going on in the background. A classical example could be one where you can sign up your system to be a part of a SETI (Search for Extraterrestrial Intelligence), program in which idle resources of your system are responsibly used to process information pertaining to the program.

Data grids give a common interface for all data repositories through which large amount of distributed data can be queried, managed and secured. They are often combined with computational grids. High-energy physics experiments fall under this category and will generate terabytes of data per day and around a petabyte per year [12], in such an environment, working without a common interface that can handle all this data effectively will be difficult.

1.8 Grid Monitoring Architecture

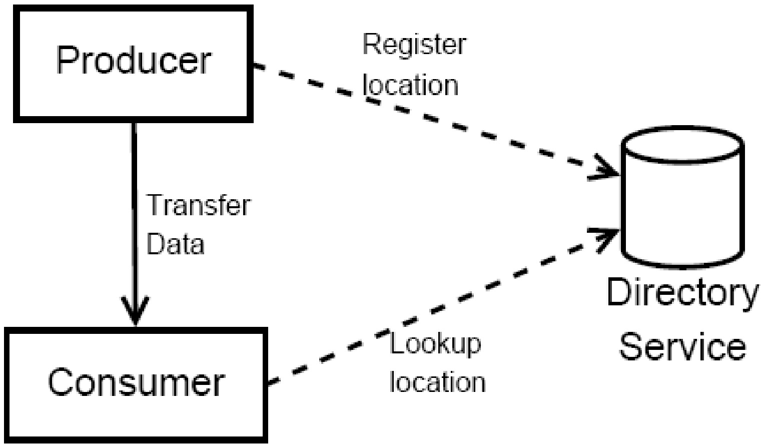


Figure 1.3: Components of Grid Monitoring Architecture (GMA) [6]

Grid monitoring architecture is a common scheme for the implementation of any grid monitoring service it is constituted by a consumer, producer and directory service. As shown in figure 3, GMA consists of three components [6]:

1. Directory service, which is like a coordinator which maintains information
2. Producer, which make monitoring information available
3. Consumer, who requests the producer for information that is of interest to them

1.8.1 Directory Service

The directory service [3] can be synonymous to registry, maintains information about where the producers and consumers are located and all information about where any specific information related to the system could be found.

At a high level the directory service will also have access rules and schemes that are concerned with producers and consumers. Transfer of information, primarily requests and responses from the producers and consumers have to go through the directory service to be initiated, however it will be direct once a connection is established using the directory service. The directory service typically provides functionalities like adding, editing and deleting entries into the directory under categories they belong.

1.8.2 Producers

A producer is any component that can send monitored data to the consumer. One producer might have multiple producer interfaces, each acting independently and linking to different kinds of measurement data sources such as hardware or software sensors, a database with historical data or other monitoring systems. [11]

A producer will have some basic responsibilities, which include:

- Updating the directory services about itself all the information available with it currently.
- Keep receiving subscriptions and queries from a consumer and keep servicing the consumers in a timely fashion.
- Termination of service between producer and consumer should be two way and can be initiated by any one, hence the producer should facilitate it.

1.8.3 Consumer

A consumer is any component that uses producers to receive monitored data .One consumer can receive data from different producers. There are different kinds of consumers; some may store the received data, some may collect monitoring data in real time the, while others collect information from different sources to make decisions.

The basic responsibilities of a consumer include,

- Updating the directory services about itself all the information available with it currently.
- Find a required producer by browsing the directory service, and initiating subscriptions with them if necessary.
- Query and store incoming data according to its needs from the producer.
- Manage subscription termination requests from producers appropriately.

1.9 Grid Monitoring Systems

Grid monitoring has been incorporated as integral part of grid workload management systems such as Condor and Globus. They are workload management middleware developed for grids [13]; they provide services such as CPU management storage management, security provisioning, data movement, monitoring [13] and usually a toolkit that also allows custom development of small applications.

Grid monitoring is also established by external monitoring systems called Nagios. Nagios is basically a Network monitoring application software which is used to monitor network hosts and services [12]; it is used sometimes with grid computing monitoring applications so as to obtain extra information about network usage which cannot be provided by the typical work load managers [12].

Grid finds its application wherever there is a potential need for large CPU and memory resources. Analysis of DNA and genes is a computer intensive task and needs more CPU power. High-energy physics is one such area where grids have been used for a long time. These experiments usually demand huge computational power and hence monitoring of the resources becomes indispensable.

Grid based monitoring provides information with respect to the jobs that are running across the various job sites or resources. This information could be simple ones ranging from the number of jobs running across the resource to others such as error codes and reasons if by any chance a job fails across the resource. Information, such as, why a job fails while running across a specific site alone could be useful, if the problem is most likely to also occur at other grid sites; counter measures could be taken. Moreover if monitoring was not established , finding out what went wrong where could just be a hassle and would actually involve manually debugging to figure out the problem also resulting in wastage of man hours in the process. For reasons cited above and many other reasons, monitoring for a distributed system like a grid becomes an important aspect in grid computing.

1.10 Grid Monitoring

1.10.1 The Need for Monitoring:

Grids are a new area in computational science and could actually create a dynamic change in the way computation is currently done. They allow on-demand access and composition of computational resources provided by multiple independent sources [13]. While providing many of advantages, the heterogeneity of the grid, the distribution of resources at various locations and the need to traverse through all these administrative domains pose new challenges which need to be addressed effectively. [11]

These technical challenges can be categorized into two separate areas that need attention they are;

- Fault diagnosis
- Failure management strategy.

When a failure occurs in a complex system such as a grid, it is quite difficult to zero down the problem to one particular component. This is typically the case in many instances. The problem could be with the user side and that it is missing some necessary configuration components or it could be due to a grid certificate, since grid certificates are a necessary to gain access to the resources hosted by the organization and they have to be obtained from the VOs. The problem could have also been due to a first time configuration problem on the resource organization’s side and might need the attention of a system administrator for it to be resolved, or it could have also occurred due to a disk crash in any of the system that the user is requesting access to. In most of the cases the error messages for these problems are quite abstruse and will require some level of understanding of the grid structure to decipher the problem effectively.

Monitoring of the grid resources will prove to be a very useful step here to avoid necessity of heavy staffing, also it is to be noted that the downtime of any grid resource is the loss to the experiment and the virtual organizations hosting it. If fault diagnosis is one motivation for establishing a good monitoring system, failure management strategy and failure mitigation the next time are other factors which further emphasize the need for a good monitoring system. Also with a good monitoring system, failures can be detected early and steps can be taken to make sure the same type of failure does not occur in the other clusters that are associated with the same organization, or even across different organizations. This kind of failure mitigations strategy will make sure the resources are put to full use and that a small glitch does not stall the whole system; resulting in reduced outputs and longer processing times.

1.11 Monitor Types

Two main types of monitoring can be identified in any monitoring system

- Infrastructure monitoring
- Application monitoring

Infrastructure monitoring aims at collecting information about grid resources; it can also maintain the history of observations in order to perform retrospective analysis. Application monitoring aims at enabling the observation of a particular execution of an application; the collected data can be useful to the monitor application activity or for visualizing its behavior of it when running in a distributed environment. [14]

As listed above at a very high level these are the two prime areas that need to be monitored in a grid. In Infrastructure monitoring the areas that need monitoring will typically be the ones like CPU consumption on each of the nodes, disk space usage monitoring, cache level alert etc. CPU consumption is the ratio of the CPU's currently in use to the CPU's available. This is to avoid overloading of a few nodes alone. Monitoring the CPU usage will give crucial information whether the load is evenly distributed or if a particular cluster is getting overloaded with numerous requests.

Disk usage monitoring will assure that we do not end up with the disks being full which will result in the system entering a state where it will hang eventually and will not be able to accept incoming requests. This might lead to a restart and will result in the loss of data which might prove quite costly especially when huge amounts of data are transmitted over the Internet.

Cache management is also an interesting area that needs monitoring when we are dealing with distributed systems. Since effective handling of data plays a major part in grid computing, it is important that we make sure that most commonly used data is readily available to the jobs coming in. We have to ensure that the response time for requested data, for a particular job coming to the grid site, is kept as low as possible so as to achieve maximum efficiency.

1.11.1 Infrastructure Monitoring

Network monitoring is probably one of the most important areas that will need efficient monitoring when we are dealing with grids. Networks are almost an integral part of a grid system, since we are dealing with distributed systems the Internet becomes an inevitable component for effective functioning of the grid.

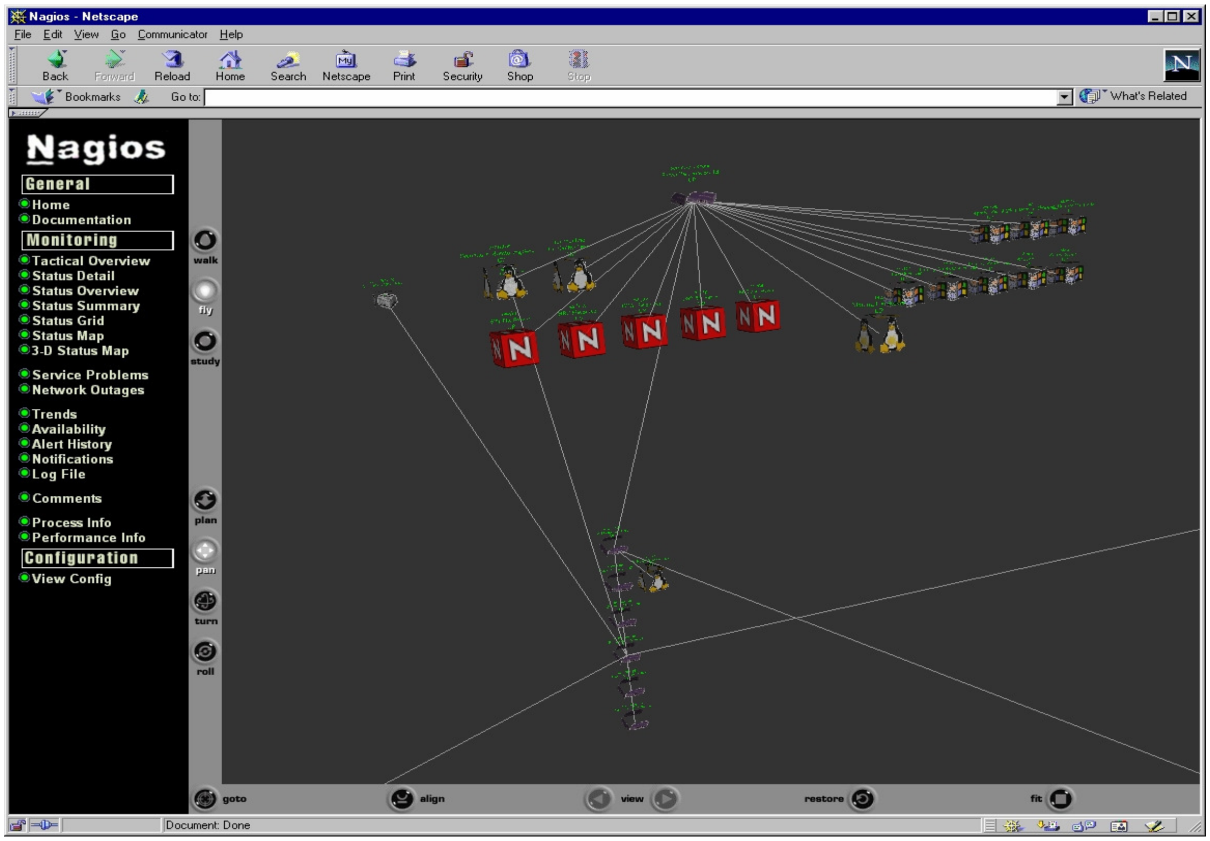


Figure1.4: Nagios screenshot showing the different Operating system hosts [12]

The monitoring of the network that is associated with the grid may be a part of the load manager system or it might sometimes be integrated with the grid middleware that is developed separately in order to suffice the monitoring features not available in the load-manager. Alternatively, it is also sometimes achieved by means of autonomous tools designed specifically to monitor networks, these systems are integrated with the grid middleware or sometimes operate totally in an individual fashion to gather the vital information to monitor the health of the network that supports the grid. One example of such system is Nagios. Nagios is an open source framework for monitoring network hosts and services with the purpose of failure detection. Nagios has a core which is responsible for most of the information processing and automatic recovery of the system, it gets most of its help from the network sensors which work in tandem with the core to process all the important pointers which tell how effective the network is doing [11].

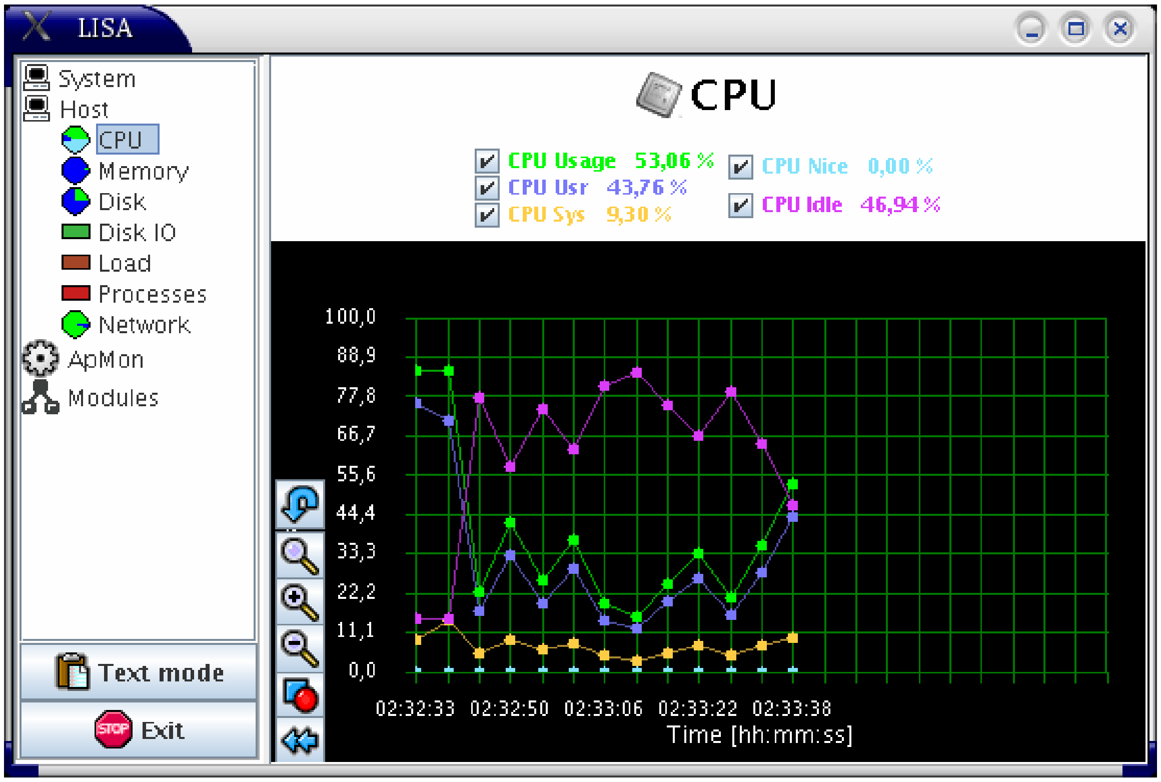
1.11.2 Application Monitoring

The next type of monitoring can be categorized as application monitoring. This is specific to the kind of experiment using the grid resource. The user will usually be interested in having a look at the individual type of jobs that were submitted by him, so the monitoring system usually has facilities for querying specifying job by means of identifiers. Application monitoring also includes monitoring performance of a particular grid site. This information comes in handy to the system administrators, it would be convenient to get a consolidated data sheet or graph indicating the daily, weekly or monthly performance of the particular grid site.

Application monitoring is hence very flexible and can be customized to any level based on the user preferences; it is usually embedded in the middleware and has to be customized by small applications which will aid in providing the information the user may be interested. This might range from simple information such as the state of the job the individual has submitted to more important information such as reasons if a job suddenly failed.

1.12 Current Monitoring Systems

Monalisa [13], - (Monitoring Agents using a Large Integrated Services Architecture), provides integrated monitoring for many of grid sites individually and collectively for separate VOs also. The framework is based on Dynamic Distributed Service Architecture and is able to provide complete monitoring, control and global optimization services for complex systems. [13]



✓ CPU Usage 53,06 %
✓ CPU Usr 43,76 %
✓ CPU Sys 9,30 %
✓ CPU Nice 0,00 %
✓ CPU Idle 46,94 %

Figure1.5: A screenshot showing the CPU related information from Monalisa

The system is designed as an ensemble of autonomous multi-threaded, self-describing agent-based subsystems which are registered as dynamic services [13], and are able to collaborate and cooperate in performing a wide range of information gathering and processing tasks. These agents can analyze and process the information in a distributed manner, to provide optimization decisions in large scale distributed applications. An agent-based architecture provides the ability to invest the system with increasing degrees of intelligence, to reduce complexity and make global systems manageable in real time. The scalability the system derives is from the use of multithreaded execution engine to host a variety of loosely coupled self-describing dynamic services or agents and the ability of each service to register itself and then to be discovered and used by any other services, or clients that require such information.[13]

The Lemon (LHC Era Monitoring) is a client server based monitoring solution for distributed systems [14]. Developed by CERN as part of the ELFms tool suite [14] (Extreme Large Fabric management system) this toolkit is now used by many grid sites in production, in CERN it is deployed at over 2500 nodes. System administrators and developers are participating in service and data challenges. Lemon works in a way, where on each monitored node an agent is running, and it launches and communicates using a push-pull protocol with sensors which are responsible for retrieving monitoring information [14].

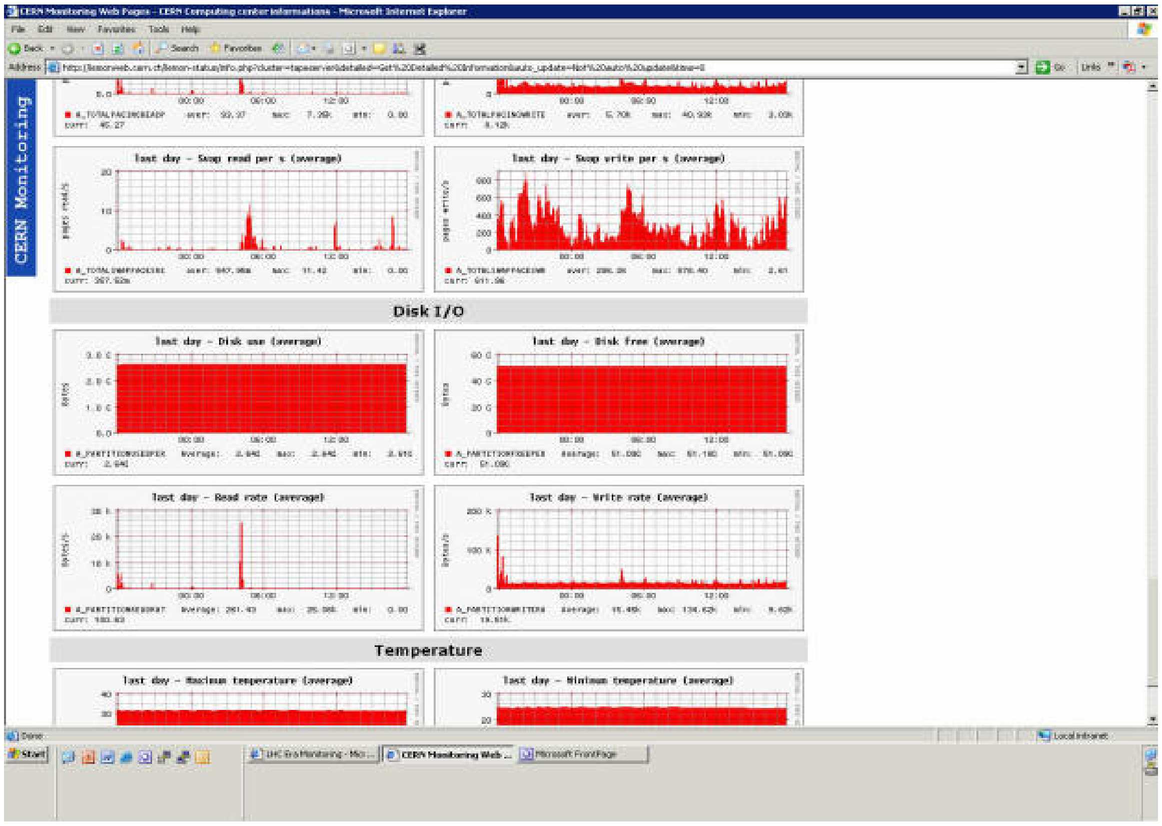

CERN Monitoring
Disk I/O
Temperature
last day - Swap read per s (average)
last day - Swap write per s (average)
last day - Disk use (average)
last day - Disk free (average)
last day - Read rate (average)
last day - Write rate (average)
last day - Maximum temperature (average)
last day - Minimum temperature (average)

Figure 1.6: A snapshot from Lemon showing disk usage and temperature [14]

The extracted samples are stored on a local cache and forwarded to a central Measurement Repository using UDP (User Datagram Protocol) or TCP (Transmission Control Protocol). The Measurement Repository can interface to a relational database or a flat-file backend for storing the received samples.

The Sam Grid [27] is the grid assisting the D0 experiment going on in Fermi Lab and it has its own monitoring system customized to provide the necessary information for its users. The monitoring system attempts at providing the user with site level monitoring, the grid-job submission level monitoring, and the progress of execution of the job at the execution site, among other important information pertinent to the entire grid. [15].

Among other nice features it has a geographical map, which serves as an anchor to the execution sites (as shown in Figure 7), there is also a hyperlink to monitor the submission sites on the grid. The monitoring system can be launched to monitor a particular site by clicking on the available hyperlink on the map. The information from the execution sites at a particular monitoring site is retrieved from the information servers deployed at the monitoring site itself.

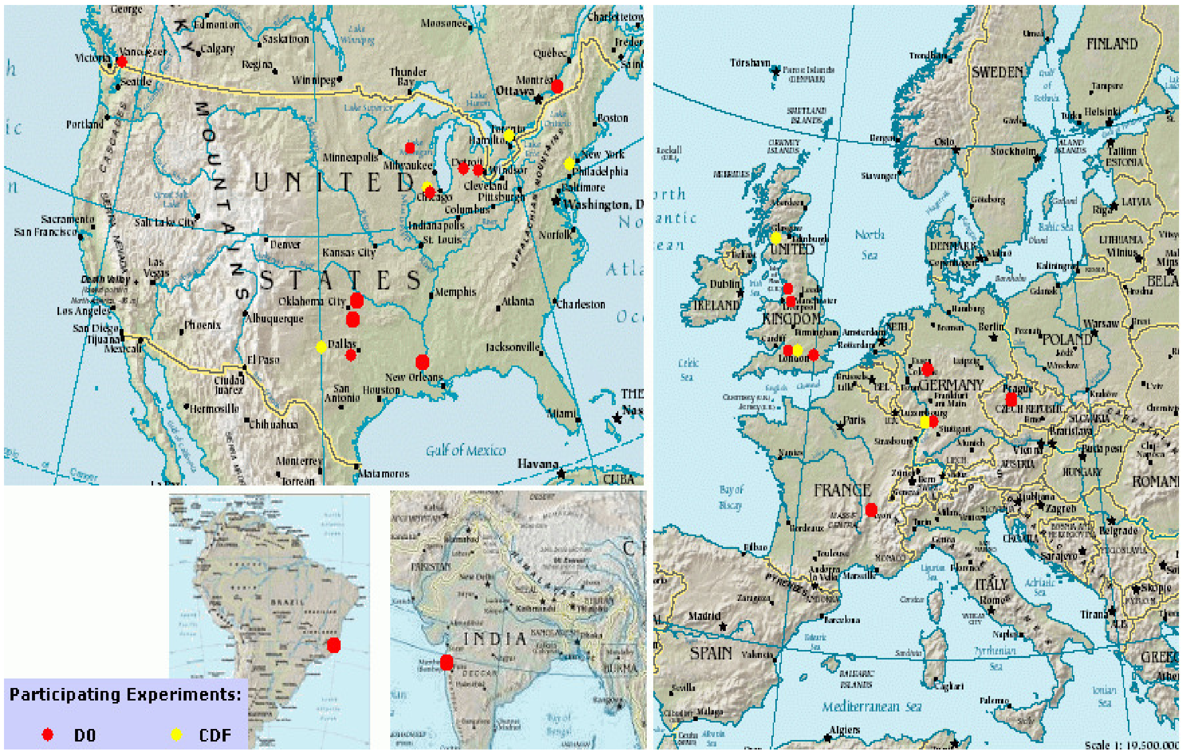


Figure 1.7: First page of SAM grid monitor - a geographical access page.

GridIce [16] is a distributed monitoring tool designed for grid systems which is being developed in the framework of the EGEE project [17] [18]. The design is based on the different abstraction level of a Grid:

- Virtual Organization level
- Grid Operation Center level
- Site Administration level
- End-User level.

The system uses LEMON to collect the host related metrics on each site and enable a publishing service next to each LEMON server. GridIce offers a standard interface to publish the monitoring data at the Grid level, different aggregations and partitions of monitoring data are provided based on the specific needs of different user's categories like VO, site. From being able to start from summary views and to drill down to details, it is possible to verify the composition of virtual pools or to sketch the sources of problems. A complete history of monitoring data is also maintained to deal with the need for retrospective analysis. [16]

R-GMA [20] is a monitoring and information management service for distributed resources based on GMA (Grid Monitoring Architecture). It has a relational model with SQL support to provide static and dynamic information about grid resources. Note that R-GMA doesn't provide a general distributed RDBMS but is relational in the sense that producers announce what they have to publish with a sql create table statement and publish with a sql insert and that consumers use a sql select to collect the information they need.

1.13 What Panda Monitoring Offers

Panda is a Grid middleware system developed to meet the data processing needs of the ATLAS experiment in the United States. ATLAS processing of data, places challenging requirements on throughput, scalability, robustness, minimal operations manpower, efficient integrated data management and processing management. The current estimate for the number of jobs is 200-300 thousand for a day in the United States alone when the experiment begins to run on a full scale. Panda is also a system that is slowly evolving into a generic high level workload manager for the OSG (Open Science Grid) experiments and can be used by other experiments running on the OSG sites in the United States.

Such important improvements to panda, place it at a very important position to have a good monitoring system so as to completely understand the effectiveness with which all the sites are running jobs. Monitoring in Panda again varies from giving simple information in the form of the number of jobs running collectively at any given instance of time across the sites, to more important attributes such as the reason why some jobs might have failed. It is also responsible to give error codes if for some reason a job has been waiting at a particular site without getting started even though there may be resources available. These are just a few high level functionalities of the Panda Monitoring system.

The current version of panda monitor is developed in Python. The whole system runs on Python, Apache and Mysql, where Mysql databases implement the job queue, all metadata and monitoring repositories. The monitoring server works with the Mysql DBs, including a logging DB populated by system components recording incidents via a simple web service behind the standard python logging module, to provide web browser based monitoring and browsing of the system and its jobs and data

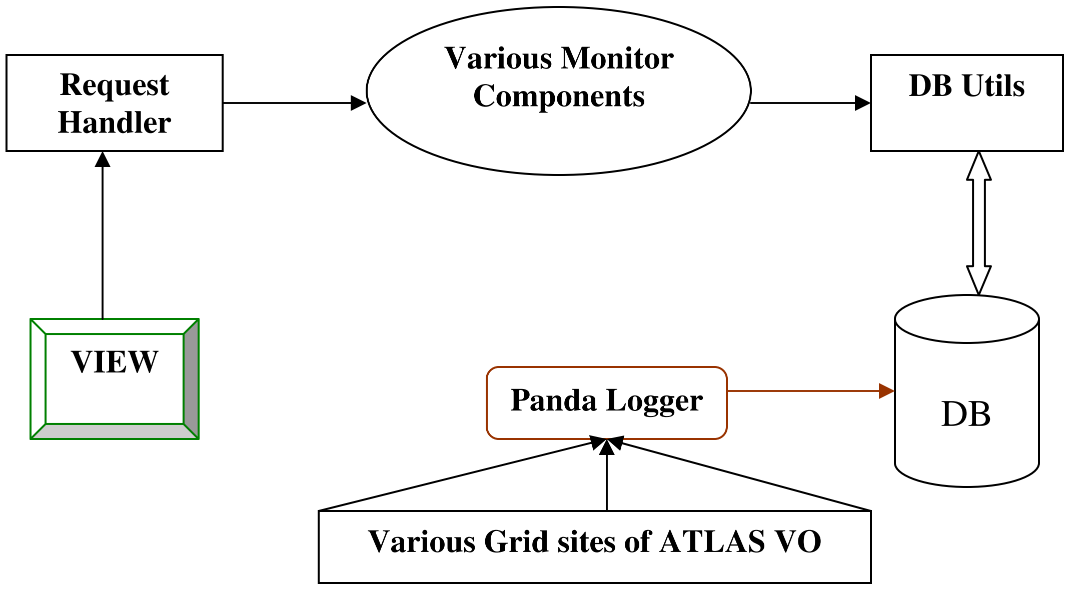
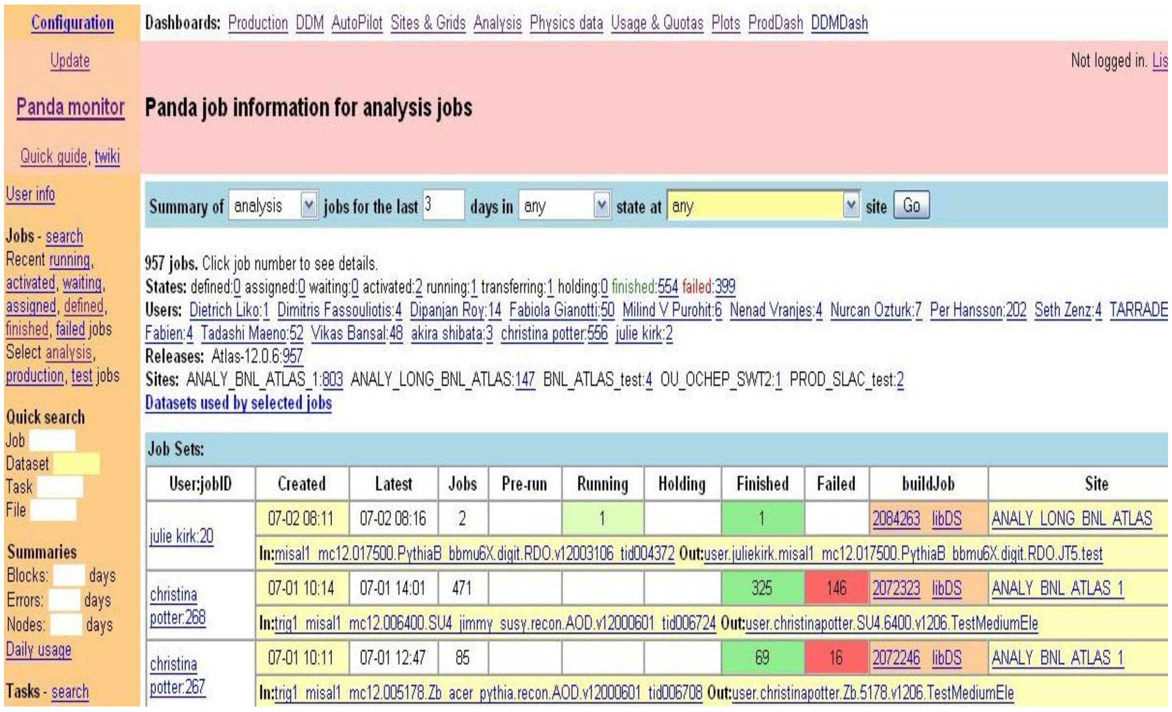


Figure 1.8: A High level Architecture of the Panda Monitor

The information collection for Panda is handled by a module called the Panda logger; it is most critical component since it records all the information to a central server which the monitor uses to process user and system queries, to present the requested information by means of a web interface. The module makes use of the standard python logging module. The logger service runs off of the same Apache server as the monitor. The logger service receives HTTP logging messages from clients and registers them in a logging DB in Mysql. The logger makes its entries into the repositories from where the monitor picks up the necessary information. The Figure 8 above gives a very high level description of the Panda monitor without going into the interaction aspects of each of the components. The components mentioned in the above diagram can be defined at a high level as the most important ones needed to get a monitor page running


Configuration
Update
Panda monitor
Quick guide, twiki
User info
Jobs - search
Recent running, activated, waiting, assigned, defined, finished, failed jobs
Select analysis, production, test jobs
Quick search
Job
Dataset
Task
File
Summaries
Blocks: days
Errors: days
Nodes: days
Daily usage
Tasks - search
Dashboards: Production DDM AutoPilot Sites & Grids Analysis Physics data Usage & Quotas Plots ProdDash DDMDash
Not logged in. Lis

Panda job information for analysis jobs

Summary of analysis **jobs for the last** 3 **days in** any **state at** any **site** Go

957 jobs. Click job number to see details.
States: defined:0 assigned:0 waiting:0 activated:2 running:1 transferring:1 holding:0 finished:554 failed:399
Users: Dietrich Liko:1 Dimitris Fassouliotis:4 Dipanjan Roy:14 Fabiola Gianotti:50 Milind V Purohit:6 Nenad Vranjes:4 Nurcan Ozturk:7 Per Hansson:202 Seth Zenz:4 TARRADE Fabien:4 Tadashi Maeno:52 Vikas Bansal:48 akira shibata:3 christina potter:556 julie kirk:2
Releases: Atlas-12.0.6:957
Sites: ANALY_BNL_ATLAS_1:803 ANALY_LONG_BNL_ATLAS:147 BNL_ATLAS_test:4 OU_OCHEP_SWT2:1 PROD_SLAC_test:2
Datasets used by selected jobs

Job Sets:

User:jobID	Created	Latest	Jobs	Pre-run	Running	Holding	Finished	Failed	buildJob	Site
julie kirk:20	07-02 08:11	07-02 08:16	2		1		1		2084263 libDS	ANALY LONG BNL ATLAS
	In:misal1 mc12.017500.PythiaB bbmu6X.digit.RDO.v12003106 tid004372 **Out:**user.juliekirk.misal1 mc12.017500.PythiaB bbmu6X.digit.RDO.JT5.test									
christina potter:268	07-01 10:14	07-01 14:01	471				325	146	2072323 libDS	ANALY BNL ATLAS 1
	In:trig1 misal1 mc12.006400.SU4 jimmy susy.recon.AOD.v12000601 tid006724 **Out:**user.christinapotter.SU4.6400.v1206.TestMediumEle									
christina potter:267	07-01 10:11	07-01 12:47	85				69	16	2072246 libDS	ANALY BNL ATLAS 1
	In:trig1 misal1 mc12.005178.Zb acer pythia.recon.AOD.v12000601 tid006708 **Out:**user.christinapotter.Zb.5178.v1206.TestMediumEle									

Figure 1.9: Snapshot of the Existing Panda Monitor for Analysis Jobs.

The current Panda monitor helps in monitoring important information such as the number of jobs running collectively under panda in a given period of time. The same can be assessed individually for every individual site; this turns out to be an important factor in determining the individual performance of every site. It is also helpful in monitoring user information in terms of quotas which provides information about the limit available for every user to submit jobs. The existing monitor is also responsible for monitoring the information about the pilots that are submitted, it gives the current state of the pilot in the system such as submitted, scheduled, running, finished, failed. Additionally it also gives error messages to understand why the pilot failed, if it went to a failed state. These error messages play a key role in determining the potential problems that might arise in other sites due to similar outages and helps in mitigating the resource downtime before if can occur.

CHAPTER 2

GOAL OF THE THESIS

2.1 Goal of the Monitoring System

It was elaborated in detail in the previous chapter about the Grid infrastructure and the various kinds of monitoring available in general and a brief introduction to grid monitoring was given.

Increasing number of components in computing environments, due to the computing needs in society and the low prices of hardware, has led to many techniques and tools being developed to help system administrators to manage their computing resources. Monitoring is one of the most important ones of those that evolved in this paradigm, and involves the use of software mainly to track computer activities and hardware sensors to certain extent to collect other data. Monitoring may include tracking of network activities and security threats, alert resource failures as well as keeping check on Internet usage, data entry, e-mail and other computer applications used from by individual users or computers. The need for a good monitoring system becomes indispensable, considering the numerous components associated with the grid.

There are a few important factors that have to be considered before looking into the design and deployment of monitoring structure, as described below

Monitored data should have a certain determined lifetime. Depending on the volume of monitored data and the space available on hard disks, it will be necessary to define how long the data will be required.

Data storage is an important area, certain parameters need to be stored directly while some others will always have to go through some processing before they can actually make sense or provide useful information to monitor. Both the cases have to be handled efficiently.

Update frequency varies depending on the data that is being recorded and it will have to be altered according to the importance of the data and how long the recorded value is legitimate. For more critical data the update frequency is maintained in small numbers so that it can be made sure that the information is in synchronization with the actual values.

Network data rate is another important aspect, in most of the monitoring systems information has to be preserved in more than one place; the availability of the World Wide Web has facilitated this easily. As this is the case at most places, a monitoring system needs to be backed up by a strong networking system, the data rate should be rapid enough to transfer the latest monitored information to the monitoring repository in a prompt fashion.

Latency can be defined as the amount of time taken for a packet to travel from source to destination. It includes transmission and processing time taken at both the source and at the destination. This is heavily dependent on the network's performance and the processing power of monitoring system. Latency which is commonly called as lag should be tried and kept at its bare minimal always because, it proves to be a huge problem when scaling any system.

Robustness is an important characteristic for any system, should a failure happen to the monitoring system, monitored data should still be able to reach the destination without trouble. Loss in information cannot be tolerated in any system, especially in a monitoring system. Usually robustness is achieved by acting in a proactive manner and making copies of the data before it is transmitted anywhere and once there is an acknowledgement for the data to have reached safely, the backed up data is processed accordingly.

Security of the monitored information is very important, hence effective storage of the data in secure locations and usage of encryption techniques while sending the data over the Internet, or even between a LAN's is an important step to ensure safe transmission.

2.2 Monitoring System Types

Two main types of monitoring can be reported they are. [21]

- Time Driven Monitoring
- Event Driven Monitoring

This technique is based on acquiring periodic status information to provide an immediate view of the behavior of the objects being monitored. This may extend over a prolonged period of time frame and could be used to study the performance of the system over a long time.

This approach is based on obtaining information about the occurrence of interesting events. This is a dynamic pattern of monitoring since only events evoking special interests to a set of people are monitored. It could be trigged by any special case occurrence or anomalies that may be stumbled upon.

2.3 Information Retrieval for Monitoring

Retrieval of information from a monitoring system can also be classified into two types. They are [22].

- Passive
- Active

Passive monitoring is like a wait and watch mode model, here any relevant information is not requested specifically but is sent in a periodical fashion by the device itself. The incoming data from the specific sensors or devices are recorded in a database and used later to be processed individually or with other data.

Active monitoring is a demand and serve model. Periodical requests are sent in the form of external signals to the monitored system, and it follows the requests and measures the requested values and reports back.

2.4 What to Monitor

There are various areas which can be monitored and a few of them can be given as [23]:

- Performance

- Fault
- Accounting
- Security

2.4.1 Performance

In the above mentioned monitoring areas, performance is an important aspect of any system. A monitoring system's performance can be measured various factors like,

- Availability
- Response time
- Throughput
- Utilization

Availability of a system could be given simply as the percent of time the system is available for a user. Availability is based on the reliability of the system. If a system consists of more than one component, then the reliability is the collective reliability of all the components. The reliability is calculated by the probability that a component will perform its specified function for a specified time under specified conditions. It can be expressed by:

$$\text{Availability} = \text{MTBF}*100/ (\text{MTBF}+\text{MTTR})$$

Where MTBF is Mean Time Between Failures and represents the component failure and MTTR for Mean Time to Repair. Hence as the number of components comprising a system grows, calculation of availability becomes relatively complex.

Response time is the time it takes for a system to respond to a given user input or query. Shorter response times are desirable in all systems and creating a monitoring system is focused towards achieving optimized response times.

Throughput can be defined as the rate at which application oriented events occur. This is an interesting performance analysis area and could even be useful to predict the demand that could be there for the system at a given time. This is done by collection of historical data which could be used to predict times when the system will be overloaded. Throughput usually helps us give an estimate demand that will exist for the system given certain conditions.

Utilization is a parameter which can give the percentage of time the system was used as against the total time it was up and running. It could also help in probability analysis to say the most likely times during which the system is to be overloaded .It is a fairly simple but effective tool which can measure network efficiency like parameters.

2.4.2. Fault

Fault monitoring is mainly done with fault mitigation in aim. Fault monitoring aims to identify faults as quickly as possible and to identify cause of such occurrences. Speedy fault detection enables remedial actions to be taken reducing the overhead caused but this has certain problems.

- Some devices do not have an effective fault detection mechanism.
- Late response from a monitored resource may mean even just network congestion but could give an impression that the device is faulty.

- Some times failure patterns that occur could not be detected with just one or a few occurrences and could lead to a situation where the problem could still not be tackled.

Failures can be anticipated by defining thresholds and by sending notification when the monitored values cross the set limits. This is a proactive way to effectively isolate and diagnose various faults

2.4.3 Accounting

The goal of accounting is to record information on resources and service usage of the system. The accounted resources are usually hardware usage, communication facilities or services. It is mainly done for policy purposes to enforce certain rules on a organization or user. This is a kind of Application monitoring that was discussed in the types of monitoring and has lower significance than an Infrastructure monitoring.

2.4.4 Security

Monitoring can be used to check the security of the system. Deploying a monitoring solution enables the identification of unofficial services or servers. It can also be a precious tool in helping to detect network security violation such as intrusions or compromised host. Detecting suspicious activity and cutting it off immediately is of prime importance since the data that we are dealing is usually of high importance and its integrity cannot be compromised.

2.5 Monitoring Vs Over-Monitoring

The maximum parameters monitored in a system, the better can be the understanding of the same. When designing a monitoring system, it is often tempting to monitor everything. This can prove to be a costly affair as too much information makes it hard to see what is important and might cripple the service. That's why it is crucial to carefully choose the services to monitor, depending on needs and goals.

Monitoring is one such area where redundant information can infiltrate easily and can actually reduce the efficiency with which a monitoring system can actually service the users. It is very easy to slip in that zone which starts using the resources available for monitoring which will otherwise be used to do process the actual jobs.

2.6 Striking the Balance

It is highly important that we clearly categorize what is crucial for the current system to be monitored, so that it results in better performance ,we have to have the users high level requirements in mind to look at what we are monitoring. Striking the perfect balance between crucial, necessary and redundant data is an important aspect of a good and efficient monitoring system.

Consider a simple case of IP packet sniffing for security purposes over a network. By default, assume all packet level transactions are recorded as a part of security measures, and the system suddenly detects a suspicious activity, a good monitoring system should be able to detect the spurious activity and be able to automatically shut down the source from accessing the system further, but how a monitoring systems achieves this depends on the architect who laid the plans for it.

A normal system would just start logging the spurious activity once it starts picking it up ,on top of the information it is already recording previously and utilize a few other resources to come to the conclusion that something is wrong. An efficiently designed system would not log the redundant information again and once it picks up a suspicious activity would probably just report the IP address from where it is picking it up to the system, which is already logging all this information. This system should be able to successfully process all this information without disturbing other resources which might be serving other requests, and must still be able to shut down the source of the suspicious activity.

Over-monitoring has various disadvantages; a few of them can be listed as wastage of resources which will otherwise be involved in processing the requests, also it results in added burden to the database of the system when excessive data is collected by overlooking. The added overhead to the network lines that carry this entire excessive payload should also be noted. These are just a few disadvantages enumerated, much trouble is caused by over monitoring and it is in the best interest of the system to plan ahead and monitor the crucial and necessary data alone, this helps in establishing a robust system.

2.7 Goals of Panda Monitor

The Panda Monitor is a grid middleware monitoring system which is a classical example of an infrastructure monitor and an application monitor which works in tandem.

The current version of Panda Monitor is developed completely in Python. The whole system runs on Python Apache and Mysql, where Mysql databases implement the job queue and all metadata and monitoring repositories. A monitoring server works with the Mysql DBs, including a logging DB populated by system components recording incidents through a simple web service, it provides web browser based monitoring and browsing of the system for its jobs and related data. Though this proves very much fine right now, it is the common opinion that it will not scale well once the original jobs start coming into the system to the tune of 300,000 jobs a day. The current version of Panda only runs 3000-4000 jobs everyday. The current monitor response times range between 23 to 84 seconds depending on the query. This lag and greater response time can be attributed to the clear non compliance with a clean Model View Controller (MVC) architecture by most of the code which is responsible for the Panda monitor; this especially proves vital when we are dealing with web based applications.

The Figure below illustrates a high level interaction scheme of the Panda monitor with all the other components of the Panda, the python logging scripts which are responsible for collecting all the information can be seen in orange.

There is a python monitor logger server which is centralized and picks up all the information about different sites from its head nodes and also interacts with the Data manager which is responsible for providing information about the status of the respective DQ2 datasets which are specific to each site. A more elaborate description and the individual working components of the monitor will be discussed in detail in the sections to come.

The current Panda monitors important information such as the number of jobs running collectively under Panda in a given period of time, the same can be assessed individually for every individual site, this turns out to be an important factor in determining the individual performance.

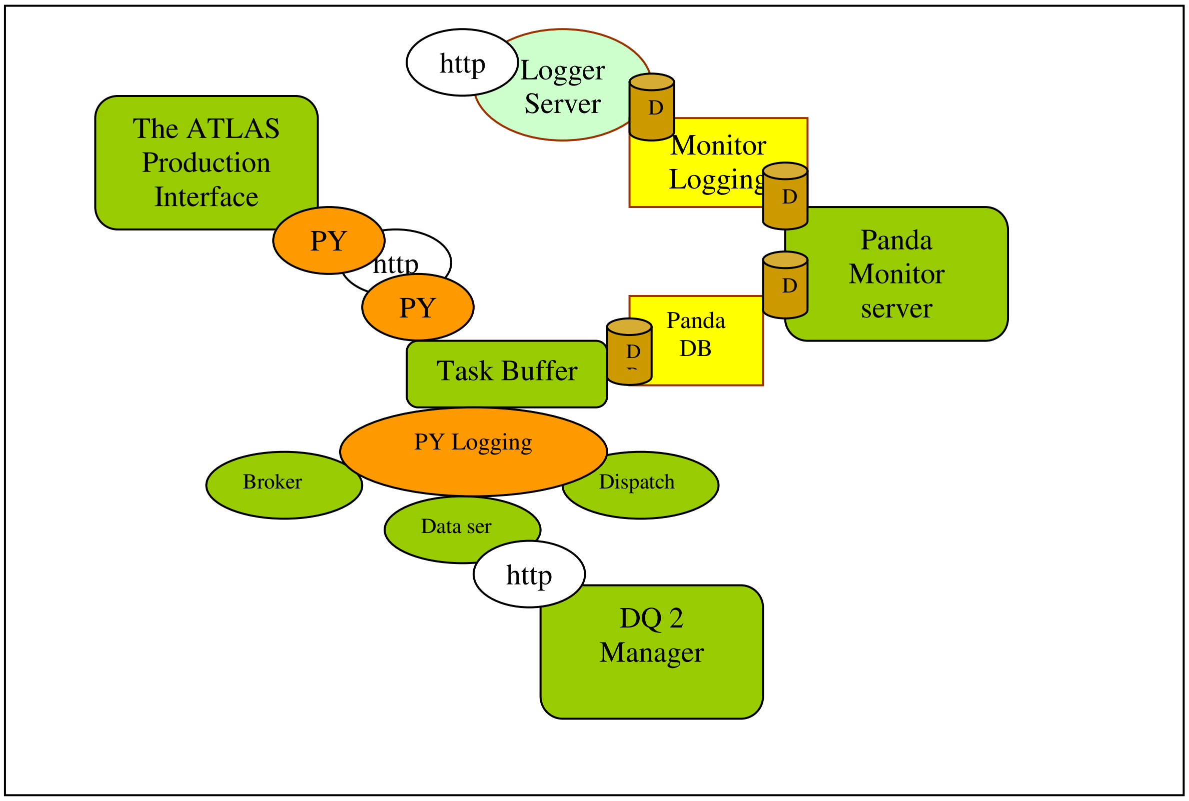


Figure 2.1: Illustration of the Panda Monitor interaction at a very high level

It is also helpful in monitoring user information in terms of quotas which provides information about the limit available for every user to submit jobs. The existing monitor is also responsible for monitoring the information about the pilots that are submitted, it gives the current state of the pilot in the system such as submitted, scheduled, running, finished and failed. Additionally it also gives error messages to understand why the pilot failed, if incase it went to a failed state. These error messages play a key role in determining the potential problems that might arise in other sites due to similar outages and help in mitigating the resource downtime before if can occur.

CHAPTER 3

THE PANDA ARCHITECTURE

3.1 Introduction

In ATLAS experiment where we deal with heavy nuclei collision and proton-proton collision, data collected from LHC could reach several petabytes [28] a year from the detector into the production system. To deal with huge amount of data, we need collaboration of clusters of computers from many different places. Panda (Production and Distributed Analysis System) as a grid middleware is designed to effectively handle the data from the ATLAS production system and send them to the computer cluster that has enough resource, right capability to process and also to facilitate data analysis.

Panda is an effort by the US ATLAS to meet the requirements of the ATLAS experiment for full scale production and distributed analysis processing. Current estimates of the number of jobs that could come in every day into the USA are placed at 200,000 – 300,000 [28] and the actual number could be more as the experiment begins. This huge amount of jobs requires a system that is scalable, robust and has efficient integrated data/processing management. Panda was built with all these requirements in mind.

3.2 Atlas Production System

In the ATLAS production system, Panda functions as a regional executor for the OSG sites, interacting with an ATLAS production system supervisor like the Ewoyn, to receive and report production work. Panda also operates as an efficient executor system to serve both production and analysis workloads.

The Architecture of the ATLAS Prodsys can be given by four major components [28]:

- Supervisor: The supervisor is the first level where the interaction occurs with the collection of the data, an example of a supervisor is Eowyn (second generation)
- ProdDB*:* Production data base is the one where the raw data from the colliders are collected.
- DDM: Distributed data management system DQ2
- Executors: Capone can be cited as an example for Executors. Panda is one which is moving in as a generic executor in the OSG sites.

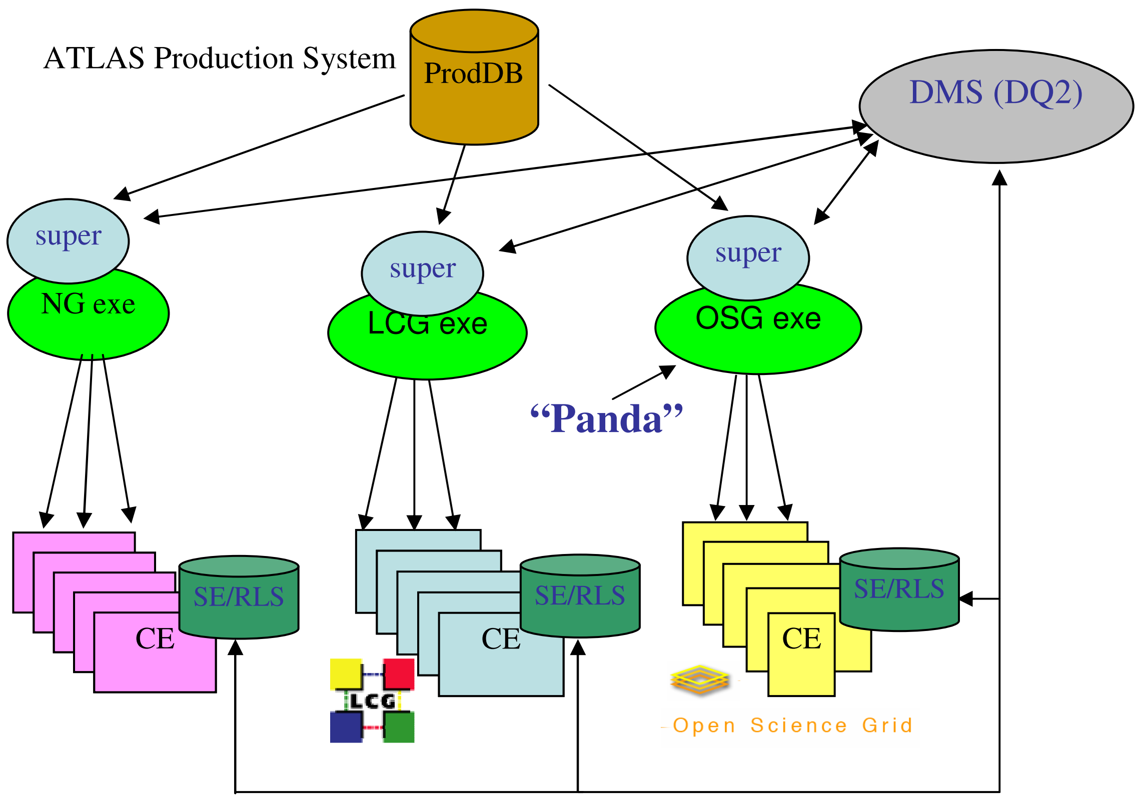


Figure 3.1: A figure showing the complete ATLAS Production System [12]

Jobs are submitted to Panda through a simple python client interface by which users define job sets, their associated datasets and the input/output files. Job specifications are transmitted to the Panda server through HTTP, with submission information returned to the client. This client interface has been used to implement Panda front ends for ATLAS production (Python Executor Interface) distributed analysis (Pathena) [28] and US regional production.

3.3 Panda Architecture

Panda has a number of components the important ones are

- Panda Server
- Panda Job Scheduler
- Panda Pilot
- Distributed Data Management

Panda was designed with the aim to support all job sources like ATLAS production, regional, group and user production. It facilitates interactive and distributed analysis. It is a system that is tightly integrated with the ATLAS DDM (DQ2).The scheme for data management is clearly a data driven and dataset based workflow. The data is pre-staged at the grid site before the job is dispatched; this is one of the most important characteristics of Panda which makes it reliable. It can be said with certainty that the data will be available to the site before any processing will even begin and it can be assured that the resources at the other end will not sit idle waiting for data to arrive.

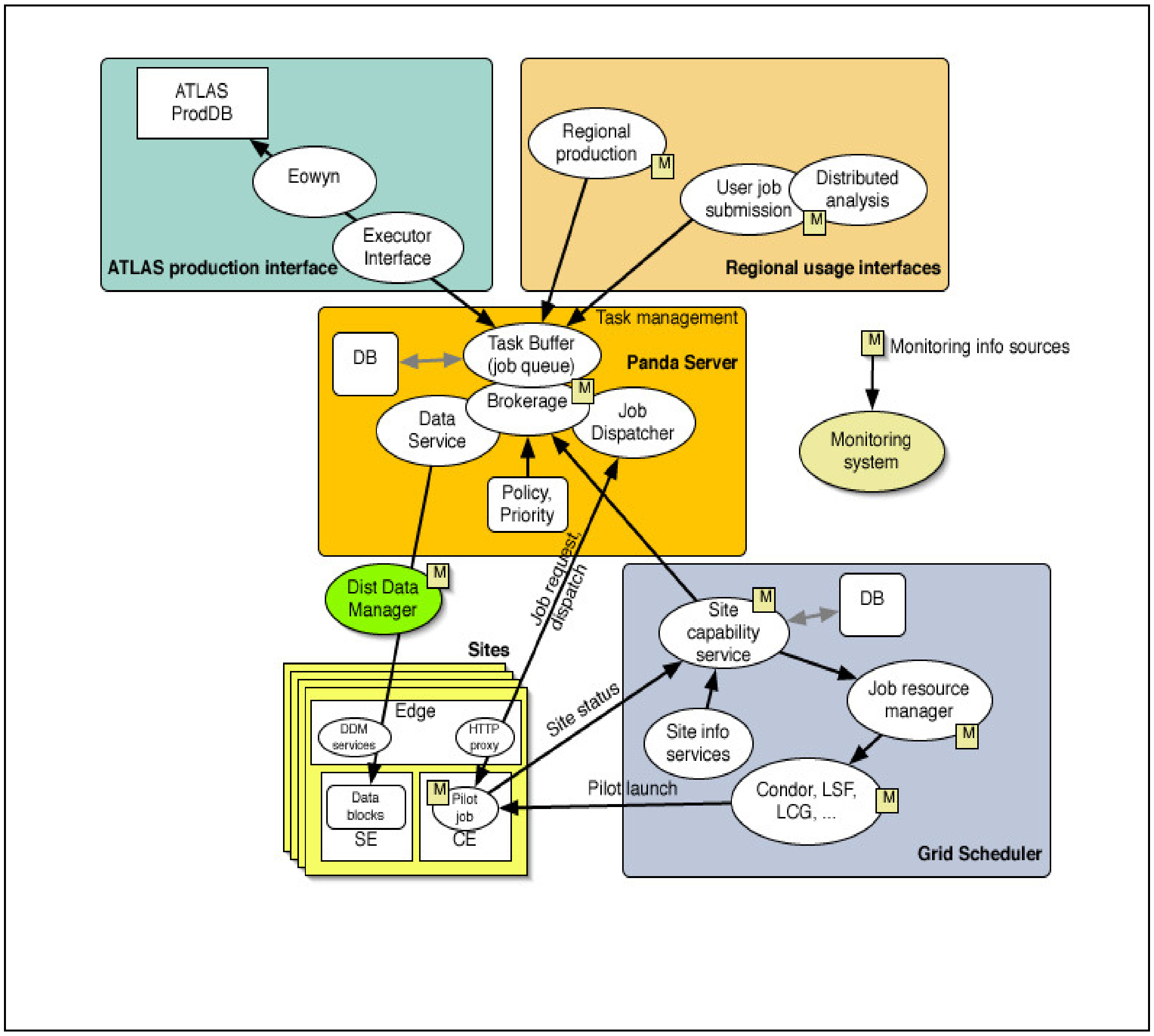


Figure 3.2: Figure giving the over all Panda Architecture and its components [28]

Job scheduling and assignments are taken care of internally within Panda itself and hence it does not need any external help in the form of middleware to achieve job scheduling. The same is the case with job dispatching, which makes the design more congruent where similar tasks grouped together before dispatching thereby lowering the burden of grid.

3.4 A Brief Overview

The working of the Panda system can be understood better by having a brief overview in the hierarchy of the systems that are involved in the ATLAS experiment.

The main center of the system is classified as Tier0 [7] CERN (Center of European Union Nuclear Research). It is responsible for archiving and distribution of raw data from event filter (EF) [29]. This is the first level of prompt reconstruction of calibration.

The next level in the hierarchy are called the Tier 1 sites .There are many organizations in Tier 1 and Brookhaven National Labs is one among them .BNL hosts and provides long-term access and archiving to a subset of the raw data, and reprocessing of raw data.

The next level in the hierarchy is the Tier 2 sites which provide calibration constant, simulation and analysis. These also play an important part since they have more information about DDM data and can prove to be vital while we are trying to access information about the whereabouts of a particular datasets. Sites which are relatively big in size are the ones which fall under this category. For example, UTA is a Tier 2 site and is responsible for job management in its zone and also will house a major portion of the datasets.

The last level in the structure is smaller organizations which have a few grids established and contribute to the processing of data. There are many such organizations throughout the United States in the form of universities or independent organizations which interact with their Zonal Tier 2 sites to obtain jobs in order to process the data at their end.

DDM provides services for data cataloging and data transfers between Atlas sites. Datasets comes from [32]

- RAW data flowing into T0
- Managed production of ESD, and AOD
- Simulation production all yield datasets

The supervisor has a number of responsibilities; some of them include translating job descriptions held in DB into appropriate scripts or commands. This is in the form of XML since transactions are through a HTTPS by using XML. The supervisor is also responsible for submitting jobs, validating a job when it finishes and resubmitting job in case of failure. The supervisor is also responsible for updating the production database to make sure all logging of the available datasets are the current information.

The Supervisor sends a request for the number of jobs wanted to the executor (for example Panda) and then it responds with a default number say 1000 jobs .Since Panda is a data driven model, and it follows a execution pattern where data is already deposited before the job arrives, the supervisor gets data from production DB (ATLAS production system) and then sends it to executor in the form of XML code. Then the executor parses the XML, puts the job description to a proper data structure and then sends it to task buffer

The task buffer after obtaining the job transfers control to the Brokerage unit from where the brokerage unit makes a request to Task buffer to send jobs over and groups according to certain preferences like locality of distribution of datasets, this is where the DDM or data services of Panda gains importance. They collectively group datasets in a pattern where they are closely associated and they also choose sites to this group of jobs based on the DDM data blocks. Then it transfers control to its own Data service to organize a file transfer through DDM. The brokerage unit is also responsible for updating the Task buffer about transfers that just happened.

The Panda server by itself has other important components they can be given as

- Panda Task buffer
- Panda Brokerage
- Panda Job Dispatcher
- Panda Data Service

The Panda server can be described as the central Panda hub composed of several components that make up the core of Panda; it is implemented as a stateless REST (Representational State Transfer) web service.

The Panda server receives work from these front ends into a global job queue, upon which a brokerage module operates to prioritize and assign work on the basis of job type, priority, input dataset and its locality, and available CPU resources. Allocation of job blocks to sites is followed by the dispatch of input data to those sites, handled by a data service interacting with the ATLAS distributed data management system [32]. Data pre-placement is a strict precondition for job execution; jobs are not released for processing until the data arrives at the processing site. When data dispatch completes, jobs are made available to a job dispatcher.

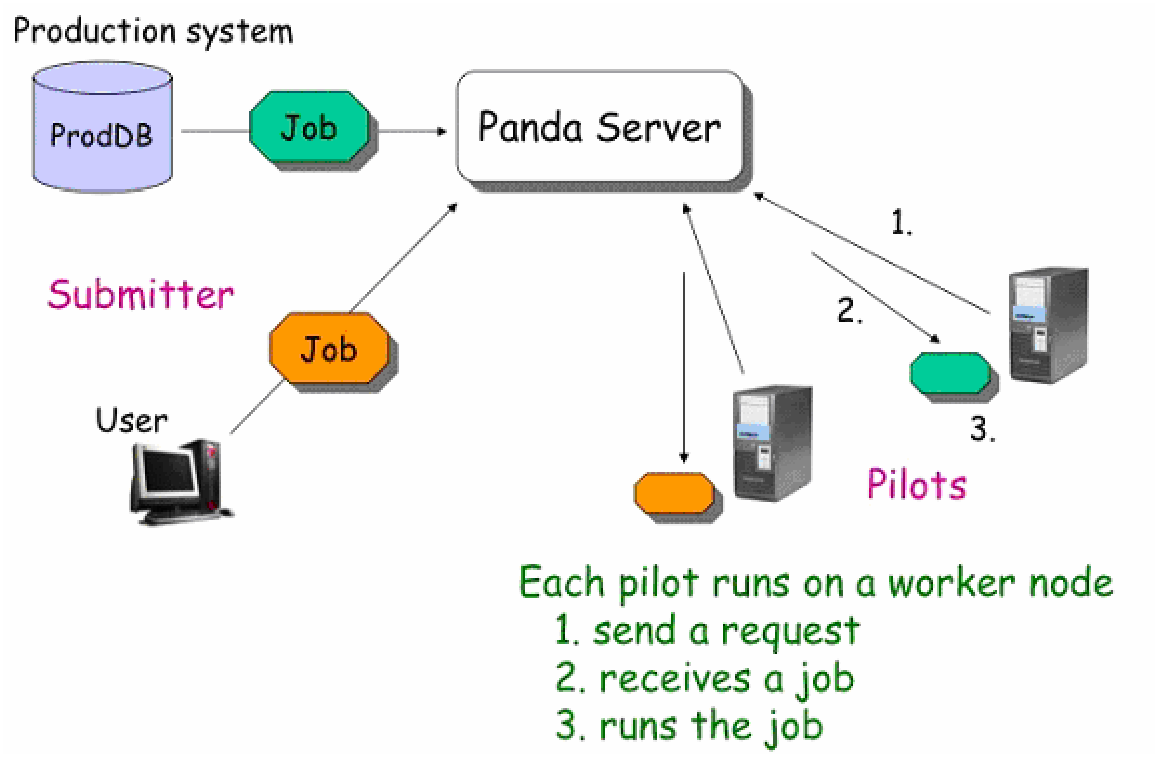


Figure 3.3: A high level working view of the Panda Server [32]

An independent subsystem manages the delivery of pilot jobs to worker nodes through a number of scheduling systems. A pilot once launched on a worker node contacts the dispatcher and receives an available job appropriate to the site. If no appropriate job is available, the pilot may immediately exit or may pause and ask again later, depending on its configuration. Minimal latency from job submission to launch is important, is that the pilot dispatch mechanism bypasses any latency in the scheduling system for submitting and launching the pilot itself.

The Data service is one of the most important components in Panda in the execution sequence. It is responsible for requesting and selecting groups of jobs from task buffer. It also takes in account lists of available sites before doing such an assignment of jobs and grouping them. The data service also requests the list of available sites from the brokerage and is also responsible for requests DDM to reserve and move blocks of data these are the input files which come from the Brokerage and Tier 1 sites. The data services are also responsible for checking the status of the group of jobs in task buffer and it also receives notification from DDM on completion of transfer to trigger downstream actions. The data services also requests DDM to move and archive output files.

The dispatcher is responsible for making sure the specific requirement for the jobs are directed to the right site where there are enough resources available. Moreover it should also be made sure that the load balancing is achieved in a proper manner so that simple jobs are not directed to sites with excess resources. This might result in other bigger jobs being blocked.

Firstly the dispatcher requests the task buffer for the highest priority job which meets site requirements and whose input files had already been pre-staged [32] **.**The availability of data at the site which is being explored for job execution is a major requirement in Panda and hence it has to be met before jobs can be sent to any specific sites.

This is the pattern in which execution happens in Panda, finally to have an overview jobs flow through executor interface, task buffer and brokerage in a continuous manner.

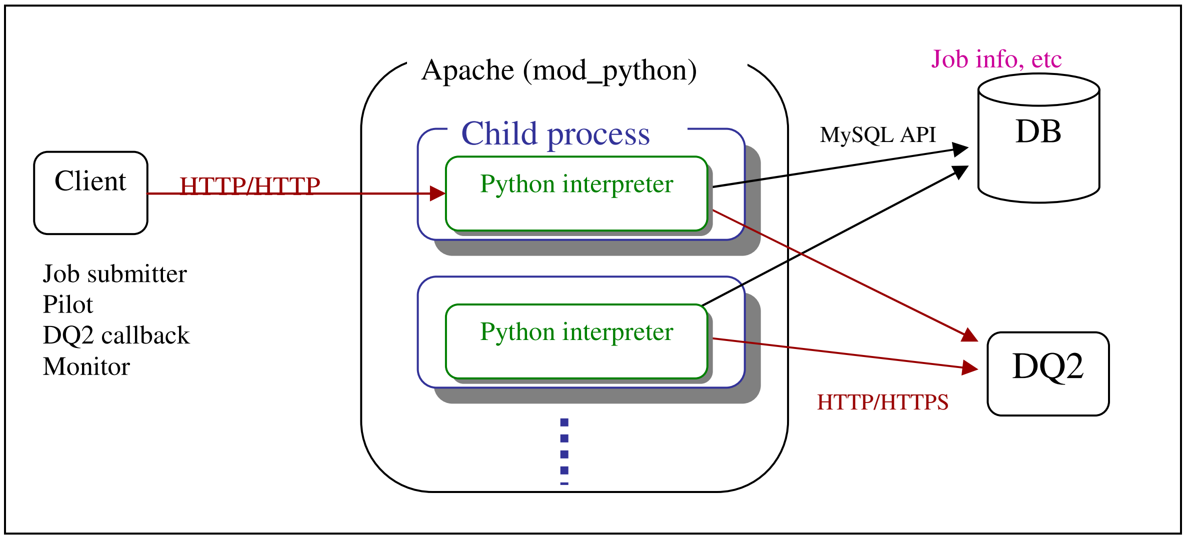


Figure 3.4: The Panda Server Interaction with DQ2

So while new jobs keep coming into task buffer, some of them are already being processed by the brokerage and ready for taking input files and those corresponding jobs in. Task Buffer ready for input files are maintained in such a manner that they are of higher priority than the other jobs being processed by the Dispatcher. The next step is to create executable job wrapper send it across to the specific site that was designated to it based on availability of resources, it then monitors progress of jobs and updates the Task Buffer [31].

The executor then sends a query to the Task Buffer to find out the status of the job. This is the final stage where the status of the job is recorded if it went through or it is in the waiting state.

The Panda brokerage is also a part of the server and it works closely with all the other components of the server like the task buffer and job dispatcher to make sure jobs are handled properly. The main work of the brokerage is to manage where jobs and associated data are sent based on job characteristics, data locality, priorities, user/group role, and site. It does the important job of keeping track of which job goes where and is also responsible for making some important decisions as to where the job has to go based on information of resources available in each site. The brokerage makes the decisions based on the resources and capacities matched to job needs, and dynamic site information. Also the proximity of the datasets to the site if some of them are not available already is a deciding factor for all the decision.

3.5 DDM-Distributed Data Management:

DDM is an important component which is responsible for moving the data around for the effective working of Panda; the main aim of the DDM is to provide a service for data cataloguing and data transfer between Atlas sites .The new DDM software is called DQ2 (Don Quixote 2) [29].At a very high level DDM can be given by two major components

- Catalogue Services which can be further divided into
    - Catalogue client
    - Catalogue server
- Site Services

The Catalogue services are responsible for tracking data movement across the grid sites. Large amount of files are grouped into dataset based on attributes (e.g. physics characteristics, chronological productions and so on). Data are only moved or replicated in units of data blocks (immutable datasets). Also, the cataloguing services allow dataset based lookup and it is achieved in a manner where it is reliable without complex mechanisms to maintain global consistency of the data .A separate catalog is used to map dataset into its constituent files.

The site services are responsible for moving data from site to site .The decisions on how to move datasets between sites is achieved by interacting with the global catalogue and the local dataset to facilitate the moving of data across. A very brief overview of the working of DQ2 is provided below to have an understanding of the component.

The DQ2 system can be given as two major components at a very high level as shown in the architecture diagram they are the [32]:

- Global catalog
- Local catalog

The global catalog has partial repository of all the objects in the local catalogs

The important components of the DQ2 system are:

- Content Catalog: This component is responsible for mapping each dataset to its constituent files
- Dataset Repository: This holds all dataset names and unique IDs representing them along with the system metadata.
- Subscription Catalog: This stores subscriptions of datasets to sites

- Location Catalog: This is one of the major components which have the actual physical location of the datasets.

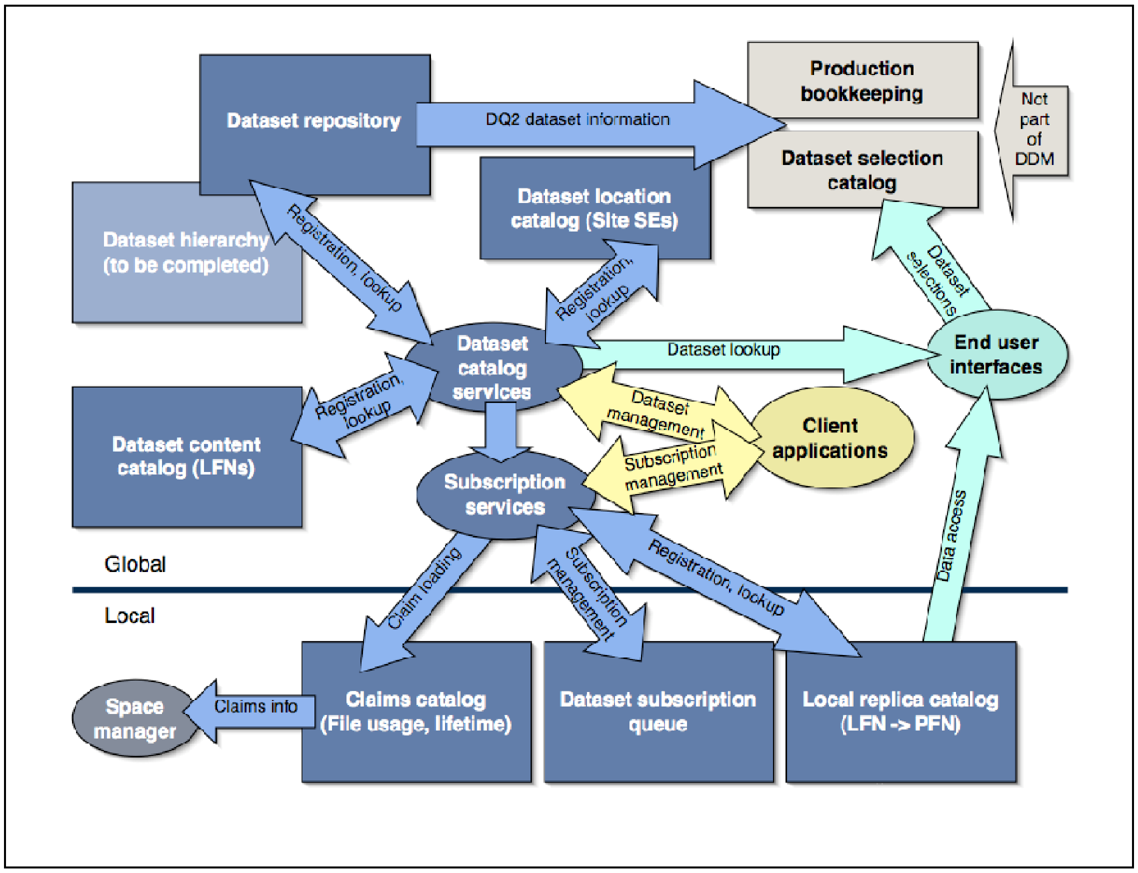


Figure 3.5: Architecture of the DDM services (DQ2) [32]

The Dataset repository is a catalog of datasets; it also serves as a principle catalog and look-up source for datasets but this is not the place where users perform queries to retrieve datasets. The Dataset-selection-catalogue provides detailed information by adding descriptive, query able content (physics metadata: physics attributes e.g. luminosity) [32].

The next step in the process is looking into a dataset select catalogue which is followed by dataset content catalogue where the actual mapping from a dataset to its constituent logical files happens. This process retrieves metadata which are nothing but the dataset Id and any other names given to it, then it looks into the content files with the metadata obtained in the previous stage and figures out which site the files are physically present .It is always possible that some of the files are not always located at one particular grid site and are actually distributed. It looks up the location catalog to locate these files on the other sites and pulls them up together. The location catalogue provides look-up of the sites where copies of those data blocks can be found.

Data block is the unit of data replication and transfer in DDM. Those logical representations of files and grouping of files can be thought of as points or indices to the actual physical storage of those files. DDM's in higher tier has all the logical pointers or indices to those physical files stored in the DDM's of the lower tiers.

The site services provide logical files, physical data files which are the actual contents. Client subscription specifies which data to be transferred to which specific local sites. DQ2 local site service finds subscriptions and pulls the requested data to local site so that client subscribes can trigger dataset replication to local site.

Datasets are grouped together with rules defined by their content, hierarchy (from general to specific), location and other parameters these rules are pre-defined through data coming from production jobs. The datasets will go from top tiers to lower tiers necessary and will be stored in their DDM systems as required; later these datasets will be categorized and organized by those respective DDM systems through catalog.

Dataset repository provides a nice data presentation through Data selection catalog so that end user can choose desired datasets more easily, an interface to facilitate data selections

Subscription services in Panda is an important activity that makes sure the datasets that are much in requirement are replicated properly and put up at various sites where they will be used widely for jobs. In this process a T1 facility like BNL subscribes to a group of datasets, and then T0 locates those datasets, make a replica of the dataset and sends it to T1.

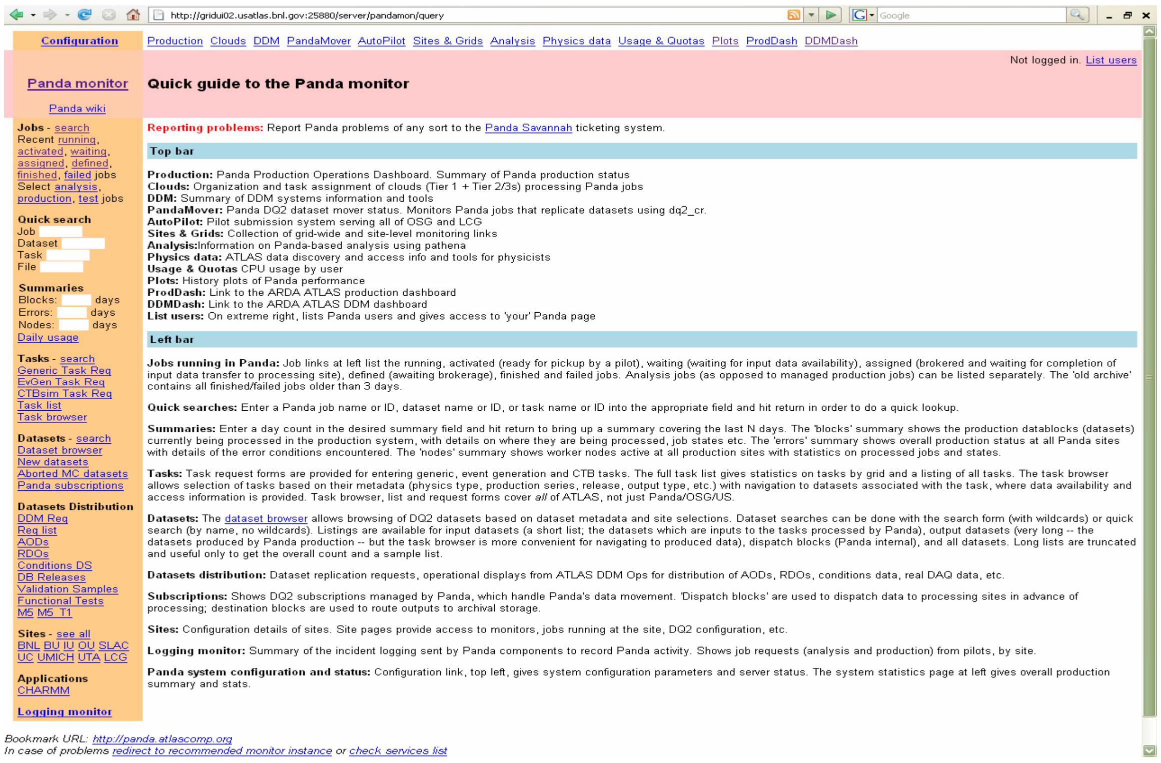
Data subscription is associated with production jobs that have all the rules for grouping datasets. The user then finds what they want and send subscriptions for the specific data. Catalog service then checks content, hierarchy and location catalog then gets back to end user with datasets available for use with proper dataset.

CHAPTER 4

PANDA MONITOR FEATURES

4.1 Panda Main Page

The figure below is that of the introductory page to the Panda monitor. It gives a brief description of the various features that are available in the panda monitor and the various kinds of information you can obtain related to ATLAS experiment from it. We will explore some of the features that Panda monitor offers in detail.


http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query

Configuration

Production Clouds DDM PandaMover AutoPilot Sites & Grids Analysis Physics data Usage & Quotas Plots ProdDash DDMDash

Not logged in. List users

Panda monitor

Panda wiki

Quick guide to the Panda monitor

Jobs - search
Recent running, activated, waiting, assigned, defined, finished, failed jobs
Select analysis, production, test jobs

Quick search
Job
Dataset
Task
File

Summaries
Blocks: days
Errors: days
Nodes: days
Daily usage

Tasks - search
Generic Task Req
EvGen Task Req
CTBsim Task Req
Task list
Task browser

Datasets - search
Dataset browser
New datasets
Aborted MC datasets
Panda subscriptions

Datasets Distribution
DDM Req
Req list
AODs
RDOs
Conditions DS
DB Releases
Validation Samples
Functional Tests
M5 M5_T1

Sites - see all
BNL BU IU OU SLAC
UC UMICH UTA LCG

Applications
CHARMM

Logging monitor

Reporting problems: Report Panda problems of any sort to the Panda Savannah ticketing system.

Top bar

Production: Panda Production Operations Dashboard. Summary of Panda production status
Clouds: Organization and task assignment of clouds (Tier 1 + Tier 2/3s) processing Panda jobs
DDM: Summary of DDM systems information and tools
PandaMover: Panda DQ2 dataset mover status. Monitors Panda jobs that replicate datasets using dq2_cr.
AutoPilot: Pilot submission system serving all of OSG and LCG
Sites & Grids: Collection of grid-wide and site-level monitoring links
Analysis: Information on Panda-based analysis using pathena
Physics data: ATLAS data discovery and access info and tools for physicists
Usage & Quotas CPU usage by user
Plots: History plots of Panda performance
ProdDash: Link to the ARDA ATLAS production dashboard
DDMDash: Link to the ARDA ATLAS DDM dashboard
List users: On extreme right, lists Panda users and gives access to 'your' Panda page

Left bar

Jobs running in Panda: Job links at left list the running, activated (ready for pickup by a pilot), waiting (waiting for input data availability), assigned (brokered and waiting for completion of input data transfer to processing site), defined (awaiting brokerage), finished and failed jobs. Analysis jobs (as opposed to managed production jobs) can be listed separately. The 'old archive' contains all finished/failed jobs older than 3 days.

Quick searches: Enter a Panda job name or ID, dataset name or ID, or task name or ID into the appropriate field and hit return in order to do a quick lookup.

Summaries: Enter a day count in the desired summary field and hit return to bring up a summary covering the last N days. The 'blocks' summary shows the production datablocks (datasets) currently being processed in the production system, with details on where they are being processed, job states etc. The 'errors' summary shows overall production status at all Panda sites with details of the error conditions encountered. The 'nodes' summary shows worker nodes active at all production sites with statistics on processed jobs and states.

Tasks: Task request forms are provided for entering generic, event generation and CTB tasks. The full task list gives statistics on tasks by grid and a listing of all tasks. The task browser allows selection of tasks based on their metadata (physics type, production series, release, output type, etc.) with navigation to datasets associated with the task, where data availability and access information is provided. Task browser, list and request forms cover *all* of ATLAS, not just Panda/OSG/US.

Datasets: The dataset browser allows browsing of DQ2 datasets based on dataset metadata and site selections. Dataset searches can be done with the search form (with wildcards) or quick search (by name, no wildcards). Listings are available for input datasets (a short list; the datasets which are inputs to the tasks processed by Panda), output datasets (very long -- the datasets produced by Panda production -- but the task browser is more convenient for navigating to produced data), dispatch blocks (Panda internal), and all datasets. Long lists are truncated and useful only to get the overall count and a sample list.

Datasets distribution: Dataset replication requests, operational displays from ATLAS DDM Ops for distribution of AODs, RDOs, conditions data, real DAQ data, etc.

Subscriptions: Shows DQ2 subscriptions managed by Panda, which handle Panda's data movement. 'Dispatch blocks' are used to dispatch data to processing sites in advance of processing; destination blocks are used to route outputs to archival storage.

Sites: Configuration details of sites. Site pages provide access to monitors, jobs running at the site, DQ2 configuration, etc.

Logging monitor: Summary of the incident logging sent by Panda components to record Panda activity. Shows job requests (analysis and production) from pilots, by site.

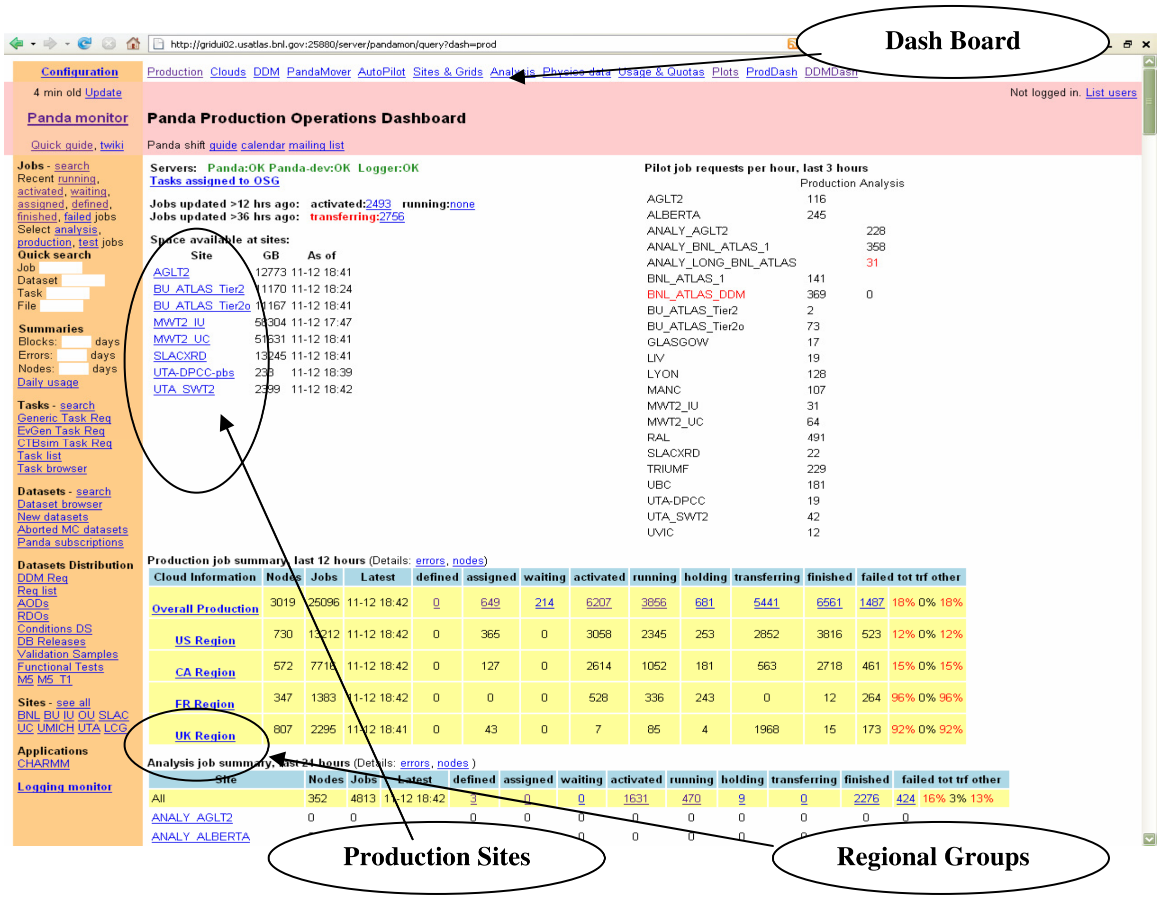
Panda system configuration and status: Configuration link, top left, gives system configuration parameters and server status. The system statistics page at left gives overall production summary and stats.

Bookmark URL: http://panda.atlascomp.org
In case of problems redirect to recommended monitor instance or check services list

Figure 4.1: Startup page for Panda Monitor

4.2 Production and Analysis page

Panda monitor extensively allows the monitoring of production and analysis jobs. The top part of the monitor which is called the dashboard has quick links to the most frequently browsed information on the panda monitor. It has various links which we will explore in detail, the first of them being production and analysis. A click on production redirects to a collective information page on the production sites in United States.



http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query?dash=prod

Configuration

Production Clouds DDM PandaMover AutoPilot Sites & Grids Analysis Physics data Usage & Quotas Plots ProdDash DDMDash

4 min old Update

Not logged in. List users

Panda monitor

Panda Production Operations Dashboard

Quick guide, twiki

Panda shift guide calendar mailing list

Jobs - search
Recent running, activated, waiting, assigned, defined, finished, failed jobs
Select analysis, production, test jobs
Quick search
Job
Dataset
Task
File

Summaries
Blocks: days
Errors: days
Nodes: days
Daily usage

Tasks - search
Generic Task Req
EvGen Task Req
CTBsim Task Req
Task list
Task browser

Datasets - search
Dataset browser
New datasets
Aborted MC datasets
Panda subscriptions

Datasets Distribution
DDM Req
Req list
AODs
RDOs
Conditions DS
DB Releases
Validation Samples
Functional Tests
M5 M5_T1

Sites - see all
BNL BU IU OU SLAC UC UMICH UTA LCG

Applications
CHARMM

Logging monitor

Servers: Panda:OK Panda-dev:OK Logger:OK
Tasks assigned to OSG

Jobs updated >12 hrs ago: activated:2493 running:none
Jobs updated >36 hrs ago: transferring:2756

Space available at sites:

Site	GB	As of
AGLT2	12773	11-12 18:41
BU_ATLAS_Tier2	11170	11-12 18:24
BU_ATLAS_Tier2o	11167	11-12 18:41
MWT2_IU	58304	11-12 17:47
MWT2_UC	51631	11-12 18:41
SLACXRD	13245	11-12 18:41
UTA-DPCC-pbs	238	11-12 18:39
UTA_SWT2	2399	11-12 18:42

Pilot job requests per hour, last 3 hours

	Production	Analysis
AGLT2	116	
ALBERTA	245	
ANALY_AGLT2		228
ANALY_BNL_ATLAS_1		358
ANALY_LONG_BNL_ATLAS		31
BNL_ATLAS_1	141	
BNL_ATLAS_DDM	369	0
BU_ATLAS_Tier2	2	
BU_ATLAS_Tier2o	73	
GLASGOW	17	
LIV	19	
LYON	128	
MANC	107	
MWT2_IU	31	
MWT2_UC	64	
RAL	491	
SLACXRD	22	
TRIUMF	229	
UBC	181	
UTA-DPCC	19	
UTA_SWT2	42	
UVIC	12	

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

Cloud Information	Nodes	Jobs	Latest	defined	assigned	waiting	activated	running	holding	transferring	finished	failed tot trf other	
Overall Production	3019	25096	11-12 18:42	0	649	214	6207	3856	681	5441	6561	1487	18% 0% 18%
US Region	730	13212	11-12 18:42	0	365	0	3058	2345	253	2852	3816	523	12% 0% 12%
CA Region	572	7716	11-12 18:42	0	127	0	2614	1052	181	563	2718	461	15% 0% 15%
FR Region	347	1383	11-12 18:42	0	0	0	528	336	243	0	12	264	96% 0% 96%
UK Region	807	2295	11-12 18:41	0	43	0	7	85	4	1968	15	173	92% 0% 92%

Analysis job summary, last 24 hours (Details: errors, nodes)

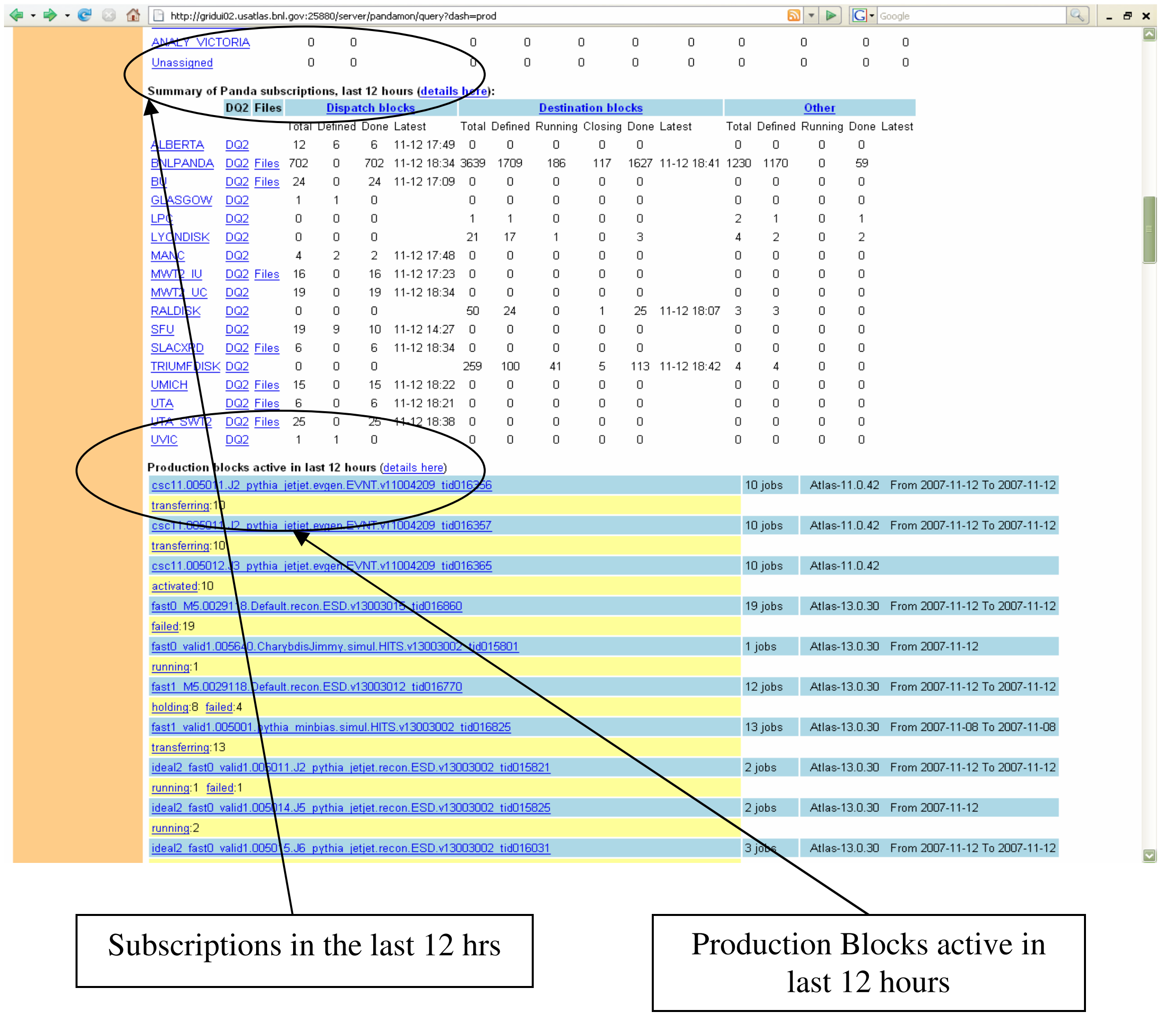
Site	Nodes	Jobs	Latest	defined	assigned	waiting	activated	running	holding	transferring	finished	failed tot trf other	
All	352	4813	11-12 18:42	3	0	0	1631	470	9	0	2276	424	16% 3% 13%
ANALY_AGLT2	0	0		0	0	0	0	0	0	0	0	0	
ANALY_ALBERTA						0	0	0	0	0			

Figure 4.2: Panda production operations page

The production page allows you to have a look at all production sites at a glance and it also allows the production sites to be seen region wise for example the production sites in the United States alone or in Canada alone could be retrieved separately.

It also gives information about the pilot job requests in the last 3 hours throughout the productions sites in United States. It also gives you more specific information in numbers about the status of the jobs running at each site like running, pending, failed and so on. This helps in collectively seeing if all sites are doing good or if at some place there are failures they can be immediately inspected.

The production job page also provides information about the subscriptions for panda in the last 3 hours and the production blocks of datasets that have been active for the past 12 hours, both these features are illustrated in the image above. Clicking on the respective links will give more information about the dataset in the form of Panda Id which has been assigned to it the time it was created and other information pertinent to it.



http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query?dash=prod

ANALY_VICTORIA	0	0	0	0	0	0	0	0	0	0	0	
Unassigned	0	0	0	0	0	0	0	0	0	0	0	

Summary of Panda subscriptions, last 12 hours (details here):

	DQ2	Files	Dispatch blocks				Destination blocks						Other				
			Total	Defined	Done	Latest	Total	Defined	Running	Closing	Done	Latest	Total	Defined	Running	Done	Latest
ALBERTA	DQ2		12	6	6	11-12 17:49	0	0	0	0	0		0	0	0	0	
BNLPANDA	DQ2	Files	702	0	702	11-12 18:34	3639	1709	186	117	1627	11-12 18:41	1230	1170	0	59	
BU	DQ2	Files	24	0	24	11-12 17:09	0	0	0	0	0		0	0	0	0	
GLASGOW	DQ2		1	1	0		0	0	0	0	0		0	0	0	0	
LPC	DQ2		0	0	0		1	1	0	0	0		2	1	0	1	
LYONDISK	DQ2		0	0	0		21	17	1	0	3		4	2	0	2	
MANC	DQ2		4	2	2	11-12 17:48	0	0	0	0	0		0	0	0	0	
MWT2_IU	DQ2	Files	16	0	16	11-12 17:23	0	0	0	0	0		0	0	0	0	
MWT2_UC	DQ2		19	0	19	11-12 18:34	0	0	0	0	0		0	0	0	0	
RALDISK	DQ2		0	0	0		50	24	0	1	25	11-12 18:07	3	3	0	0	
SFU	DQ2		19	9	10	11-12 14:27	0	0	0	0	0		0	0	0	0	
SLACXRD	DQ2	Files	6	0	6	11-12 18:34	0	0	0	0	0		0	0	0	0	
TRIUMFDISK	DQ2		0	0	0		259	100	41	5	113	11-12 18:42	4	4	0	0	
UMICH	DQ2	Files	15	0	15	11-12 18:22	0	0	0	0	0		0	0	0	0	
UTA	DQ2	Files	6	0	6	11-12 18:21	0	0	0	0	0		0	0	0	0	
UTA_SWT2	DQ2	Files	25	0	25	11-12 18:38	0	0	0	0	0		0	0	0	0	
UVIC	DQ2		1	1	0		0	0	0	0	0		0	0	0	0	

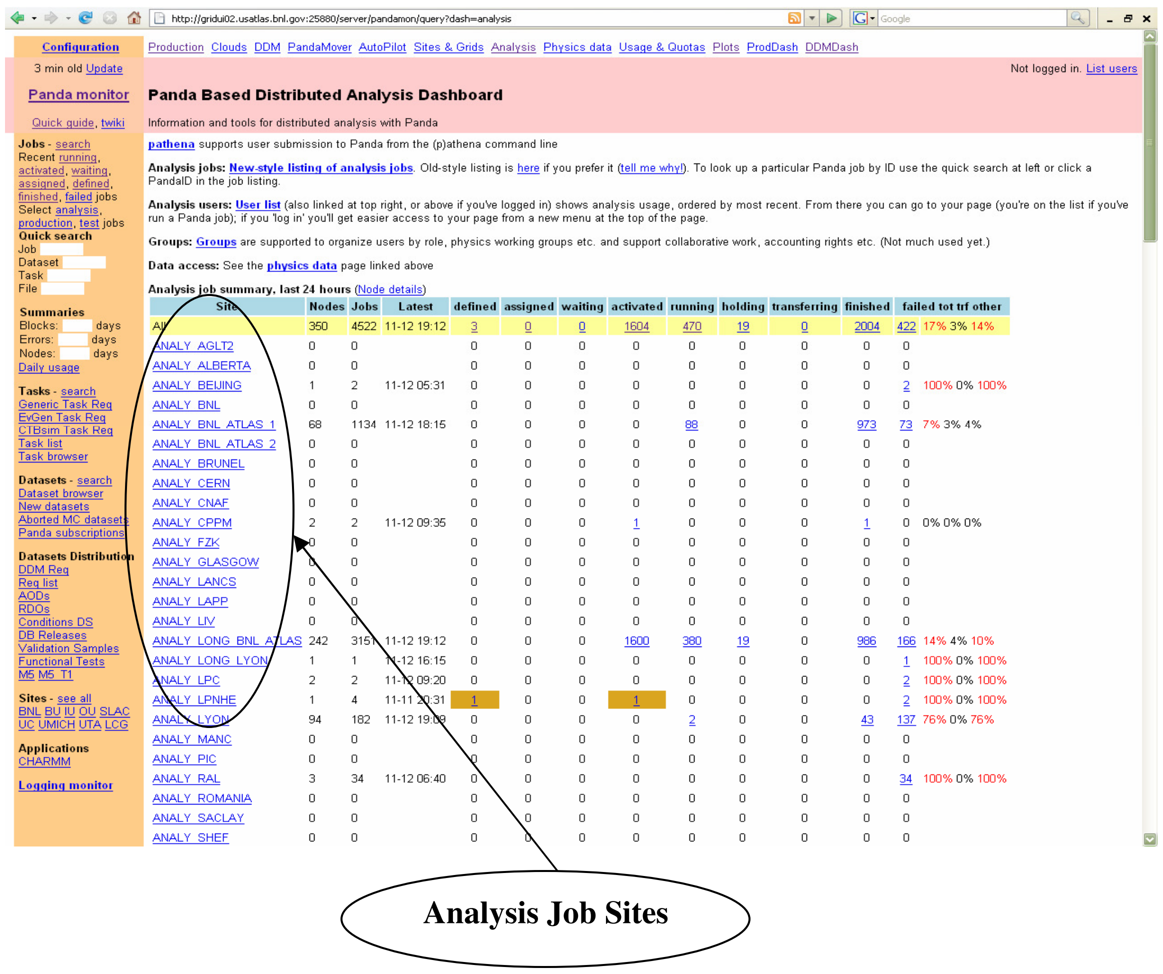
Production blocks active in last 12 hours (details here)

csc11.005011.J2_pythia_jetjet.evgen.EVNT.v11004209_tid016356	10 jobs	Atlas-11.0.42 From 2007-11-12 To 2007-11-12
transferring:10		
csc11.005011.J2_pythia_jetjet.evgen.EVNT.v11004209_tid016357	10 jobs	Atlas-11.0.42 From 2007-11-12 To 2007-11-12
transferring:10		
csc11.005012.J3_pythia_jetjet.evgen.EVNT.v11004209_tid016365	10 jobs	Atlas-11.0.42
activated:10		
fast0_M5.0029118.Default.recon.ESD.v13003015_tid016860	19 jobs	Atlas-13.0.30 From 2007-11-12 To 2007-11-12
failed:19		
fast0_valid1.005640.CharybdisJimmy.simul.HITS.v13003002_tid015801	1 jobs	Atlas-13.0.30 From 2007-11-12
running:1		
fast1_M5.0029118.Default.recon.ESD.v13003012_tid016770	12 jobs	Atlas-13.0.30 From 2007-11-12 To 2007-11-12
holding:8 failed:4		
fast1_valid1.005001.pythia_minbias.simul.HITS.v13003002_tid016825	13 jobs	Atlas-13.0.30 From 2007-11-08 To 2007-11-08
transferring:13		
ideal2_fast0_valid1.005011.J2_pythia_jetjet.recon.ESD.v13003002_tid015821	2 jobs	Atlas-13.0.30 From 2007-11-12 To 2007-11-12
running:1 failed:1		
ideal2_fast0_valid1.005014.J5_pythia_jetjet.recon.ESD.v13003002_tid015825	2 jobs	Atlas-13.0.30 From 2007-11-12
running:2		
ideal2_fast0_valid1.005015.J6_pythia_jetjet.recon.ESD.v13003002_tid016031	3 jobs	Atlas-13.0.30 From 2007-11-12 To 2007-11-12

Subscriptions in the last 12 hrs
Production Blocks active in last 12 hours

Figure 4.3: Production Page showing subscription and active blocks

The analysis job page is equally important as that of the production page and it is specifically the page where jobs submitted by users are put. This helps them in easily traversing to the specific jobs that they submitted and having a look at its status. The figure below gives a snapshot of the analysis page of the panda monitors and gives a preview of the options available to obtain relevant information.



http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query?dash=analysis

Configuration

Production Clouds DDM PandaMover AutoPilot Sites & Grids Analysis Physics data Usage & Quotas Plots ProdDash DDMDash

3 min old Update

Not logged in. List users

Panda monitor

Panda Based Distributed Analysis Dashboard

Quick guide, twiki

Information and tools for distributed analysis with Panda

Jobs - search
Recent running, activated, waiting, assigned, defined, finished, failed jobs
Select analysis, production, test jobs
Quick search
Job
Dataset
Task
File

Summaries
Blocks: days
Errors: days
Nodes: days
Daily usage

Tasks - search
Generic Task Req
EvGen Task Req
CTBsim Task Req
Task list
Task browser

Datasets - search
Dataset browser
New datasets
Aborted MC datasets
Panda subscriptions

Datasets Distribution
DDM Req
Req list
AODs
RDOs
Conditions DS
DB Releases
Validation Samples
Functional Tests
M5 M5_T1

Sites - see all
BNL BU IU OU SLAC UC UMICH UTA LCG

Applications
CHARMM

Logging monitor

pathena supports user submission to Panda from the (p)athena command line

Analysis jobs: New-style listing of analysis jobs. Old-style listing is here if you prefer it (tell me why!). To look up a particular Panda job by ID use the quick search at left or click a PandaID in the job listing.

Analysis users: User list (also linked at top right, or above if you've logged in) shows analysis usage, ordered by most recent. From there you can go to your page (you're on the list if you've run a Panda job); if you 'log in' you'll get easier access to your page from a new menu at the top of the page.

Groups: Groups are supported to organize users by role, physics working groups etc. and support collaborative work, accounting rights etc. (Not much used yet.)

Data access: See the **physics data** page linked above

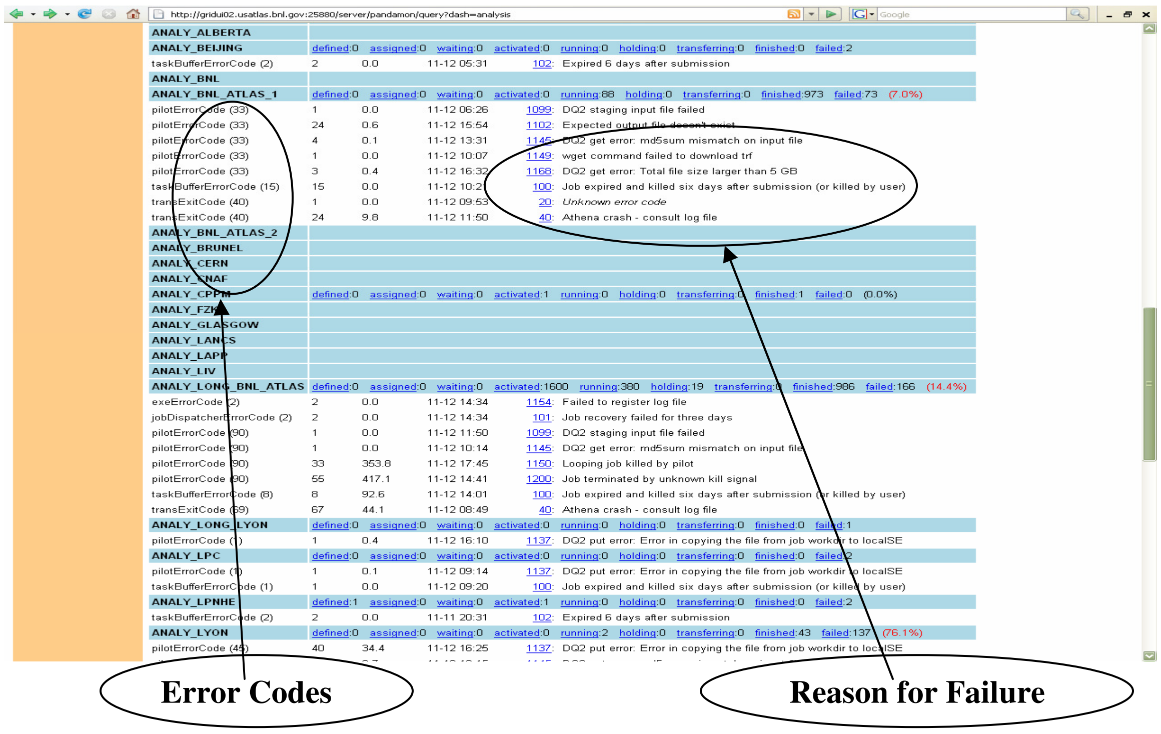
Analysis job summary, last 24 hours (Node details)

Site	Nodes	Jobs	Latest	defined	assigned	waiting	activated	running	holding	transferring	finished	failed	tot	trf	other
All	350	4522	11-12 19:12	3	0	0	1604	470	19	0	2004	422	17%	3%	14%
ANALY_AGLT2	0	0		0	0	0	0	0	0	0	0	0			
ANALY_ALBERTA	0	0		0	0	0	0	0	0	0	0	0			
ANALY_BEIJING	1	2	11-12 05:31	0	0	0	0	0	0	0	0	2	100%	0%	100%
ANALY_BNL	0	0		0	0	0	0	0	0	0	0	0			
ANALY_BNL_ATLAS_1	68	1134	11-12 18:15	0	0	0	0	88	0	0	973	73	7%	3%	4%
ANALY_BNL_ATLAS_2	0	0		0	0	0	0	0	0	0	0	0			
ANALY_BRUNEL	0	0		0	0	0	0	0	0	0	0	0			
ANALY_CERN	0	0		0	0	0	0	0	0	0	0	0			
ANALY_CNAF	0	0		0	0	0	0	0	0	0	0	0			
ANALY_CPPM	2	2	11-12 09:35	0	0	0	1	0	0	0	1	0	0%	0%	0%
ANALY_FZK	0	0		0	0	0	0	0	0	0	0	0			
ANALY_GLASGOW	0	0		0	0	0	0	0	0	0	0	0			
ANALY_LANCS	0	0		0	0	0	0	0	0	0	0	0			
ANALY_LAPP	0	0		0	0	0	0	0	0	0	0	0			
ANALY_LIV	0	0		0	0	0	0	0	0	0	0	0			
ANALY_LONG_BNL_ATLAS	242	3151	11-12 19:12	0	0	0	1600	380	19	0	986	166	14%	4%	10%
ANALY_LONG_LYON	1	1	11-12 16:15	0	0	0	0	0	0	0	0	1	100%	0%	100%
ANALY_LPC	2	2	11-12 09:20	0	0	0	0	0	0	0	0	2	100%	0%	100%
ANALY_LPNHE	1	4	11-11 20:31	1	0	0	1	0	0	0	0	2	100%	0%	100%
ANALY_LYON	94	182	11-12 19:09	0	0	0	0	2	0	0	43	137	76%	0%	76%
ANALY_MANC	0	0		0	0	0	0	0	0	0	0	0			
ANALY_PIC	0	0		0	0	0	0	0	0	0	0	0			
ANALY_RAL	3	34	11-12 06:40	0	0	0	0	0	0	0	0	34	100%	0%	100%
ANALY_ROMANIA	0	0		0	0	0	0	0	0	0	0	0			
ANALY_SACLAY	0	0		0	0	0	0	0	0	0	0	0			
ANALY_SHEF	0	0		0	0	0	0	0	0	0	0	0			

Figure 4.4: Analysis Jobs page of the Panda monitor

This page allows a user to specifically traverse to an analysis site where the user might be interested to see the performance of even a particular node of an organization that he might be affiliated to .For example if a person from UTA wants to see the number of jobs running at a particular node for a given analysis job he could find that information in this page. It gives the finished or running jobs on a particular node and if further information is needed to see the name of the jobs that can also be obtained by just clicking on the number and it would give the jobs running at that particular node at UTA.

Furthermore the actual error codes for the jobs that failed at a particular site can also be obtained at this page. For example, the snapshot below reveals the failed job codes in the site ANALY_BNL_ATLAS_1. This is made even mode transparent in the recent version of the monitor by giving the explanation for the failure codes just by their side. This greatly helps in site administrators and people submitting jobs to inspect it if failure becomes redundant due to a same error and corrective action can be taken immediately.

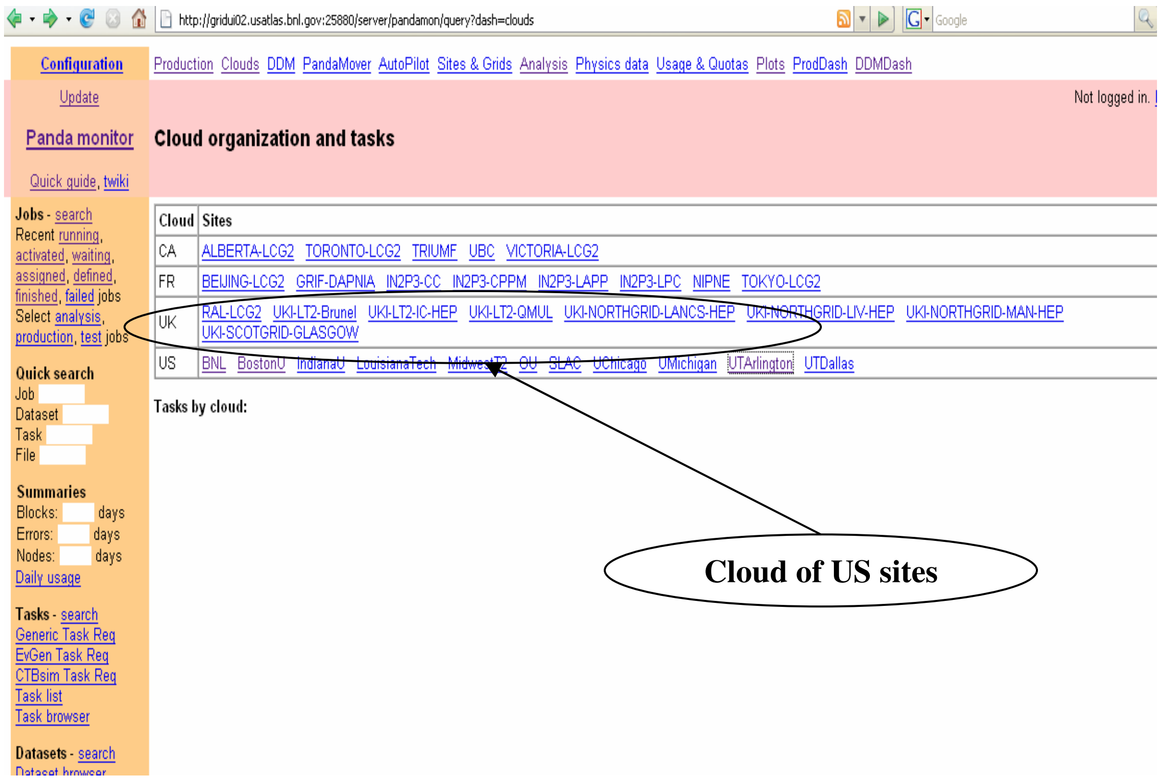

http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query?dash=analysis
Google
Error Codes
Reason for Failure

```
ANALY_ALBERTA
ANALY_BEIJING           defined:0  assigned:0  waiting:0  activated:0  running:0  holding:0  transferring:0  finished:0  failed:2
taskBufferErrorCode (2)    2    0.0    11-12 05:31    102:  Expired 6 days after submission
ANALY_BNL
ANALY_BNL_ATLAS_1       defined:0  assigned:0  waiting:0  activated:0  running:88  holding:0  transferring:0  finished:973  failed:73  (7.0%)
pilotErrorCode (33)        1    0.0    11-12 06:26   1099:  DQ2 staging input file failed
pilotErrorCode (33)       24    0.6    11-12 15:54   1102:  Expected output file doesn't exist
pilotErrorCode (33)        4    0.1    11-12 13:31   1145:  DQ2 get error: md5sum mismatch on input file
pilotErrorCode (33)        1    0.0    11-12 10:07   1149:  wget command failed to download trf
pilotErrorCode (33)        3    0.4    11-12 16:32   1168:  DQ2 get error: Total file size larger than 5 GB
taskBufferErrorCode (15)  15    0.0    11-12 10:2     100:  Job expired and killed six days after submission (or killed by user)
transExitCode (40)         1    0.0    11-12 09:53     20:  Unknown error code
transExitCode (40)        24    9.8    11-12 11:50     40:  Athena crash - consult log file
ANALY_BNL_ATLAS_2
ANALY_BRUNEL
ANALY_CERN
ANALY_CNAF
ANALY_CPPM              defined:0  assigned:0  waiting:0  activated:1  running:0  holding:0  transferring:0  finished:1  failed:0  (0.0%)
ANALY_FZK
ANALY_GLASGOW
ANALY_LANCS
ANALY_LAPP
ANALY_LIV
ANALY_LONG_BNL_ATLAS    defined:0  assigned:0  waiting:0  activated:1600  running:380  holding:19  transferring:0  finished:986  failed:166  (14.4%)
exeErrorCode (2)           2    0.0    11-12 14:34   1154:  Failed to register log file
jobDispatcherErrorCode (2) 2    0.0    11-12 14:34    101:  Job recovery failed for three days
pilotErrorCode (90)        1    0.0    11-12 11:50   1099:  DQ2 staging input file failed
pilotErrorCode (90)        1    0.0    11-12 10:14   1145:  DQ2 get error: md5sum mismatch on input file
pilotErrorCode (90)       33  353.8    11-12 17:45   1150:  Looping job killed by pilot
pilotErrorCode (90)       55  417.1    11-12 14:41   1200:  Job terminated by unknown kill signal
taskBufferErrorCode (8)    8   92.6    11-12 14:01    100:  Job expired and killed six days after submission (or killed by user)
transExitCode (69)        67   44.1    11-12 08:49     40:  Athena crash - consult log file
ANALY_LONG_LYON         defined:0  assigned:0  waiting:0  activated:0  running:0  holding:0  transferring:0  finished:0  failed:1
pilotErrorCode (1)         1    0.4    11-12 16:10   1137:  DQ2 put error: Error in copying the file from job workdir to localSE
ANALY_LPC               defined:0  assigned:0  waiting:0  activated:0  running:0  holding:0  transferring:0  finished:0  failed:2
pilotErrorCode (1)         1    0.1    11-12 09:14   1137:  DQ2 put error: Error in copying the file from job workdir to localSE
taskBufferErrorCode (1)    1    0.0    11-12 09:20    100:  Job expired and killed six days after submission (or killed by user)
ANALY_LPNHE             defined:1  assigned:0  waiting:0  activated:1  running:0  holding:0  transferring:0  finished:0  failed:2
taskBufferErrorCode (2)    2    0.0    11-11 20:31    102:  Expired 6 days after submission
ANALY_LYON              defined:0  assigned:0  waiting:0  activated:0  running:2  holding:0  transferring:0  finished:43  failed:137  (76.1%)
pilotErrorCode (45)       40   34.4    11-12 16:25   1137:  DQ2 put error: Error in copying the file from job workdir to localSE
```

Figure 4.5: Analysis Page showing error codes and Reason for failure

4.3 Cloud Organizations

This is currently a new feature that is introduced in the Panda monitor. It allows monitoring of sites and tasks in the form of organization they belong to. A snapshot of the page is given below. Firstly they are displayed region wise where they are grouped according to the continents and inside that classification we have the big umbrella organizations which have collective information on all the tasks they are assigned. The snapshot shows BNL, UT Arlington, UT Dallas and a few other organizations are represented under the cloud organizations in the United States. Also clicking on the specific cloud organization will give information about the organization about all the queues that are operational with the cloud organization. It also gives some additional information about the number of running, failed and finished jobs.


http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query?dash=clouds
Google
Configuration
Production Clouds DDM PandaMover AutoPilot Sites & Grids Analysis Physics data Usage & Quotas Plots ProdDash DDMDash
Update
Not logged in.
Panda monitor
Quick guide, twiki
Jobs - search
Recent running, activated, waiting, assigned, defined, finished, failed jobs
Select analysis, production, test jobs
Quick search
Job
Dataset
Task
File
Summaries
Blocks: days
Errors: days
Nodes: days
Daily usage
Tasks - search
Generic Task Req
EvGen Task Req
CTBsim Task Req
Task list
Task browser
Datasets - search
Dataset browser
Cloud of US sites

Cloud organization and tasks

Cloud	Sites
CA	ALBERTA-LCG2 TORONTO-LCG2 TRIUMF UBC VICTORIA-LCG2
FR	BEIJING-LCG2 GRIF-DAPNIA IN2P3-CC IN2P3-CPPM IN2P3-LAPP IN2P3-LPC NIPNE TOKYO-LCG2
UK	RAL-LCG2 UKI-LT2-Brunel UKI-LT2-IC-HEP UKI-LT2-QMUL UKI-NORTHGRID-LANCS-HEP UKI-NORTHGRID-LIV-HEP UKI-NORTHGRID-MAN-HEP UKI-SCOTGRID-GLASGOW
US	BNL BostonU IndianaU LouisianaTech MidwestT2 OU SLAC UChicago UMichigan UTArlington UTDallas

Tasks by cloud:

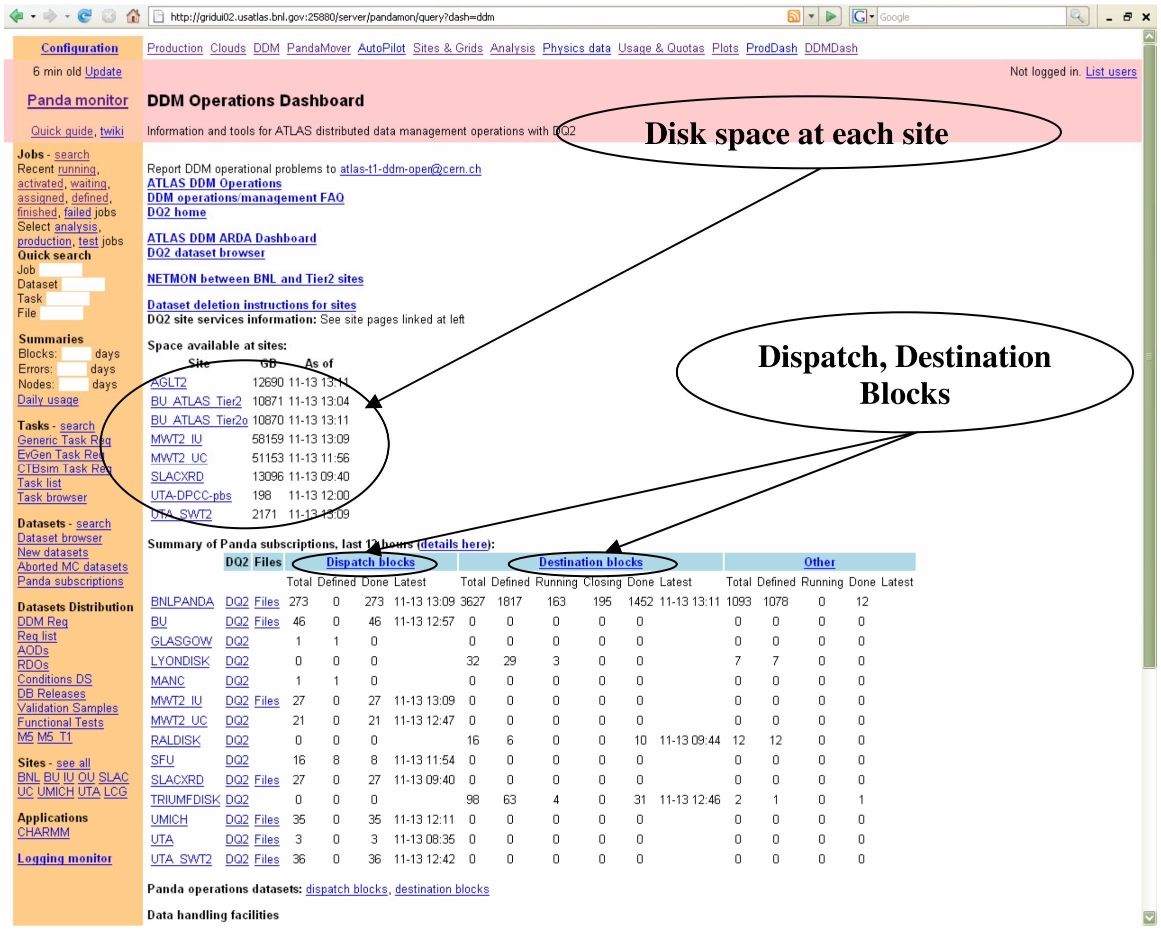
Figure 4.6: A page showing the newly added cloud feature

<u>4.4 DDM and Features</u>

DDM operations monitoring is also supported in Panda, it provides information about the DDM operations going on in between the Panda sites operating in the United States. It also has internal links within the same page to link to ATLAS DDM dashboard which provides collective information about the DDM operations across the globe. A snapshot of the page which pops up for DDM operations is presented below; it provides a comprehensive tabulation of the space available in each of the sites that house DQ2 datasets. UTA is one such site and the snapshot also shows UTA listed there with the available disk space in terms of gigabytes.

Also information related to subscriptions received in each of the sites is put up in a comprehensive manner. It provides figures which indicate the number of dispatch blocks and destination blocks at each of these sites and allow a user to get a snapshot overview of the status.

There is also another feature associated with the DDM operation is the Panda dashboard it provides a link called Panda mover. This provides a detailed status report of the datasets being moved around from site to site mostly the main source here being the Brookhaven National Laboratories keeps moving data around various Tier 2 sites which have enough resources available with them. It shows a concise table where collective numerical information is logged pertinent to the total of the datasets being moved around. This also give a concise snapshot on the status of the DQ2 moving by standard status messages such as done, transFail ,fileExists, maxAttempt, run, active. There is also a detailed description of each of these subscriptions that went to specific sites with the Panda Id, source, destination, status,



http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query?dash=ddm

Configuration

Production Clouds DDM PandaMover AutoPilot Sites & Grids Analysis Physics data Usage & Quotas Plots ProdDash DDMDash

6 min old Update

Not logged in. List users

Panda monitor

DDM Operations Dashboard

Quick guide, twiki

Information and tools for ATLAS distributed data management operations with DQ2

Jobs - search
Recent running, activated, waiting, assigned, defined, finished, failed jobs
Select analysis, production, test jobs
Quick search
Job
Dataset
Task
File

Summaries
Blocks: days
Errors: days
Nodes: days
Daily usage

Tasks - search
Generic Task Req
EvGen Task Req
CTBsim Task Req
Task list
Task browser

Datasets - search
Dataset browser
New datasets
Aborted MC datasets
Panda subscriptions

Datasets Distribution
DDM Req
Req list
AODs
RDOs
Conditions DS
DB Releases
Validation Samples
Functional Tests
M5 M5 T1

Sites - see all
BNL BU IU OU SLAC UC UMICH UTA LCG

Applications
CHARMM

Logging monitor

Report DDM operational problems to atlas-t1-ddm-oper@cern.ch
ATLAS DDM Operations
DDM operations/management FAQ
DQ2 home

ATLAS DDM ARDA Dashboard
DQ2 dataset browser

NETMON between BNL and Tier2 sites

Dataset deletion instructions for sites
DQ2 site services information: See site pages linked at left

Space available at sites:

Site	GB	As of
AGLT2	12690	11-13 13:11
BU_ATLAS_Tier2	10871	11-13 13:04
BU_ATLAS_Tier2o	10870	11-13 13:11
MWT2_IU	58159	11-13 13:09
MWT2_UC	51153	11-13 11:56
SLACXRD	13096	11-13 09:40
UTA-DPCC-pbs	198	11-13 12:00
UTA_SWT2	2171	11-13 13:09

Summary of Panda subscriptions, last 12 hours (details here):

	DQ2	Files	Dispatch blocks				Destination blocks						Other				
			Total	Defined	Done	Latest	Total	Defined	Running	Closing	Done	Latest	Total	Defined	Running	Done	Latest
BNLPANDA	DQ2	Files	273	0	273	11-13 13:09	3627	1817	163	195	1452	11-13 13:11	1093	1078	0	12	
BU	DQ2	Files	46	0	46	11-13 12:57	0	0	0	0	0		0	0	0	0	
GLASGOW	DQ2		1	1	0		0	0	0	0	0		0	0	0	0	
LYONDISK	DQ2		0	0	0		32	29	3	0	0		7	7	0	0	
MANC	DQ2		1	1	0		0	0	0	0	0		0	0	0	0	
MWT2_IU	DQ2	Files	27	0	27	11-13 13:09	0	0	0	0	0		0	0	0	0	
MWT2_UC	DQ2		21	0	21	11-13 12:47	0	0	0	0	0		0	0	0	0	
RALDISK	DQ2		0	0	0		16	6	0	0	10	11-13 09:44	12	12	0	0	
SFU	DQ2		16	8	8	11-13 11:54	0	0	0	0	0		0	0	0	0	
SLACXRD	DQ2	Files	27	0	27	11-13 09:40	0	0	0	0	0		0	0	0	0	
TRIUMFDISK	DQ2		0	0	0		98	63	4	0	31	11-13 12:46	2	1	0	1	
UMICH	DQ2	Files	35	0	35	11-13 12:11	0	0	0	0	0		0	0	0	0	
UTA	DQ2	Files	3	0	3	11-13 08:35	0	0	0	0	0		0	0	0	0	
UTA_SWT2	DQ2	Files	36	0	36	11-13 12:42	0	0	0	0	0		0	0	0	0	

Panda operations datasets: dispatch blocks, destination blocks

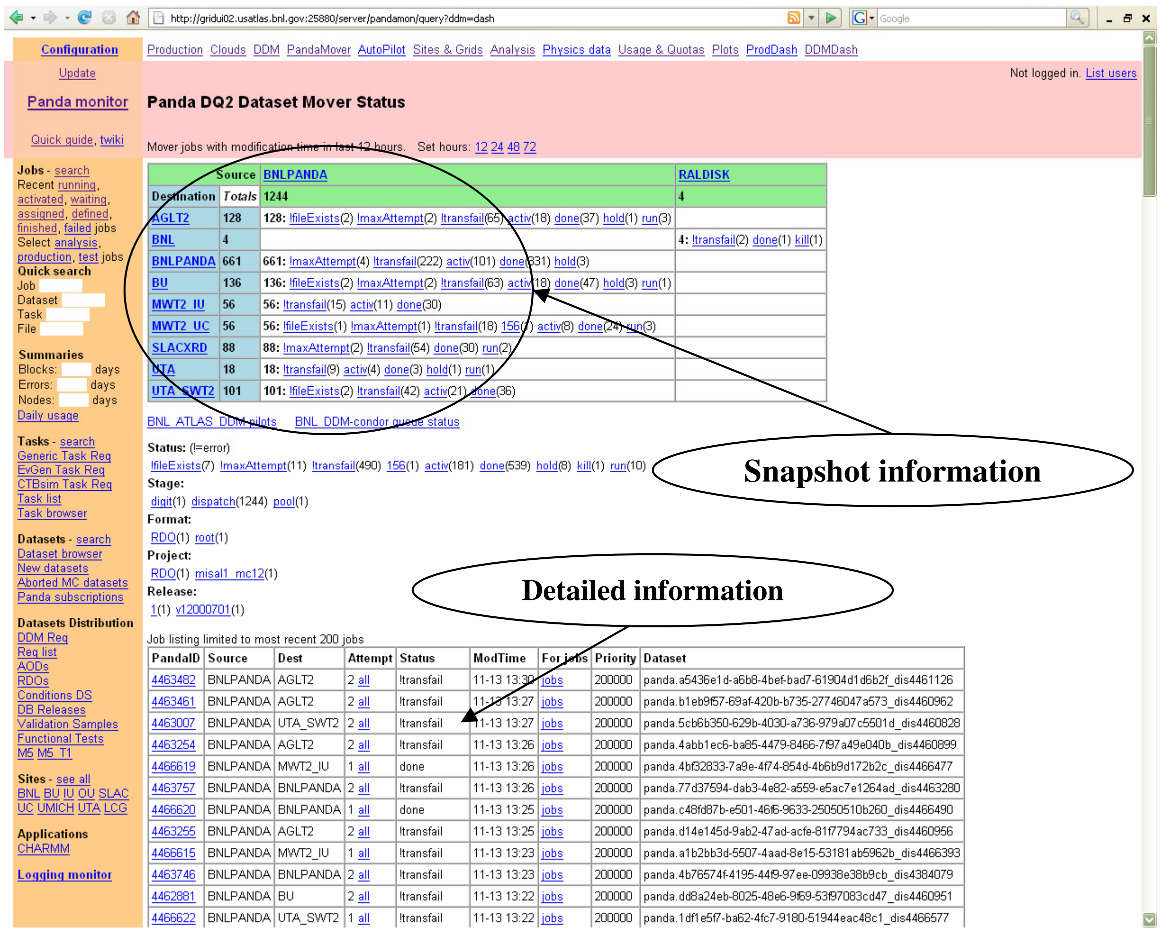
Data handling facilities

Figure 4.7: DDM operations page showing disk space and information about data blocks

dataset Id and name among other information. This helps a user in understanding the status of the subscriptions submitted and database replication going on related to the subscriptions made.

4.5 Autopilot

Auto-Pilot is a simple and generic implementation of Panda pilot and pilot-scheduler for use in more varied environments than the production pilots and schedulers currently in use with Panda.



http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query?ddm=dash

Configuration | Production Clouds DDM PandaMover AutoPilot Sites & Grids Analysis Physics data Usage & Quotas Plots ProdDash DDMDash

Update | Not logged in. List users

Panda monitor

Panda DQ2 Dataset Mover Status

Quick guide, twiki

Mover jobs with modification time in last 12 hours. Set hours: 12 24 48 72

Destination	Source / *Totals*	BNLPANDA	RALDISK
		1244	4
AGLT2	128	128: !fileExists(2) !maxAttempt(2) !transfail(65) activ(18) done(37) hold(1) run(3)	
BNL	4		4: !transfail(2) done(1) kill(1)
BNLPANDA	661	661: !maxAttempt(4) !transfail(222) activ(101) done(331) hold(3)	
BU	136	136: !fileExists(2) !maxAttempt(2) !transfail(63) activ(18) done(47) hold(3) run(1)	
MWT2_IU	56	56: !transfail(15) activ(11) done(30)	
MWT2_UC	56	56: !fileExists(1) !maxAttempt(1) !transfail(18) 156(1) activ(8) done(24) run(3)	
SLACXRD	88	88: !maxAttempt(2) !transfail(54) done(30) run(2)	
UTA	18	18: !transfail(9) activ(4) done(3) hold(1) run(1)	
UTA_SWT2	101	101: !fileExists(2) !transfail(42) activ(21) done(36)	

BNL_ATLAS_DDM pilots BNL_DDM-condor queue status

Status: (!=error)
!fileExists(7) !maxAttempt(11) !transfail(490) 156(1) activ(181) done(539) hold(8) kill(1) run(10)
Stage:
digit(1) dispatch(1244) pool(1)
Format:
RDO(1) root(1)
Project:
RDO(1) misal1_mc12(1)
Release:
1(1) v12000701(1)

Job listing limited to most recent 200 jobs

PandaID	Source	Dest	Attempt	Status	ModTime	For jobs	Priority	Dataset
4463482	BNLPANDA	AGLT2	2 all	!transfail	11-13 13:30	jobs	200000	panda.a5436e1d-a6b8-4bef-bad7-61904d1d6b2f_dis4461126
4463461	BNLPANDA	AGLT2	2 all	!transfail	11-13 13:27	jobs	200000	panda.b1eb9f57-69af-420b-b735-27746047a573_dis4460962
4463007	BNLPANDA	UTA_SWT2	2 all	!transfail	11-13 13:27	jobs	200000	panda.5cb6b350-629b-4030-a736-979a07c5501d_dis4460828
4463254	BNLPANDA	AGLT2	2 all	!transfail	11-13 13:26	jobs	200000	panda.4abb1ec6-ba85-4479-8466-7f97a49e040b_dis4460899
4466619	BNLPANDA	MWT2_IU	1 all	done	11-13 13:26	jobs	200000	panda.4bf32833-7a9e-4f74-854d-4b6b9d172b2c_dis4466477
4463757	BNLPANDA	BNLPANDA	2 all	!transfail	11-13 13:26	jobs	200000	panda.77d37594-dab3-4e82-a559-e5ac7e1264ad_dis4463280
4466620	BNLPANDA	BNLPANDA	1 all	done	11-13 13:25	jobs	200000	panda.c48fd87b-e501-46f6-9633-25050510b260_dis4466490
4463255	BNLPANDA	AGLT2	2 all	!transfail	11-13 13:25	jobs	200000	panda.d14e145d-9ab2-47ad-acfe-81f7794ac733_dis4460956
4466615	BNLPANDA	MWT2_IU	1 all	!transfail	11-13 13:23	jobs	200000	panda.a1b2bb3d-5507-4aad-8e15-53181ab5962b_dis4466393
4463746	BNLPANDA	BNLPANDA	2 all	!transfail	11-13 13:23	jobs	200000	panda.4b76574f-4195-44f9-97ee-09938e38b9cb_dis4384079
4462881	BNLPANDA	BU	2 all	!transfail	11-13 13:22	jobs	200000	panda.dd8a24eb-8025-48e6-9f69-53f97083cd47_dis4460951
4466622	BNLPANDA	UTA_SWT2	1 all	!transfail	11-13 13:22	jobs	200000	panda.1df1e5f7-ba62-4fc7-9180-51944eac48c1_dis4466577

Jobs - search
Recent running, activated, waiting, assigned, defined, finished, failed jobs
Select analysis, production, test jobs
Quick search
Job
Dataset
Task
File

Summaries
Blocks: days
Errors: days
Nodes: days
Daily usage

Tasks - search
Generic Task Req
EvGen Task Req
CTBsim Task Req
Task list
Task browser

Datasets - search
Dataset browser
New datasets
Aborted MC datasets
Panda subscriptions

Datasets Distribution
DDM Req
Req list
AODs
RDOs
Conditions DS
DB Releases
Validation Samples
Functional Tests
M5 M5_T1

Sites - see all
BNL BU IU OU SLAC UC UMICH UTA LCG

Applications
CHARMM

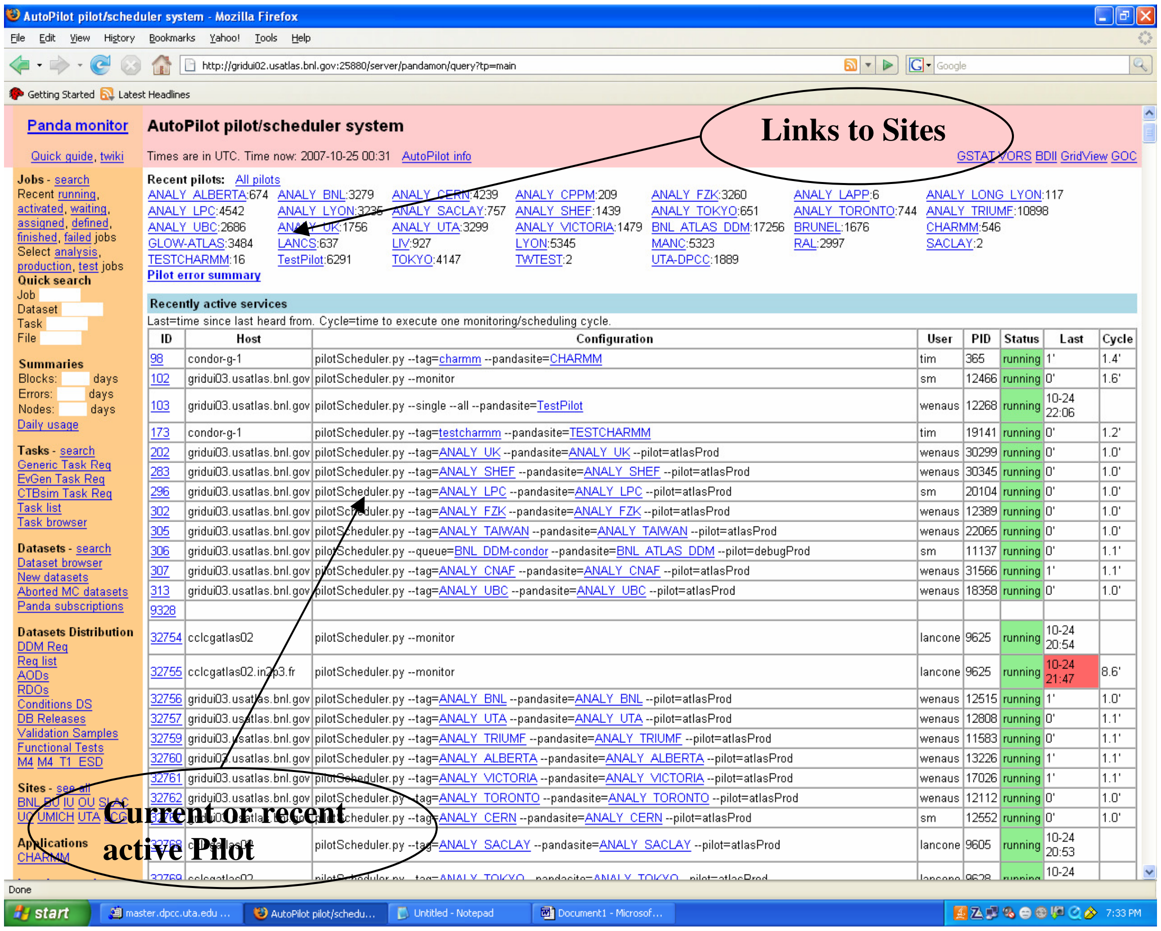
Logging monitor

Figure 4.8: A snapshot pf the newly added autopilot feature

The pilot is a lightweight execution environment used to prepare the computing resources, request the actual payload (a production or user analysis job) from Panda server, execute it, and clean up when the payload has finished. The pilots are broadcasted from the pilot scheduler to the batch systems and the grid sites. The actual payload is scheduled when a CPU becomes available, leading to a low latency for analysis tasks. For robustness, the pilot jobs can be submitted either through Condor-G or locally.

Auto-Pilot provides an pilot implementation that contains no US ATLAS or ATLAS specific content, such that it can be used in a wide range of contexts: within ATLAS but outside OSG, within OSG but outside US ATLAS, from an 'off-grid' laptop or workstation or batch queue, etc.

Firstly, different PANDASITE labels are presented as links. These links lead to pages that give a list of all recently scheduled pilots with that particular site.



AutoPilot pilot/scheduler system - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query?tp=main
Getting Started Latest Headlines
Links to Sites
Current or recent active Pilot
Done
start master.dpcc.uta.edu … AutoPilot pilot/schedu… Untitled - Notepad Document1 - Microsof… 7:33 PM

Panda monitor

Quick guide, twiki

Jobs - search
Recent running, activated, waiting, assigned, defined, finished, failed jobs
Select analysis, production, test jobs
Quick search
Job
Dataset
Task
File

Summaries
Blocks: days
Errors: days
Nodes: days
Daily usage

Tasks - search
Generic Task Req
EvGen Task Req
CTBsim Task Req
Task list
Task browser

Datasets - search
Dataset browser
New datasets
Aborted MC datasets
Panda subscriptions

Datasets Distribution
DDM Req
Req list
AODs
RDOs
Conditions DS
DB Releases
Validation Samples
Functional Tests
M4 M4_T1_ESD

Sites - see all
BNL BU IU OU SLAC
UC UMICH UTA CCG

Applications
CHARMM

AutoPilot pilot/scheduler system

Times are in UTC. Time now: 2007-10-25 00:31 AutoPilot info

GSTAT VORS BDII GridView GOC

Recent pilots: All pilots

ANALY_ALBERTA:674 ANALY_BNL:3279 ANALY_CERN:4239 ANALY_CPPM:209 ANALY_FZK:3260 ANALY_LAPP:6 ANALY_LONG_LYON:117
ANALY_LPC:4542 ANALY_LYON:3235 ANALY_SACLAY:757 ANALY_SHEF:1439 ANALY_TOKYO:651 ANALY_TORONTO:744 ANALY_TRIUMF:10898
ANALY_UBC:2686 ANALY_UK:1756 ANALY_UTA:3299 ANALY_VICTORIA:1479 BNL_ATLAS_DDM:17256 BRUNEL:1676 CHARMM:546
GLOW-ATLAS:3484 LANCS:637 LIV:927 LYON:5345 MANC:5323 RAL:2997 SACLAY:2
TESTCHARMM:16 TestPilot:6291 TOKYO:4147 TWTEST:2 UTA-DPCC:1889

Pilot error summary

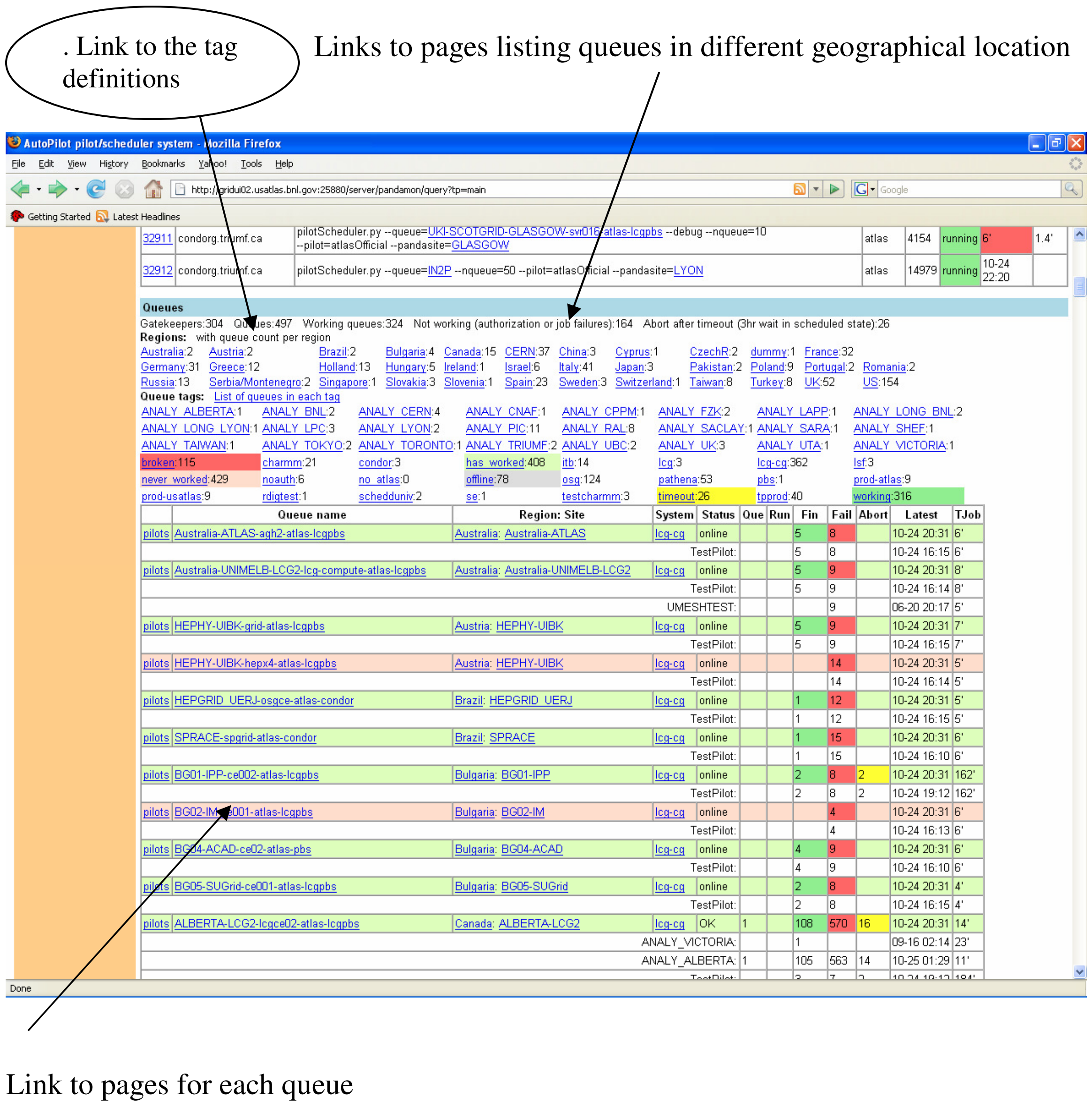
Recently active services

Last=time since last heard from. Cycle=time to execute one monitoring/scheduling cycle.

ID	Host	Configuration	User	PID	Status	Last	Cycle
98	condor-g-1	pilotScheduler.py --tag=charmm --pandasite=CHARMM	tim	365	running	1'	1.4'
102	gridui03.usatlas.bnl.gov	pilotScheduler.py --monitor	sm	12466	running	0'	1.6'
103	gridui03.usatlas.bnl.gov	pilotScheduler.py --single --all --pandasite=TestPilot	wenaus	12268	running	10-24 22:06	
173	condor-g-1	pilotScheduler.py --tag=testcharmm --pandasite=TESTCHARMM	tim	19141	running	0'	1.2'
202	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_UK --pandasite=ANALY_UK --pilot=atlasProd	wenaus	30299	running	0'	1.0'
283	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_SHEF --pandasite=ANALY_SHEF --pilot=atlasProd	wenaus	30345	running	0'	1.0'
296	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_LPC --pandasite=ANALY_LPC --pilot=atlasProd	sm	20104	running	0'	1.0'
302	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_FZK --pandasite=ANALY_FZK --pilot=atlasProd	wenaus	12389	running	0'	1.0'
305	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_TAIWAN --pandasite=ANALY_TAIWAN --pilot=atlasProd	wenaus	22065	running	0'	1.0'
306	gridui03.usatlas.bnl.gov	pilotScheduler.py --queue=BNL_DDM-condor --pandasite=BNL_ATLAS_DDM --pilot=debugProd	sm	11137	running	0'	1.1'
307	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_CNAF --pandasite=ANALY_CNAF --pilot=atlasProd	wenaus	31566	running	1'	1.1'
313	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_UBC --pandasite=ANALY_UBC --pilot=atlasProd	wenaus	18358	running	0'	1.0'
9328							
32754	cclcgatlas02	pilotScheduler.py --monitor	lancone	9625	running	10-24 20:54	
32755	cclcgatlas02.in2p3.fr	pilotScheduler.py --monitor	lancone	9625	running	10-24 21:47	8.6'
32756	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_BNL --pandasite=ANALY_BNL --pilot=atlasProd	wenaus	12515	running	1'	1.0'
32757	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_UTA --pandasite=ANALY_UTA --pilot=atlasProd	wenaus	12808	running	0'	1.1'
32759	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_TRIUMF --pandasite=ANALY_TRIUMF --pilot=atlasProd	wenaus	11583	running	0'	1.1'
32760	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_ALBERTA --pandasite=ANALY_ALBERTA --pilot=atlasProd	wenaus	13226	running	1'	1.1'
32761	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_VICTORIA --pandasite=ANALY_VICTORIA --pilot=atlasProd	wenaus	17026	running	1'	1.1'
32762	gridui03.usatlas.bnl.gov	pilotScheduler.py --tag=ANALY_TORONTO --pandasite=ANALY_TORONTO --pilot=atlasProd	wenaus	12112	running	0'	1.0'
[illegible]	[illegible]	pilotScheduler.py --tag=ANALY_CERN --pandasite=ANALY_CERN --pilot=atlasProd	sm	12552	running	0'	1.0'
32768	[illegible]	pilotScheduler.py --tag=ANALY_SACLAY --pandasite=ANALY_SACLAY --pilot=atlasProd	lancone	9605	running	10-24 20:53	
32769	cclcgatlas02	pilotScheduler.py --tag=ANALY_TOKYO --pandasite=ANALY_TOKYO --pilot=atlasProd	lancone	9628	running	10-24	

Figure 4.9: Autopilot page showing recent or running pilots

This page also provides information about current or recent active Pilot schedulers. Information like ID, name of the machine hosting this scheduler instance, the pandasite that the pilots being scheduled contain, queue or tag (group of queues) the pilots are being scheduled to by this scheduler, who started the scheduler, state of the scheduler etc. PANDASITE is a very important criterion while deciding a computing node for a payload or a job.


. Link to the tag definitions
Links to pages listing queues in different geographical location
Link to pages for each queue

AutoPilot pilot/scheduler system - Mozilla Firefox

File Edit View History Bookmarks Yahoo! Tools Help

http://gridui02.usatlas.bnl.gov:25880/server/pandamon/query?tp=main

Getting Started Latest Headlines

32911	condorg.triumf.ca	pilotScheduler.py --queue=UKI-SCOTGRID-GLASGOW-svr016-atlas-lcgpbs --debug --nqueue=10 --pilot=atlasOfficial --pandasite=GLASGOW	atlas	4154	running	6'	1.4'
32912	condorg.triumf.ca	pilotScheduler.py --queue=IN2P --nqueue=50 --pilot=atlasOfficial --pandasite=LYON	atlas	14979	running	10-24 22:20	

Queues

Gatekeepers:304 Queues:497 Working queues:324 Not working (authorization or job failures):164 Abort after timeout (3hr wait in scheduled state):26

Regions: with queue count per region

Australia:2	Austria:2	Brazil:2	Bulgaria:4	Canada:15	CERN:37	China:3	Cyprus:1	CzechR:2	dummy:1	France:32	
Germany:31	Greece:12	Holland:13	Hungary:5	Ireland:1	Israel:6	Italy:41	Japan:3	Pakistan:2	Poland:9	Portugal:2	Romania:2
Russia:13	Serbia/Montenegro:2	Singapore:1	Slovakia:3	Slovenia:1	Spain:23	Sweden:3	Switzerland:1	Taiwan:8	Turkey:8	UK:52	US:154

Queue tags: List of queues in each tag

ANALY_ALBERTA:1	ANALY_BNL:2	ANALY_CERN:4	ANALY_CNAF:1	ANALY_CPPM:1	ANALY_FZK:2	ANALY_LAPP:1	ANALY_LONG_BNL:2
ANALY_LONG_LYON:1	ANALY_LPC:3	ANALY_LYON:2	ANALY_PIC:11	ANALY_RAL:8	ANALY_SACLAY:1	ANALY_SARA:1	ANALY_SHEF:1
ANALY_TAIWAN:1	ANALY_TOKYO:2	ANALY_TORONTO:1	ANALY_TRIUMF:2	ANALY_UBC:2	ANALY_UK:3	ANALY_UTA:1	ANALY_VICTORIA:1
broken:115	charmm:21	condor:3	has_worked:408	itb:14	lcg:3	lcg-cg:362	lsf:3
never_worked:429	noauth:6	no_atlas:0	offline:78	osg:124	pathena:53	pbs:1	prod-atlas:9
prod-usatlas:9	rdigtest:1	schedduniv:2	se:1	testcharmm:3	timeout:26	tpprod:40	working:316

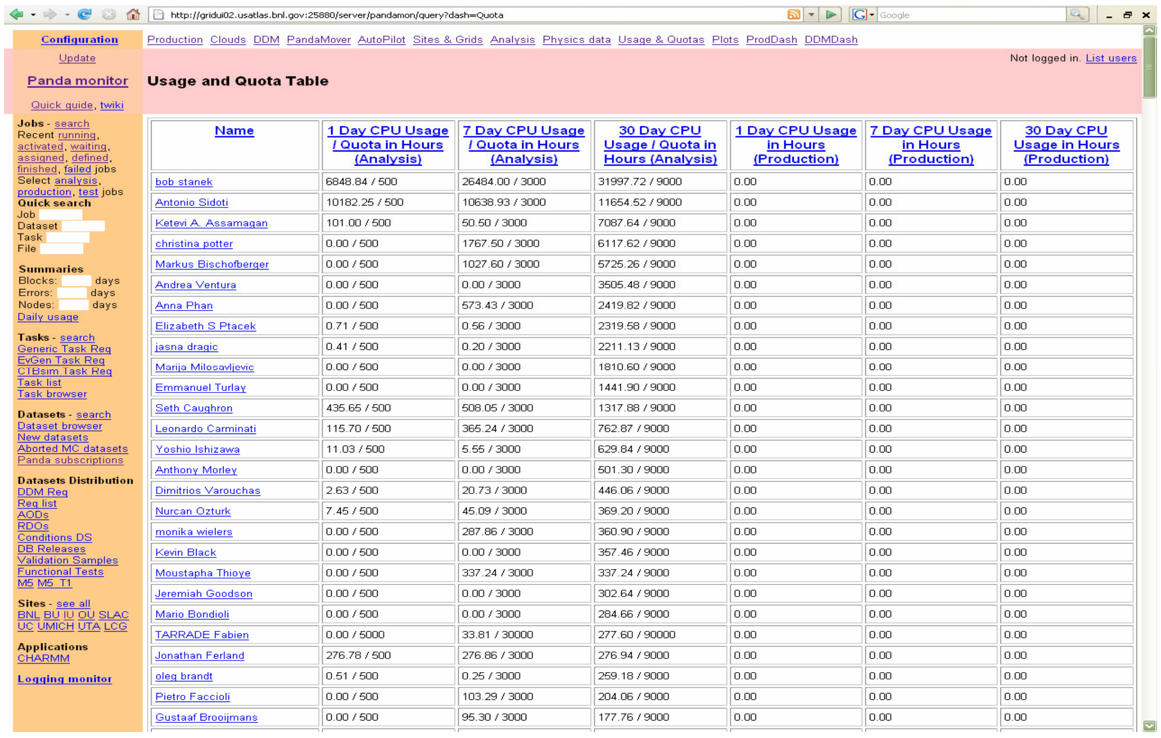
	Queue name	Region: Site	System	Status	Que	Run	Fin	Fail	Abort	Latest	TJob
pilots	Australia-ATLAS-agh2-atlas-lcgpbs	Australia: Australia-ATLAS	lcg-cg	online			5	8		10-24 20:31	6'
				TestPilot:			5	8		10-24 16:15	6'
pilots	Australia-UNIMELB-LCG2-lcg-compute-atlas-lcgpbs	Australia: Australia-UNIMELB-LCG2	lcg-cg	online			5	9		10-24 20:31	8'
				TestPilot:			5	9		10-24 16:14	8'
				UMESHTEST:				9		06-20 20:17	5'
pilots	HEPHY-UIBK-grid-atlas-lcgpbs	Austria: HEPHY-UIBK	lcg-cg	online			5	9		10-24 20:31	7'
				TestPilot:			5	9		10-24 16:15	7'
pilots	HEPHY-UIBK-hepx4-atlas-lcgpbs	Austria: HEPHY-UIBK	lcg-cg	online				14		10-24 20:31	5'
				TestPilot:				14		10-24 16:14	5'
pilots	HEPGRID_UERJ-osgce-atlas-condor	Brazil: HEPGRID_UERJ	lcg-cg	online			1	12		10-24 20:31	5'
				TestPilot:			1	12		10-24 16:15	5'
pilots	SPRACE-spgrid-atlas-condor	Brazil: SPRACE	lcg-cg	online			1	15		10-24 20:31	6'
				TestPilot:			1	15		10-24 16:10	6'
pilots	BG01-IPP-ce002-atlas-lcgpbs	Bulgaria: BG01-IPP	lcg-cg	online			2	8	2	10-24 20:31	162'
				TestPilot:			2	8	2	10-24 19:12	162'
pilots	BG02-IM-ce001-atlas-lcgpbs	Bulgaria: BG02-IM	lcg-cg	online				4		10-24 20:31	6'
				TestPilot:				4		10-24 16:13	6'
pilots	BG04-ACAD-ce02-atlas-pbs	Bulgaria: BG04-ACAD	lcg-cg	online			4	9		10-24 20:31	6'
				TestPilot:			4	9		10-24 16:10	6'
pilots	BG05-SUGrid-ce001-atlas-lcgpbs	Bulgaria: BG05-SUGrid	lcg-cg	online			2	8		10-24 20:31	4'
				TestPilot:			2	8		10-24 16:15	4'
pilots	ALBERTA-LCG2-lcgce02-atlas-lcgpbs	Canada: ALBERTA-LCG2	lcg-cg	OK	1		108	570	16	10-24 20:31	14'
				ANALY_VICTORIA:			1			09-16 02:14	23'
				ANALY_ALBERTA:	1		105	563	14	10-25 01:29	11'

Done

Figure 4.10: Autopilot page showing queues and tag names

4.6 Usage and Quotas

Another feature that panda monitor allows is to see the allotted quota of resources for each user who submits analysis and production jobs. As production jobs start coming in, once the experiment begins it will become inevitable to restrict people from submitting excessive jobs subscriptions based on their role in the experiment. The quotas for submission will be varied according to the user's organization. To monitor resource management issues, a page is put up where a user's utilization of the grid resources is tabulated in daily, weekly and monthly figures for analysis and production jobs. This feature not only allows the individual user to monitor his available quota for the day but also allows grid administrators to monitor the usage of each person individually.



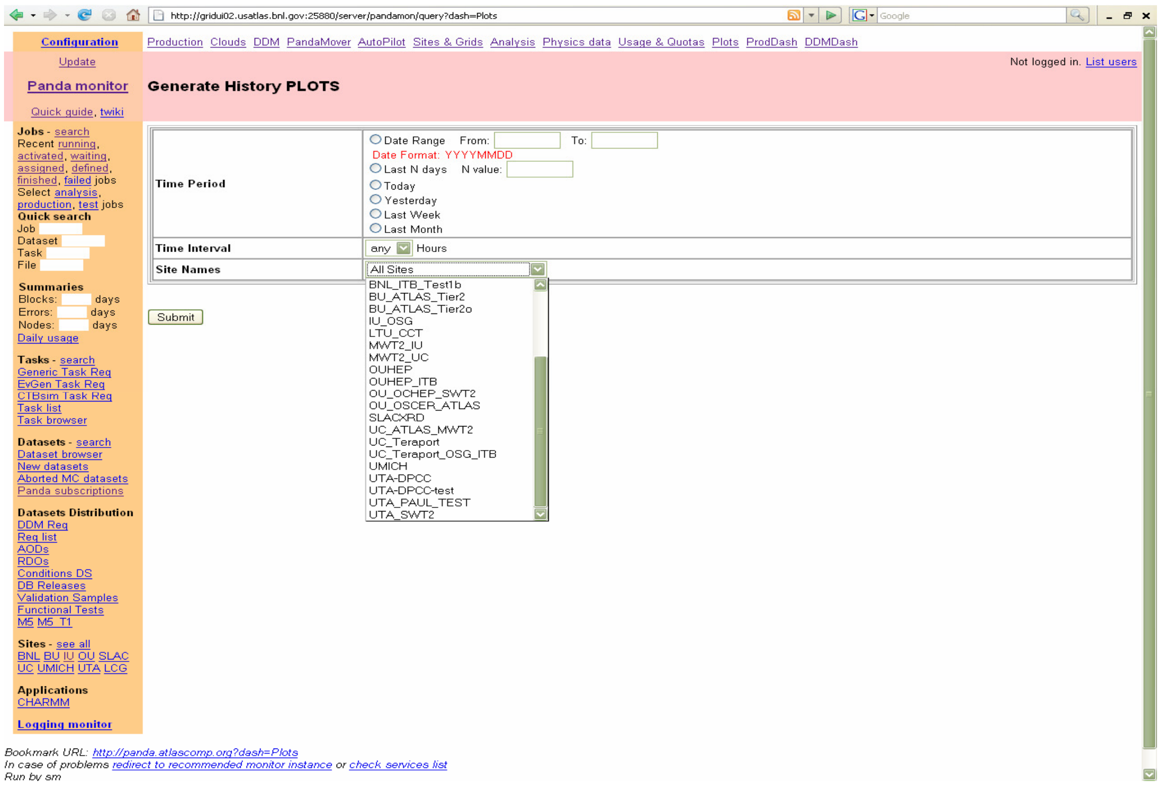
Usage and Quota Table

Name	1 Day CPU Usage / Quota in Hours (Analysis)	7 Day CPU Usage / Quota in Hours (Analysis)	30 Day CPU Usage / Quota in Hours (Analysis)	1 Day CPU Usage in Hours (Production)	7 Day CPU Usage in Hours (Production)	30 Day CPU Usage in Hours (Production)
bob stanek	6848.84 / 500	26484.00 / 3000	31997.72 / 9000	0.00	0.00	0.00
Antonio Sidoti	10182.25 / 500	10638.93 / 3000	11654.52 / 9000	0.00	0.00	0.00
Ketevi A. Assamagan	101.00 / 500	50.50 / 3000	7087.64 / 9000	0.00	0.00	0.00
christina potter	0.00 / 500	1767.50 / 3000	6117.62 / 9000	0.00	0.00	0.00
Markus Bischofberger	0.00 / 500	1027.60 / 3000	5725.26 / 9000	0.00	0.00	0.00
Andrea Ventura	0.00 / 500	0.00 / 3000	3505.48 / 9000	0.00	0.00	0.00
Anna Phan	0.00 / 500	573.43 / 3000	2419.82 / 9000	0.00	0.00	0.00
Elizabeth S Ptacek	0.71 / 500	0.56 / 3000	2319.58 / 9000	0.00	0.00	0.00
jasna dragic	0.41 / 500	0.20 / 3000	2211.13 / 9000	0.00	0.00	0.00
Marija Milosavljevic	0.00 / 500	0.00 / 3000	1810.60 / 9000	0.00	0.00	0.00
Emmanuel Turlay	0.00 / 500	0.00 / 3000	1441.90 / 9000	0.00	0.00	0.00
Seth Caughron	435.65 / 500	508.05 / 3000	1317.88 / 9000	0.00	0.00	0.00
Leonardo Carminati	115.70 / 500	365.24 / 3000	762.87 / 9000	0.00	0.00	0.00
Yoshio Ishizawa	11.03 / 500	5.55 / 3000	629.84 / 9000	0.00	0.00	0.00
Anthony Morley	0.00 / 500	0.00 / 3000	501.30 / 9000	0.00	0.00	0.00
Dimitrios Varouchas	2.63 / 500	20.73 / 3000	446.06 / 9000	0.00	0.00	0.00
Nurcan Ozturk	7.45 / 500	45.09 / 3000	369.20 / 9000	0.00	0.00	0.00
monika wielers	0.00 / 500	287.86 / 3000	360.90 / 9000	0.00	0.00	0.00
Kevin Black	0.00 / 500	0.00 / 3000	357.46 / 9000	0.00	0.00	0.00
Moustapha Thioye	0.00 / 500	337.24 / 3000	337.24 / 9000	0.00	0.00	0.00
Jeremiah Goodson	0.00 / 500	0.00 / 3000	302.64 / 9000	0.00	0.00	0.00
Mario Bondioli	0.00 / 500	0.00 / 3000	284.66 / 9000	0.00	0.00	0.00
TARRADE Fabien	0.00 / 5000	33.81 / 30000	277.60 / 90000	0.00	0.00	0.00
Jonathan Ferland	276.78 / 500	276.86 / 3000	276.94 / 9000	0.00	0.00	0.00
oleg brandt	0.51 / 500	0.25 / 3000	259.18 / 9000	0.00	0.00	0.00
Pietro Faccioli	0.00 / 500	103.29 / 3000	204.06 / 9000	0.00	0.00	0.00
Gustaaf Brooijmans	0.00 / 500	95.30 / 3000	177.76 / 9000	0.00	0.00	0.00

Figure 4.11: A snapshot of the User Quota page listing all users

4.7 History Plots

This feature allows plotting of graphs for specific intervals which are obtained as inputs from the user, the user is allowed to query for the present day, yesterday, a week or month based on his requirements. The inputs are collected from the user as to if he wishes to see the graphs plotted for specific sites or if he wishes to collectively monitor the performance of all sites in which case he chooses the All Sites option from the drop down box. Alternatively the user is allowed to specify his own dates which he is interested in to see the graphs for and he can also choose an interval limit in which he like to see the plots like a typical query could look like weekly plots for UTA-Dpcc at an interval of 1 day .



Generate History PLOTS

Time Period	○ Date Range From: To: Date Format: YYYYMMDD ○ Last N days N value: ○ Today ○ Yesterday ○ Last Week ○ Last Month
Time Interval	any Hours
Site Names	All Sites

BNL_ITB_Test1b
BU_ATLAS_Tier2
BU_ATLAS_Tier2o
IU_OSG
LTU_CCT
MWT2_IU
MWT2_UC
OUHEP
OUHEP_ITB
OU_OCHEP_SWT2
OU_OSCER_ATLAS
SLACXRD
UC_ATLAS_MWT2
UC_Teraport
UC_Teraport_OSG_ITB
UMICH
UTA-DPCC
UTA-DPCC-test
UTA_PAUL_TEST
UTA_SWT2

Bookmark URL: http://panda.atlascomp.org?dash=Plots
In case of problems redirect to recommended monitor instance or check services list
Run by sm

Figure 4.12: A snapshot of the querying page of History Plots

The above figure shows a snapshot of the page which allows the user to choose the different options that are available for querying.

Below, is a graph given which is obtained as a result of a query of collective running jobs on all sites for the past week at an interval of 1 day.

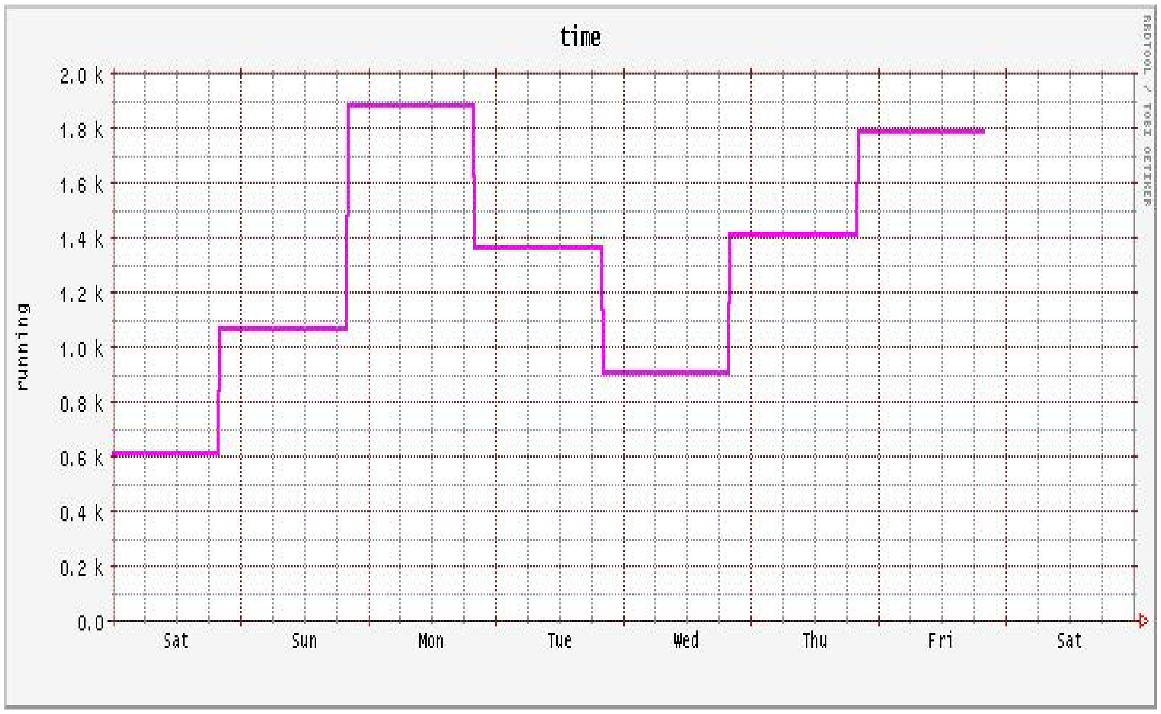


Figure 4.13: A graph that was generated from the History plots application

CHAPTER 5

PANDA MONITOR ARCHITECTURE

5.1 Monitor Architecture Overview

The following diagram represents a detailed architecture view of the Panda monitor and gives an account of its major components .The major components of the Panda monitor can be given as:

- Request handler
- DB utils
- Panda monitor tils
- HTML utils
- Panda logger
- Database

Each of the above components is elaborated below illustrating their specific roles in the effective building up of the panda monitor.

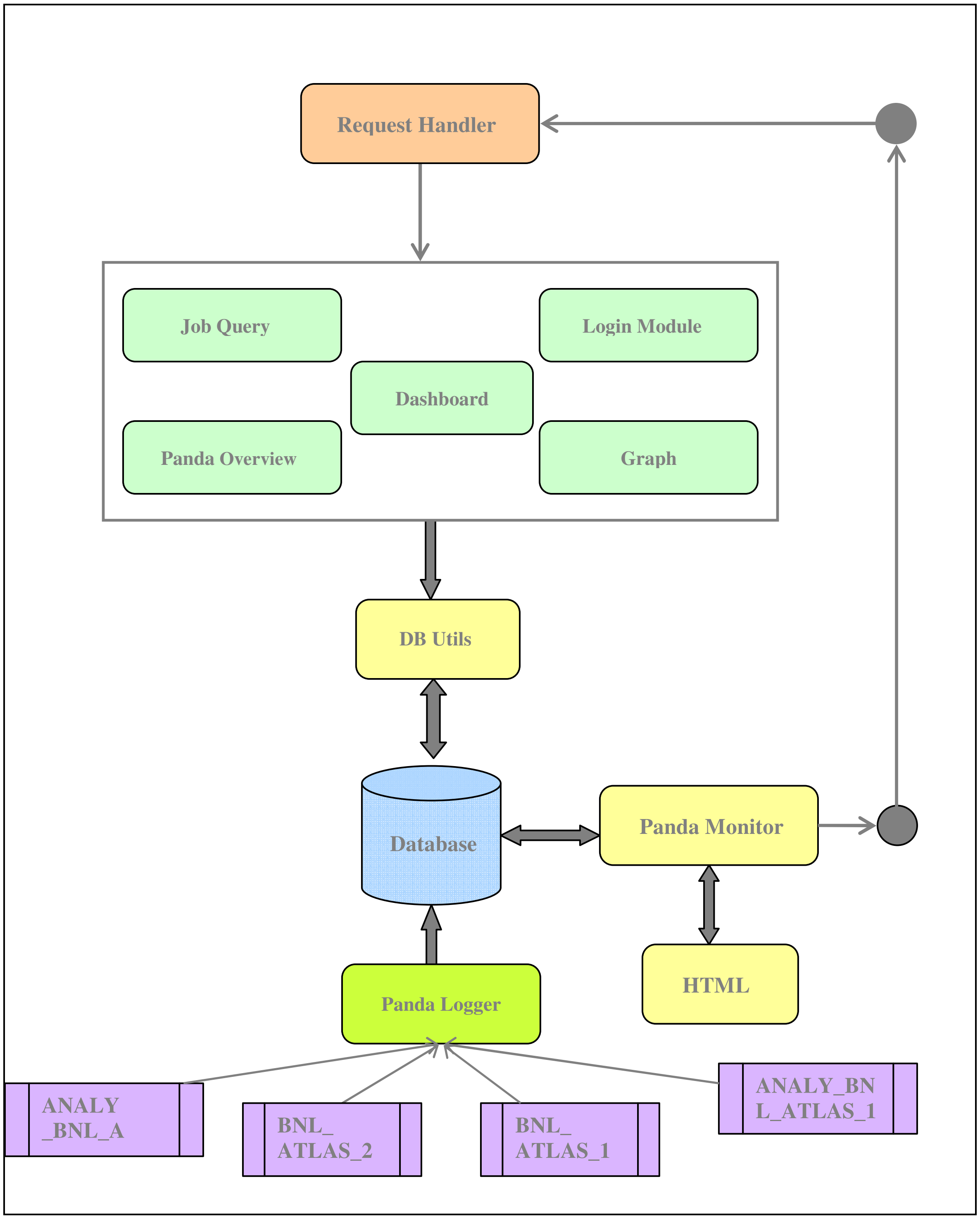


Figure 5.1: A detailed architecture view of the existing monitor

5.2 The Request Handler

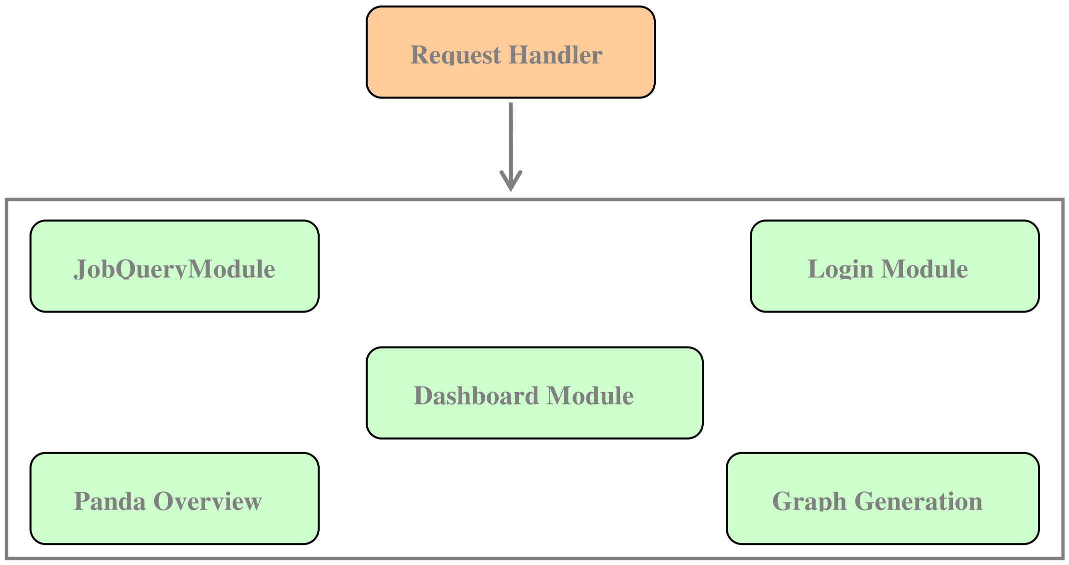


Figure 5.2: Representation of the Request Handler

The current Panda monitor helps in monitoring important information such as the number of jobs running collectively in Panda during a given period of time; the same can be assessed individually for every site. This turns out to be an important factor in determining the individual performance.

The request handler module acts as the main gateway for the entire Panda monitor. This module allows interaction between the various modules that are the building components of Panda monitor. To access the different modules, the Panda monitor user interface provides links that helps in passing different parameters to differentiate the modules to be accessed. Based on the parameters passed from the user interface of the Panda monitor, the request handler routes the calls to the various modules like job query, dashboard, and graph generation. The request handler acts like an abstract layer that encompasses all the different module invocations. In the process, basic steps like retrieving the result from the respective modules, building the html result page are also handled. Eventually, this module, acts as a pivotal module, thereby ensuring layered architecture for the Panda monitor.

Each of the above modules (in green), have a specific action to be performed. These modules are self sufficient and hence, contain all the associated functions in it. Panda overview module aids in generating the initial index pages for all the modules involved in the specific layer. It acts as an under lying module for the dashboard module and the job query module. Apart from these, there are enormous functionalities of the Panda overview; they help in viewing the log files, error handling and all search related functionalities. Hence, Panda overview module guarantees all the basic functionalities panda monitor to be ported as a web page

The login module is also found in this application. This module provides/restricts access for the users of the system, by which the security of the system is ensured.

<u>5.3 The Database</u>

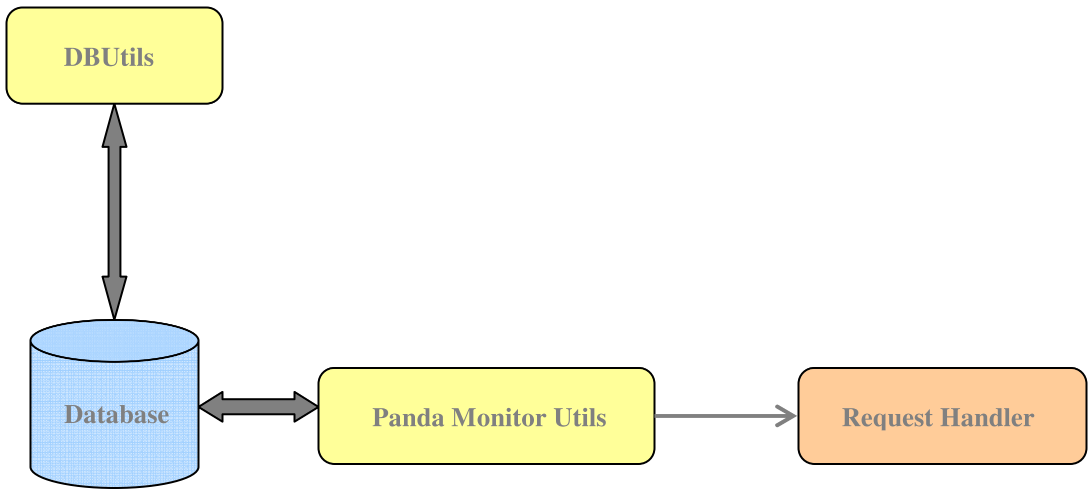


Figure 5.3: Illustration of Database operations

The DB utilities module provides the necessary interface to interact with the database involved in the Panda monitor. The Panda monitor utilities; module is the one where the database results are linked on the result page to be displayed for any call to the Panda monitor. This generated resultant page is passed over to the request handler module, which in turn displays it in the browser. Thus, it is evident that Panda monitor utilities module acts in lower level to ensure the seamless working of the system. The entire page rendering functions and dataset typecasting functions are handled in this module. This module also checks the server status and monitors the configuration settings for the application.

As it is the case with all the application systems, there is a centralized database, which collectively holds all the necessary data for the Panda monitor. This acts as a back bone for this complete application.

5.4 Logging and Utilities

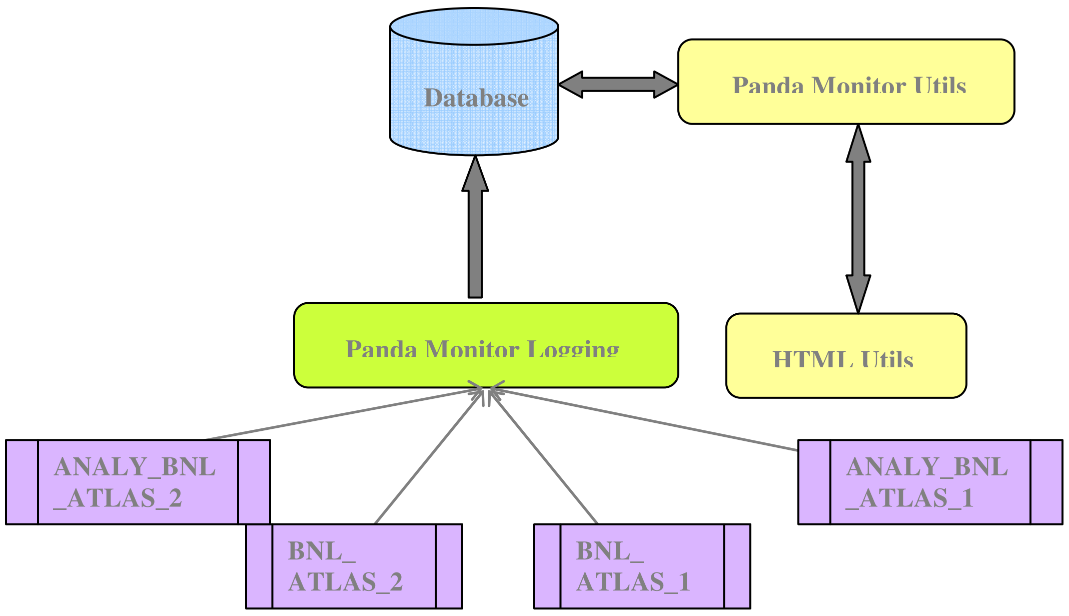


Figure 5.4: A representation to show logging of data to panda central database

5.5 HTML Utilities

This module aids in forming link displays in the pages, links to the pages and also acts as a style sheet to the elements in the HTML pages used in the application. This helps in customizing the user interface and serves as a necessary utility to build the outline of the panda monitor

5.6 Panda Logger

The Panda monitor has an extensive logging facility. This is done using the module, Panda logger. Logging is done using the HTTP handler for each page and also the respective locks required for such transactions are also ensured.

CHAPTER 6

PROPOSAL FOR A ROBUST MONITOR

6.1 Introduction and Groundwork

Panda monitor when created, served the needs of the scientists and grid administrators efficiently. But as it kept evolving, many problems were slowly being uncovered with the current monitor. One of the major concerns with the existing monitor was that it was responding very slowly to even normal queries to the system. Queries such as information about running or assigned jobs sometimes take close to a minute to build all the information needed for that page. This time delay is unacceptable; however the major concern was due to the fact that the monitor was just being tested with around 6000-8000 test jobs per day at the moment. This number could go as high as 200,000 jobs when the experiment begins. This raised the alarm and indicated that current monitor will not scale well once the experiment begins. The scalability issue with the current monitor could be attributed to group of factors.

The first thing that could be noted down in the current version of the monitor is the absence of any kind of features incorporated in the web such as JavaScript and XML to deal with the huge amount of information .The content in each page is relatively large to a normal web based service since the experiment demands it that way. Also the way in which information was organized in each of the pages posed a problem. A clear differentiation between users and administrators could not be made. A simple example could be cited as, if a person would like to access the page for running, defined or finished jobs, for example, he will be provided with all the information related to the kind of job he or she requested which was submitted by all panda users. Usually this kind of information spans to a large number of pages and will also be difficult for the user to scroll down to look into selective information that he or she is interested in. A person usually accessing such a page will be more interested in his/her jobs submitted or jobs related to his/her organization.

This kind of user level filter was not in the goal of Panda monitor when it was designed initially. Collective information such as the one described above proved to be very useful to system administrators or super administrators who would like to see all the information in one click.

Panda is evolving faster than it was originally expected to; plans are ongoing to make it common grid middleware which could help experiments using OSG sites. One such experiment which already took to using Panda for their purposes is CHAARM (Chemistry at HARvard Molecular Mechanics). When experiments are using Panda to be their new grid middleware and when panda is moving to the next level as a successful grid manager middleware, there is an absolute need for a customized and scalable monitor which can appeal to more experiments to use panda as a part of their grid management needs.

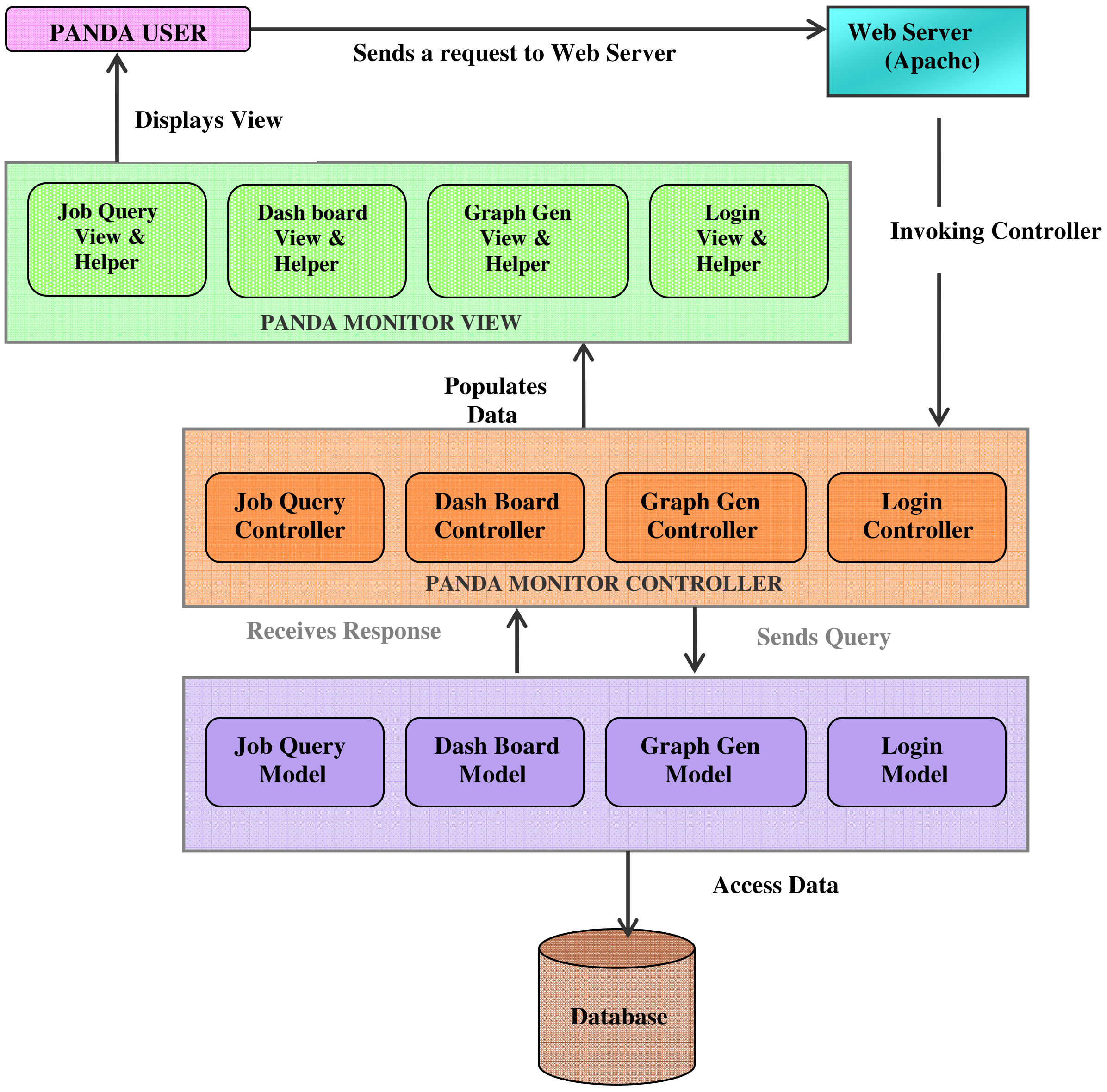


Figure 6.1: Architectural view of the Proposed Panda Monitor

The first obvious conclusion that could be arrived is to make the existing monitor more scalable with the use of the new generation of web paradigm called AJAX which is an acronym for Asynchronous Java and XML, the use of this kind of feature in a page which has to deal with huge amounts of information, could actually reduce the amount of information fetched every time without actually reloading the currently available content in the page. So the idea is pretty simple as the user is still going through the information in the page, any thing new that he requests will actually be made available without reloading the page. This could be a potentially apt solution for making the monitor quiet responsive in terms of turn around time and interactivity.

However when Panda wants to scale to another level by implementing more interactivity in the form of graphs and better ways of representation other than just text, the current monitor could still pose a problem since it was written in Python and there are not many options which offer good scalability in the web.

The next thing that was discussed to overcome the problem was to adapt user based logins for Panda and give a set of default options in the start up page where the user will have a fixed set of links which might be of interest to him. Hence pages for general users and administrators could be kept separate and users will also be given the option of actually customizing their pages according to their needs. This could also be coupled with implementing sessions so that the pattern of search of each user could be tracked and a set of default pages that he browsed previously could be automatically be loaded in the background without the user knowledge; so that if he tries to access the same pages now, he will feel them load faster. This could considerably improve the performance of the current architecture of Panda monitor.

The need for a rapid application development model was also discussed since it was understood that all requirements for an experiment with the enormity such as ATLAS could not be met upfront; this raised the need for a system which will allow applications to be rolled out of the door within a few days of request, the current model proved to be a hassle for such flexibility in terms of time. This requirement raised the need for a new architecture with proven scalability in terms of the Internet.

The search was on for a system which would curb the problems in the current version of the monitor by adapting to changing needs and the necessity for rapid application development in the monitor.

A decision was made to adopt a completely new Technology which is well formulated for web based applications, one which was already proven to scale well in handling huge amounts of information.

The other things that were looked into for the implementation of the Panda monitor resulted in the need for an inherent compliance to the new emerging Web standards cross compatibility and seamless performance across all browsers.

The need for maintenance of code was realized and a system which would enforce clean coding practices and will also allow easy maintenance of the code after development was the one which would fit the bill.

Ruby on Rails was considered as a potential candidate and was finally decided to be employed in the creation of the new generation of the Panda monitor.

6.2 Ruby on Rails and MVC Architecture

Ruby on Rails is a new Web application framework which facilitates good programming practices, this eventually leads to neatly structured and efficient code among many other benefits some of them are discussed below.

Much of the power of Rails comes from the Ruby programming language. Ruby's unique design makes it easy to create domain-specific languages and to do metaprogramming. Rails takes full advantage of this.

Rails is an MVC (model, view, controller) framework where Rails provides all the layers and they work together seamlessly. Other frameworks often implement only part of the solution, requiring the developer to integrate multiple frameworks into the application and then coerce them into working together. For example, a Java developer might use Hibernate, Struts, and Tiles to get full MVC support. This was exactly the problem with the old version of the Panda monitor it was not strictly designed to confirm to the MVC architecture. This resulted in a big problem at a later stage, since it did not confirm to code practices of the object oriented world and this eventually led to a code maintainability problem at the later stages of the current monitor.

The below diagram is drawn to represent the flow of control in the MVC architecture. The MVC architecture is gaining a lot of momentum recently, it clearly gives a cutting edge over any other suggested architectures since it enforces strict coding practices and flow of control in a sequential manner. This eventually reduces the lines of code that needs to be typed to create any application, not only saving time while creating them but also leads to easy code maintainability. In the above diagram, we can see the model view and controller clearly sketched out, The model is responsible for interacting with the databases; any transaction to the databases should always go through the model .The controller is responsible for actually interacting with the users through a web page from which it takes user requests and directs them accordingly to the model or to internal controller applications.

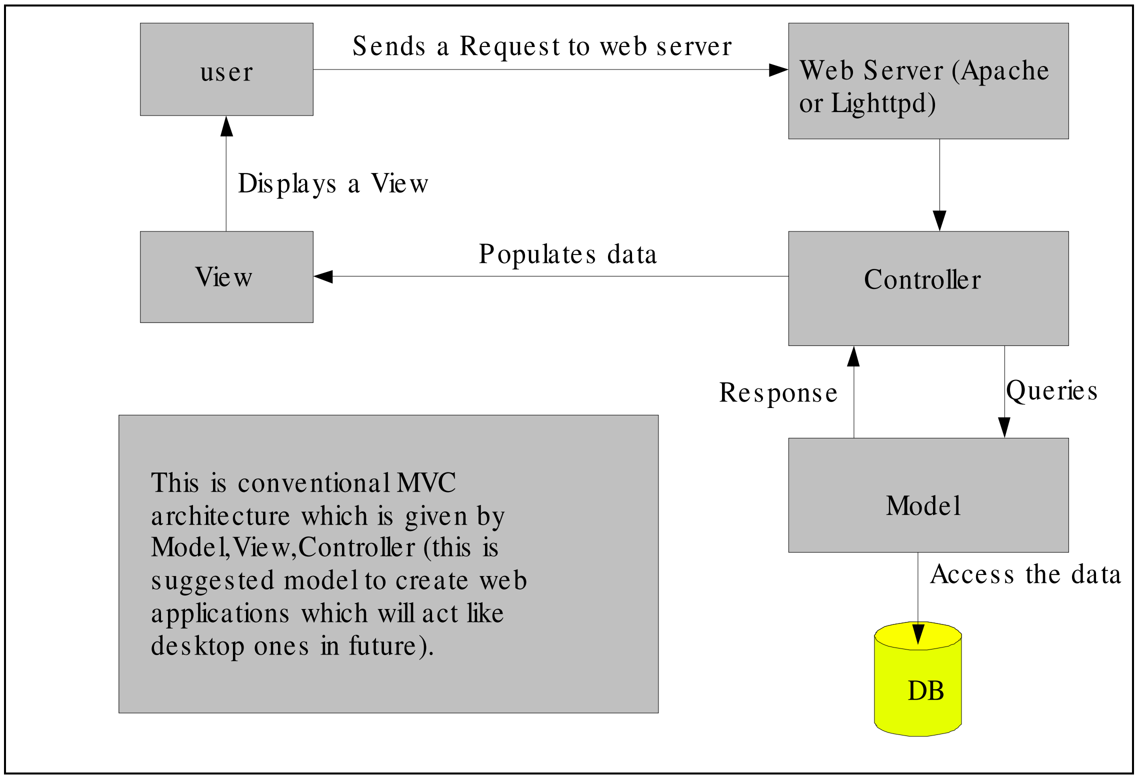

user
Sends a Request to web server
Web Server (Apache or Lighttpd)
Displays a View
View
Populates data
Controller
Response
Queries
Model
Access the data
DB
This is conventional MVC architecture which is given by Model,View,Controller (this is suggested model to create web applications which will act like desktop ones in future).

Figure 6.2: The MVC architecture and flow of control

6.3 Advantages of Ruby on Rails

Convention over configuration means an end to verbose XML configuration files in Rails; a Rails application uses a few simple programming conventions that allow it to figure everything out through reflection and discovery. For example, Rails uses intelligent reflection to automatically map database tables to Ruby objects. Your application code and your running database already contain everything Rails needs to know.

Rails programming conventions does more than just eliminate the need for configuration files. It also means that Rails can automatically handle myriad lower level details without you having to tell it to do so. This facilitates the writing fewer lines of code to implement the application. When fewer lines of code are used to develop an application it implies faster development and fewer bugs, which make code easier to understand, maintain, and enhance.

Rails use of runtime reflection and metaprogramming eliminates much of the boilerplate code that you would otherwise have to create. Using the built-in generator scripts to create most of the conventional stuff cuts the development time by many times. It is so easy to create a controller with just one line of running the script. This gives more time to concentrate on the logic of the application that need to be created instead of spending more time on the configuration to get a small thing running.

The typical development cycle for either creating or testing a change to a web app has steps such as configure, compile, deploy, reset, and test. This is time consuming and also a tedious process in most of the currently available web application frameworks. Rails combine all these steps again into one small command to regressively test and deploy the application. This reduces the turn around time to see changes where conventional frameworks could take a few hours to even show the out. Rails is so seamless that it does not even require a server restart to see the changes.

Rails can automatically create a full set of CRUD (Create, Retrieve, Update, and Delete) operations and views on any database table. This scaffolding can get you up and running quickly with manipulating your database tables. To start off with this could be a really quick way to show people how a base line of their application could look like. This option enhances the interactivity of the user at every level and introduces a spiral mode of software development where every change can be incrementally added on to the system with user feedbacks.

These are few of the many advantages that Ruby on Rails offers and hence this was decided to be an optimal platform on which the next generation of Panda monitor could sit comfortable. Also Ruby on Rails offers inbuilt adaptability to scripting packages such as Scrip.aculo.us and Prototype.js. Also the session management features in it made it a clear choice for the next generation of the monitor. All these factors made Ruby on Rails a clear choice for implementing the panda monitor.

6.4 MVC Panda monitor schema

The architecture for the Panda monitor was maintained to be in very close lines with that of the Ruby on Rails architecture of the Model View Controller, the schema for the Panda monitor based on MVC architecture will be as illustrated below in the diagram. The dashboard links and the links will each have a corresponding view. For example the link for the generic Panda Monitor will have a template view and each of the links like Production, Clouds, Jobs, and Analysis will have an individual view each. All these views have a corresponding controller to talk to; this setup allows interaction with the server from the controller and also between controllers when the user requests information. The controllers rely heavily on the models to interact with any required databases to fetch any information that may be needed to process the user request.

The models are the key to interact with the databases; the flow of control is as illustrated in diagram. The models are the only ones who can talk to the databases.

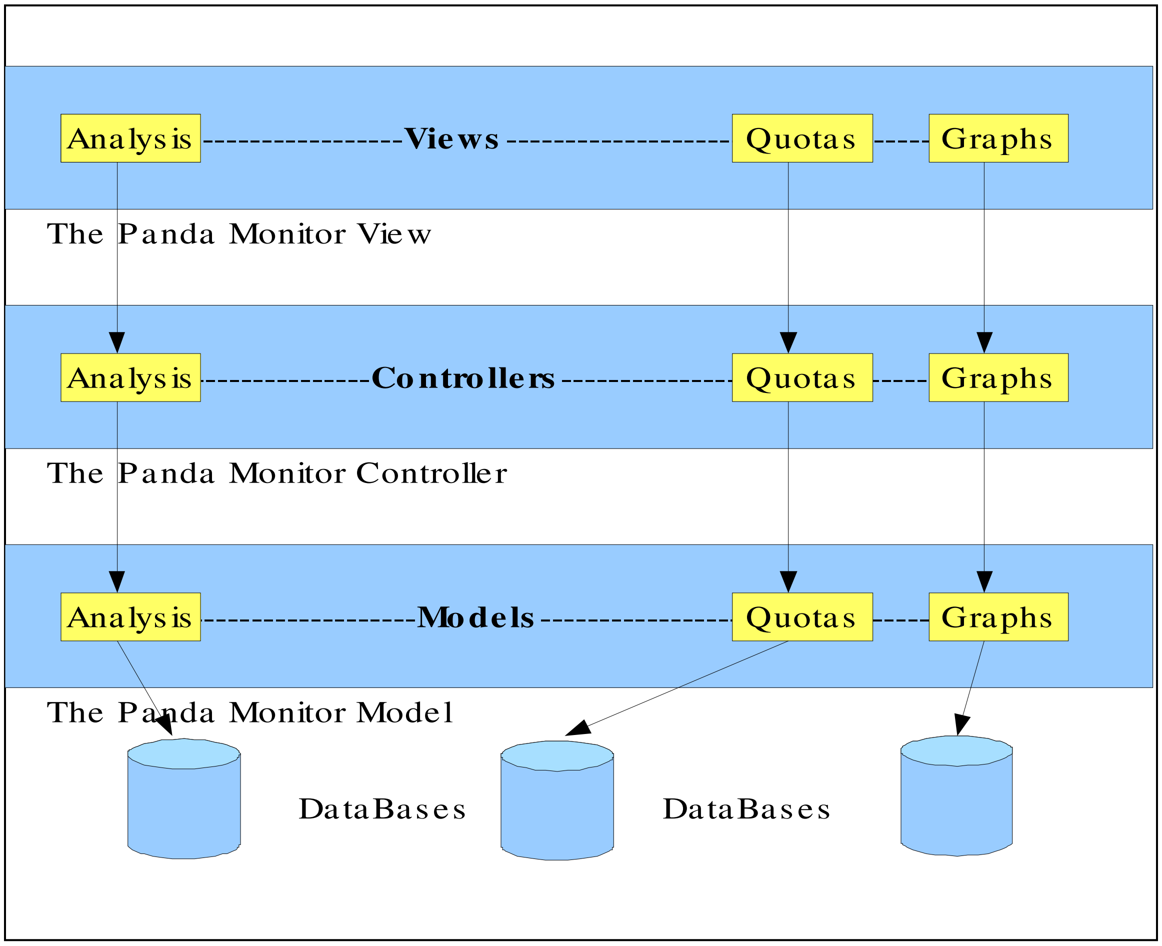


Figure 6.3: Scheme for the Panda Monitor in the MVC Architecture

They talk to the controllers and extract the required data from the databases and process them according to the user requests. The models provide a mapping to the objects in the

Ruby on rails paradigm to each of the tables. The clean structuring of control keeps context switching at its minimal. Context switch in software which is also referred to as process switch is the mechanism by which the CPU resource gets switched between processes or threads during the course of execution [].Context switches occur in the kernel space and the user space, it is quiet common with sloppy programming practices to introduce a heavy context switch in the user space. Ruby on rails avoids this by forcing a certain methodology of coding yet not making it too difficult for the programmer to conform to the standards.

The Panda monitor view is going to be very important part of the evolving architecture, unlike the current monitor view it is going to distribute and share with the controller and the model the load of presenting the data to the user.

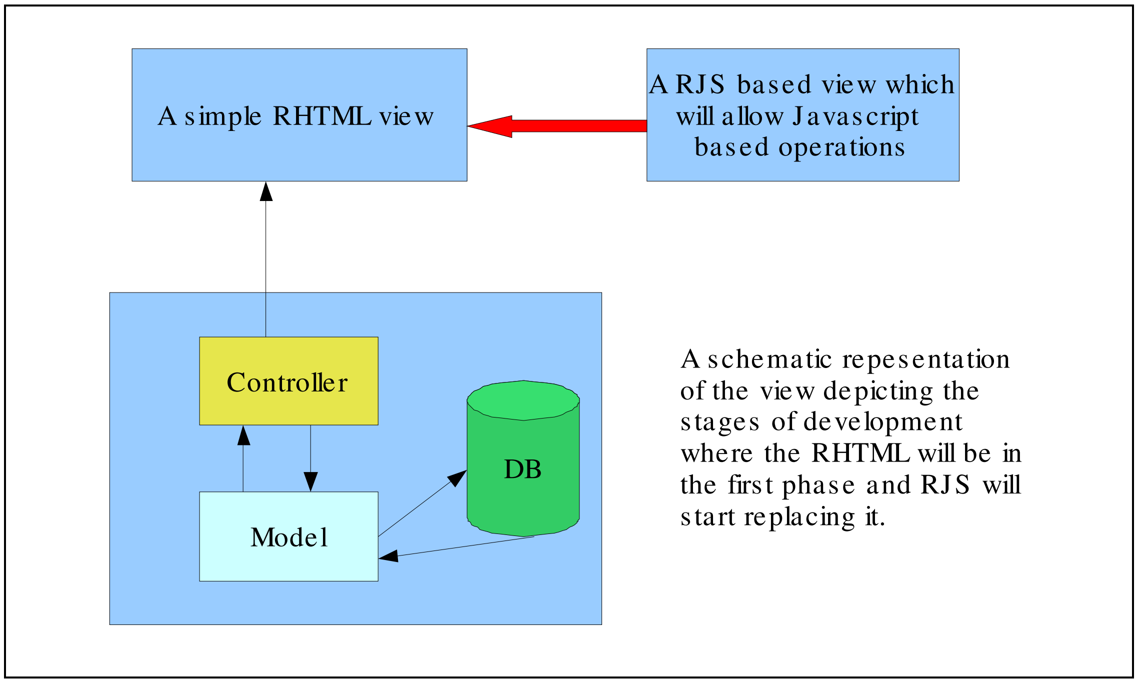

A simple RHTML view
A RJS based view which will allow Javascript based operations
Controller
DB
Model
A schematic repesentation of the view depicting the stages of development where the RHTML will be in the first phase and RJS will start replacing it.

Figure 6.4: Illustration of the View of the Proposed Monitor

The current monitor does not employ any other aided features like XML or JavaScript which can greatly improve its performance. A schematic representation of the proposed monitor view is given in the block diagram above. The initial phase of development for the monitor will just be a RHTML(Ruby Hyper Text Mark up Language).This unlike the normal HTML has added advantages by itself, also the fact that it can be seamlessly integrated with any additional plug-in that may be of use to improve the performance. RHTML is the standard in which the initial monitor. This initial setup will enable continuous user feedback and will allow the flexibility to add changes effectively. More over the page mapping scheme in Ruby on Rails is consistent with the controller mapping scheme and hence the pages are effectively mapped by means of simple naming conventions. [23].

A sample of the graph that was obtained in the new monitor scheme not only does it takes things fast. It also provides a wider variety of options.

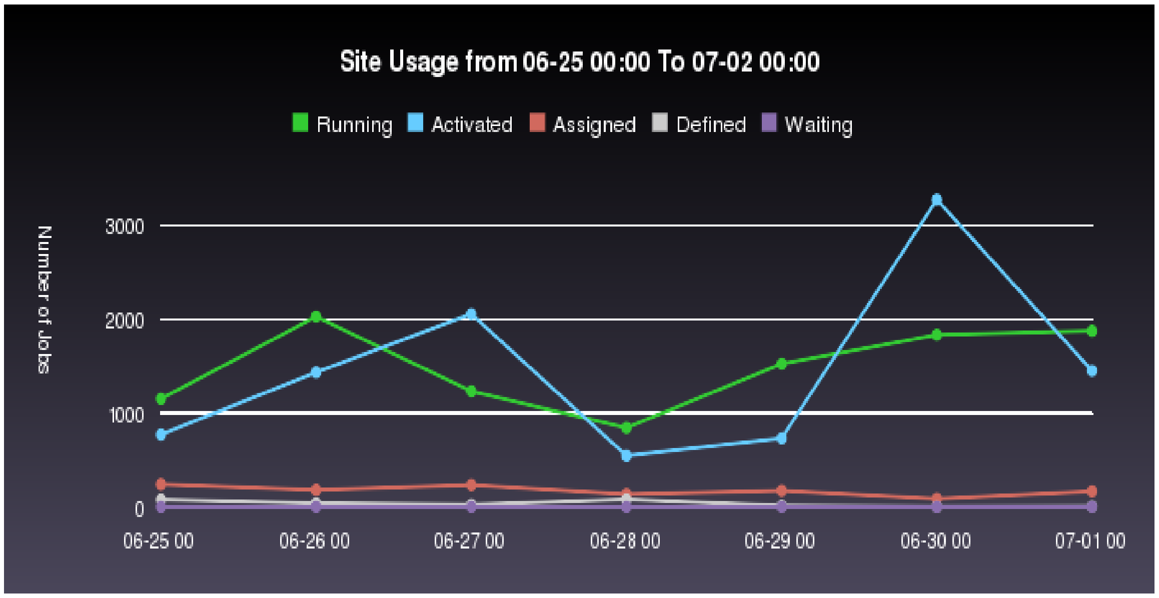


Figure 6.5: A sample Graph from the new monitor

CHAPTER 7

CONCLUSION AND RESULTS

7.1 Comparison of the two systems

The two systems can be compared in various levels to understand the effective working of the new system and the possible flaws with the earlier system. A simple parameter to compare could be the lines of code metric, this metric is considered just a simple metric which could give a high level overview of the system, though it does not reflect the effective working of the system in anyway, lines of code could still point to be an indicator that can give some good information about the whole system.

The old monitor code ranges easily around 6200 lines, and is growing at a much faster rate as new applications are being added everyday. The base framework that will be needed to bring a basic page online is near to 3400 lines in the existing monitor. In the new Ruby on Rails monitor that is being created this was cut down to less than 800 lines of code to bring a working monitor code for a graph generating application and was just around 580 lines to actually get the basic framework for the monitor up and running.

The next benefit that we immediately get by reducing the Lines of code heavily is that that code becomes easily maintainable. Software maintainability is defined as the ease of finding and correcting errors in the software. It is analogous to the hardware quality of Mean-Time-To-Repair, or MTTR.[24].Maintainability often cannot be directly measured in terms of numerical values but it can be roughly understood by finding the reliability of a software, this is mainly due to the fact that maintainability often associated with modularity, self or internal documentation, code readability, and structured coding techniques. These same attributes also improve sustainability, the ability to make improvements to the software.

For a system like Panda which is expected to evolve and be around for a number of years code maintainability becomes imperative. If software reaches a point after which it becomes too cumbersome to maintain or make changes it will eventually be replaced by another system.

The existing version of the Panda monitor could make it quite difficult for new code to be added at a later point in time when the experiment. This can be attributed the design since to even get something simple running entries have to be made into multiple files to let all the other components inside the monitor know that something new has been added. In the new version of the monitor this can be cleverly avoided, for any new application to be up and running there just needs to be a view, a corresponding controller and finally a model which can talk to the databases and make sure everything goes on smoothly.

This model will allow easy and custom applications to be made and added according to an individual's needs without much effort and extensive knowledge about coding.

7.2 Experimental Results

A number of pages were considered in the old monitor to check for typical response times or turn around times for a web page to turn up. The time ranges were from 4 seconds in a few pages up to 38 seconds in certain pages which dumped in more information to the user than actually necessary. A typical example that could be suggested is the Page for Analysis Jobs that has been shown in Figure 39, this page took an average of 26 seconds while it was queried at various time periods in a day ranging from low traffic to average traffic. Similar results were obtained for other pages too and the actual time to traverse through all the information in the currently displayed page was also huge for the user. Another place where the old system showed a great lag in response was in the graphs. The system responded quiet slowly for graphs that were even generated for a few days. The ideal time for graph generation for a period of 3 days at a sampling interval of 12 hours was around 14 seconds. The system took almost 24 seconds for the generation of a graph for a week at a sampling interval of 1 day. This delay in response can be clearly seen as an overhead and is a flaw with the way the old monitor was done.

To understand the impact Ruby on rails will make on the new monitor, this same sampling application was taken as benchmark and was recreated in Ruby on rails, the system scaled remarkably well and brought down response time to less than 3 seconds even for graphs that were generated for a month at a sampling interval of a day. The graphs for days less than a week were even faster, putting the response times between 1 and 2 seconds typically. Ruby on rails was found to bring remarkable reduction in turn around time in the page that was considered to be time intensive.

Graphs are plotted depicting the time that each monitor took for respective pages.

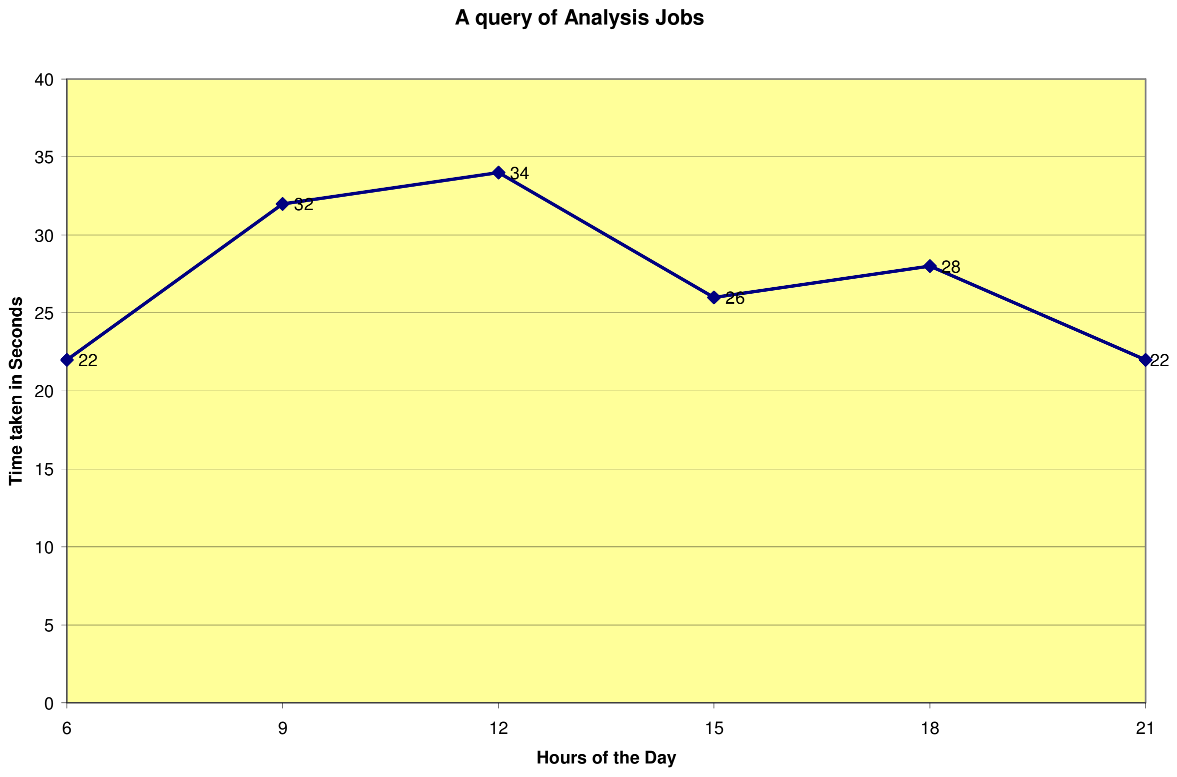


Figure 7.1: A plot showing time taken to display the analysis jobs page

The above graph is a representation of the time taken for the page containing the collective summary of the analysis jobs to be displayed in the current monitor which is in python. Samples were taken at varied times during the day to account for the traffic inflow to the monitor. It shows the simple query taking between 22 to 34 seconds at varied times during the day, this is also at a time when the total number of jobs submitted to the system ranging between 3000 -3500 jobs a day, whereas in real time when the experiment kicks off there will be easily around 200,000 jobs a day.

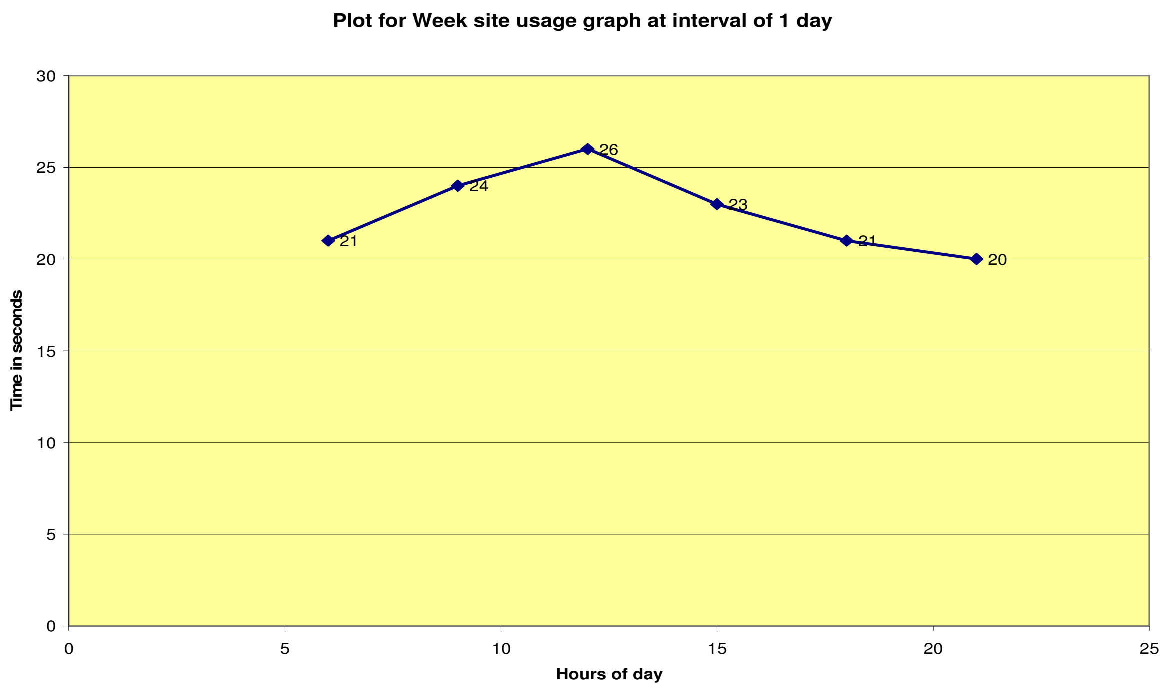


Figure 7.2: A graph showing time in seconds for a weekly plot

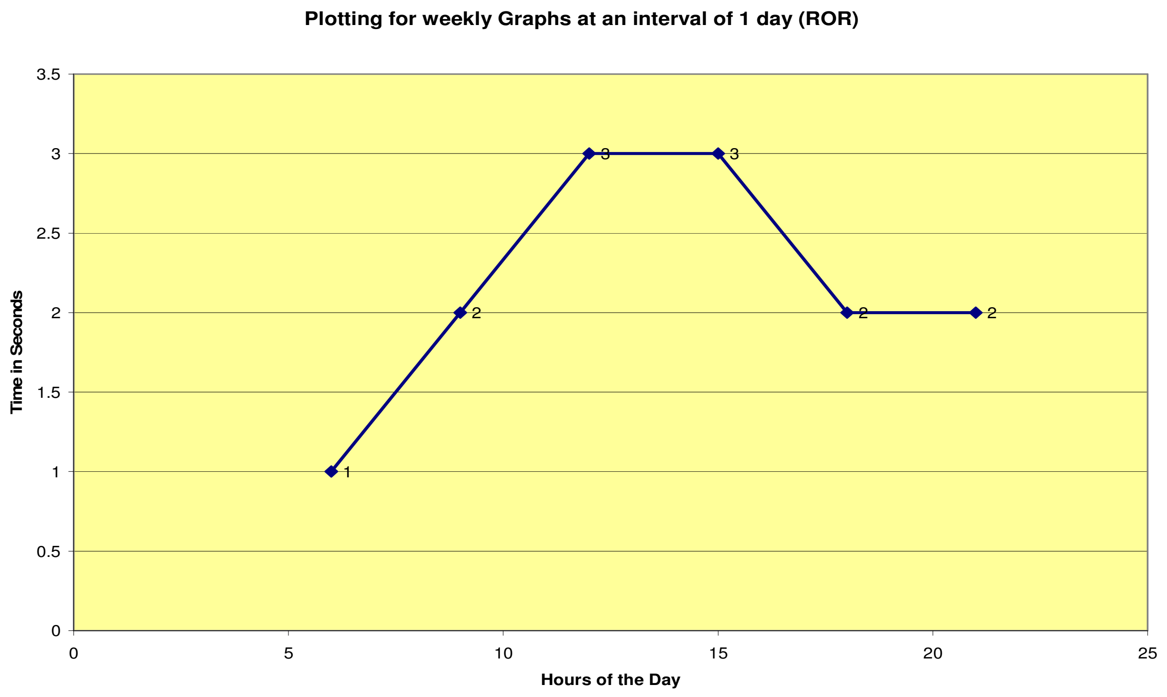


Figure 7.3: A graph showing time in seconds for a weekly plot in the new monitor

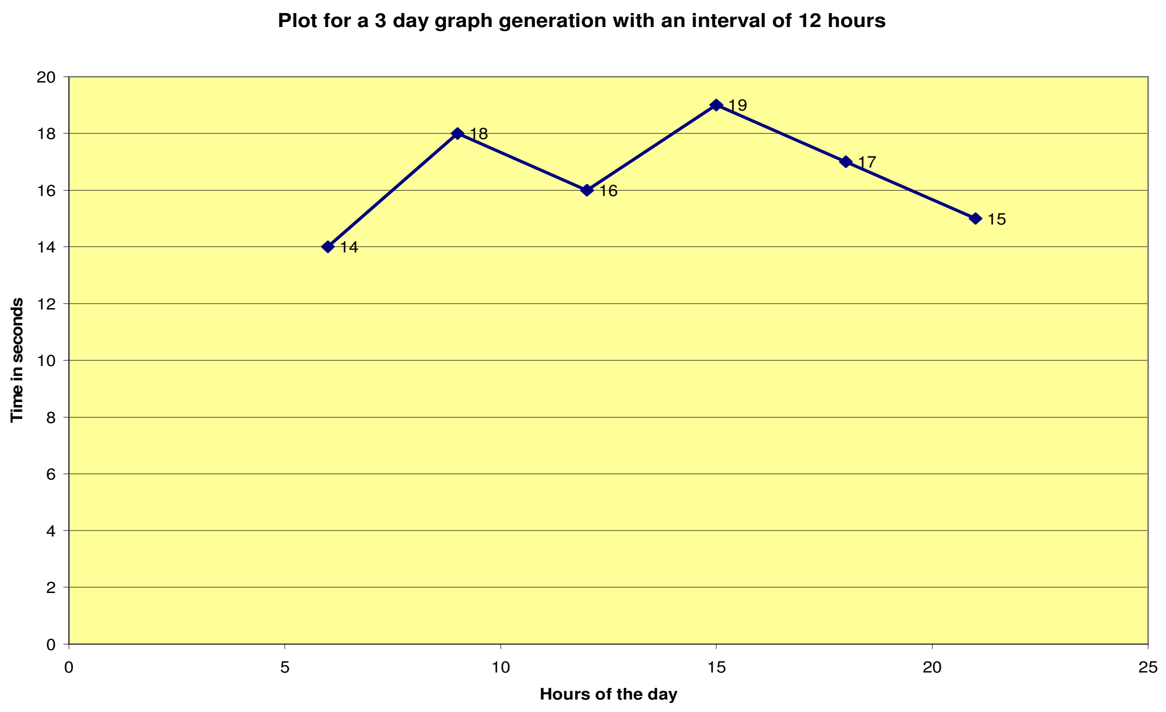


Figure 7.4: A graph showing time taken for a plot of 3 days at an interval of 12 hours

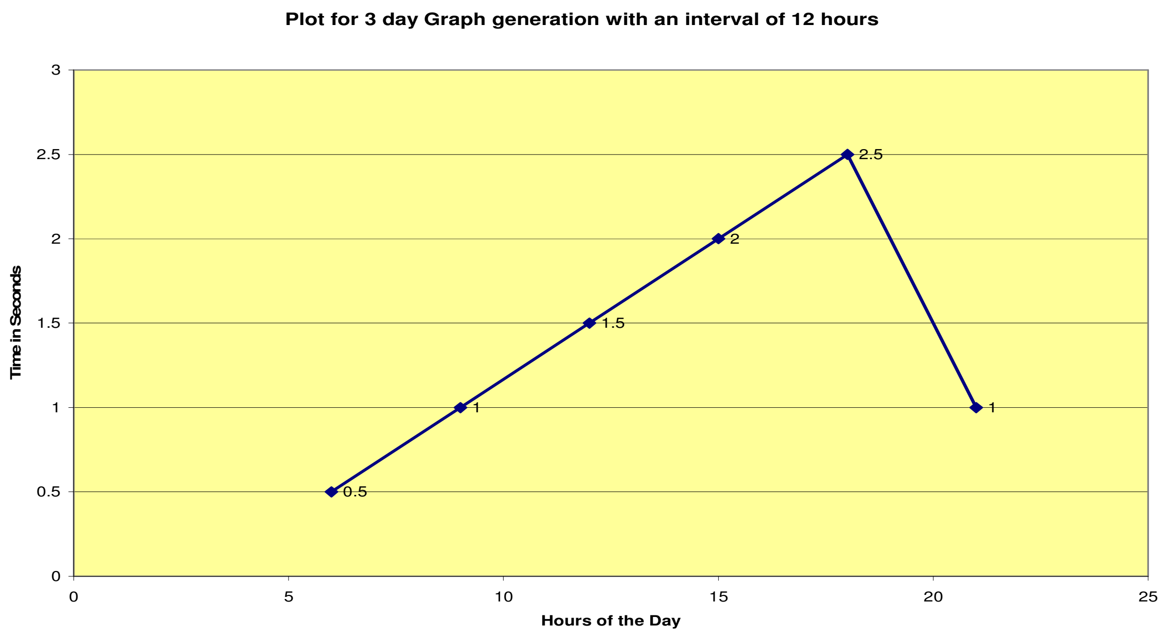


Figure 7.5: A graph showing time taken for a plot of 3 days at an interval of 12 hours-new monitor

The above two sets of plots are also drawn to illustrate the differences in performance between the two versions of the monitor for the plotter application which helps in retrieving history plots. The old version of the monitor can be seen taking much more time than the new version for the same query. Also it was noted that making updates to the application according to user requests was a cumbersome process.

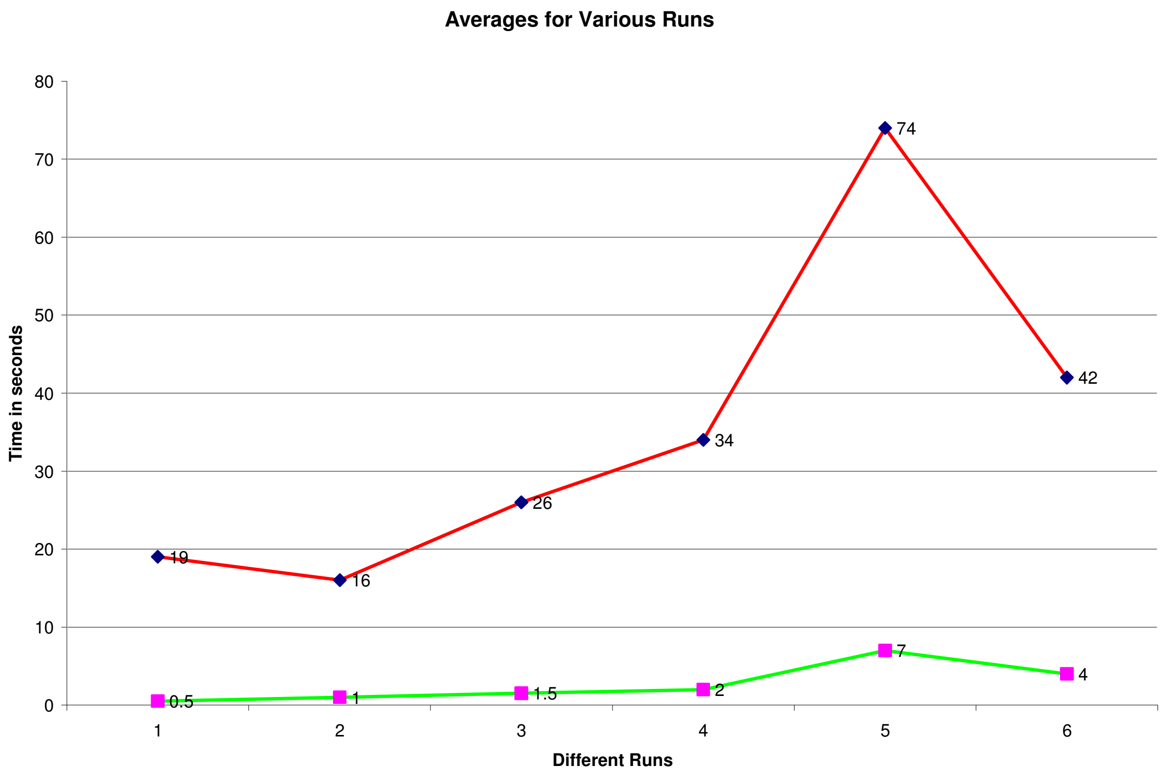


Figure 7.6: A plot showing average times recorded for various runs

The above graph was plotted in an attempt to see the performance of the new monitor against the existing monitor. The figure has two line graphs the green one representing the new monitor and the old in red.

The various scenarios that were run commonly in both the monitors are as below

1. Usage analysis across all sites for yesterday at an interval of 2 hours.

2. Usage analysis across all sites for 3 days at an interval of 12 hours.
3. Usage analysis across all sites for a week at an interval of 1 day.
4. Usage analysis across all sites for a week at an interval of 12 hours
5. Usage analysis across all sites for a month at an interval of 1 day.
6. Usage analysis across all sites for a specified date range an interval of 1 day.

The new monitor however was very good on the response time as it can be seen on the plots and for most queries it did not take more than 3 seconds to generate the graphs real-time. In the first version of the plotter application the monthly and weekly graphs were pre generated at midnight to be used for the day. This was due to the fact that some times these graphs took over a minute to be generated which is not acceptable.

7.3 Process Diagrams for the function history plots

In an attempt to elucidate clearly the advantages the new monitor will have in terms of context switches and design, an attempt is made below to give a view of the process diagram for the application called History plots in the Panda monitor.

The process diagram is a software engineering attempt to look at the applications in a better analysis view. Each of the important components that form these applications is given out and the functions that interact with each other are also given inside. The diagram can give a fairly deep understanding of the implementation and will also clearly give an idea of the context switches that will be involved in each of the systems

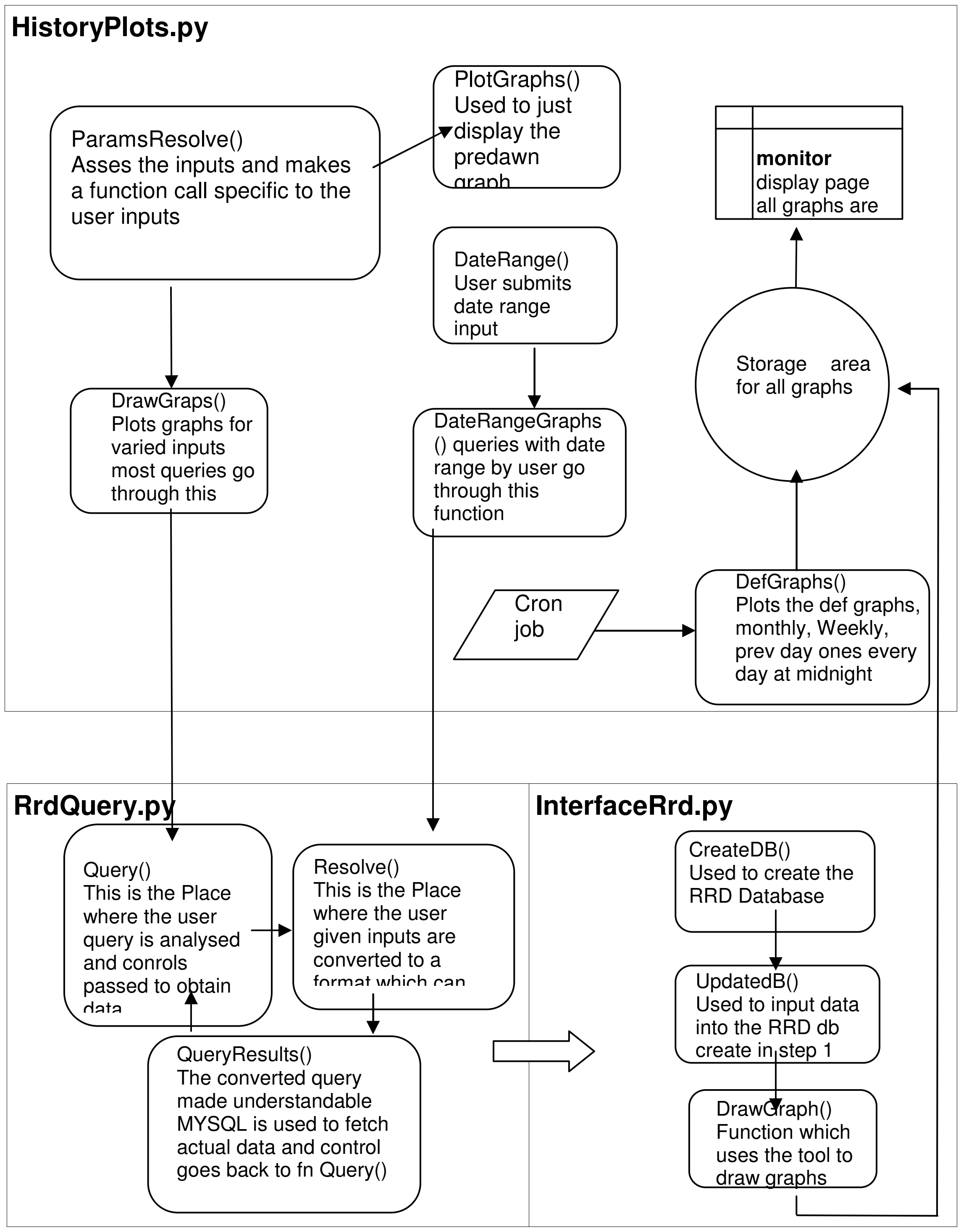

HistoryPlots.py
ParamsResolve() Asses the inputs and makes a function call specific to the user inputs
PlotGraphs() Used to just display the predawn graph
monitor display page all graphs are
DateRange() User submits date range input
Storage area for all graphs
DrawGraps() Plots graphs for varied inputs most queries go through this
DateRangeGraphs () queries with date range by user go through this function
Cron job
DefGraphs() Plots the def graphs, monthly, Weekly, prev day ones every day at midnight
RrdQuery.py
InterfaceRrd.py
Query() This is the Place where the user query is analysed and conrols passed to obtain data
Resolve() This is the Place where the user given inputs are converted to a format which can
CreateDB() Used to create the RRD Database
UpdatedB() Used to input data into the RRD db create in step 1
QueryResults() The converted query made understandable MYSQL is used to fetch actual data and control goes back to fn Query()
DrawGraph() Function which uses the tool to draw graphs

Figure 7.7: Process diagram of the existing monitor for History Plots

The above diagram clearly gives an overall picture of the flow of controls in the existing Panda monitor. In the application, the initial step is to obtain the inputs from the user using a user interface page. The inputs thus received, are passed on as parameters in the URL to the underlying code file. Based on the action to be performed on these parameters, control is transferred to the respective code module via Panda monitor overview module and request handler.

Let's consider the control flow in the case of graph generation for the history plots page of the Panda monitor. The critical file for this module is HistoryPlots.py; this interacts with the others to understand the user query and to draw the graphs. The prime responsibility of this file is to parse the parameters and format them to be sent to the RRD for further processing. In any application, the user inputs must be pre-processed before passing them onto the underlying code base and this has to be handled at the upper layer. The function params resolve () aids in categorising the inputs and formatting them. Since graphs can be generated depending upon various inputs (date range, site types), each of it has to be handled separately. For the date range inputs, the function date range () is used. drawgraphs () function routes all these calls to the rrdquery.py, which acts as an abstract layer and accepts all the graph generation invocation. In this file, function query (), the parameters are separated and each of it is appropriately placed in the SQL query. The function query results () is where the actual data from the database are read using the above constructed SQL. The results thus obtained are passed to the interfacerrd.py file. At this juncture, the RRD database is created using the results in the function createdb (). Function drawgraph () invokes the RRD tool, and the graphs are generated. The graphs thus generated are stored at a specific location. Finally, these pre-drawn graphs are displayed using the function plot graphs (). This in turn returns an html page, which is displayed to the user.

It is evident that this existing functional approach has a heavy context switching involved. The function calling is done at several layers, passing the parameter down to RRD.

Since, context switching is directly proportional to the response time for any system, this heavy context switching adversely affects the response time and hence we observe long response time in the existing Panda Architecture.

7.4 The Proposed Architecture

In the Rails framework, the functional layers are clearly distinguished by the framework itself. This framework broadly classifies these layers as Models, Views and Controllers.

Each of these entities has a set of specific actions to be performed. As the number of entities involved in the proposed architecture is less compared to the existing Python architecture, we can expect the context switching to go on the lower end, resulting in the quick response from the new system.

Let's consider the functional flow for the graph generation module in the new architecture. The figure below is an illustration of the functional sequence in the new Ruby on rails architecture.

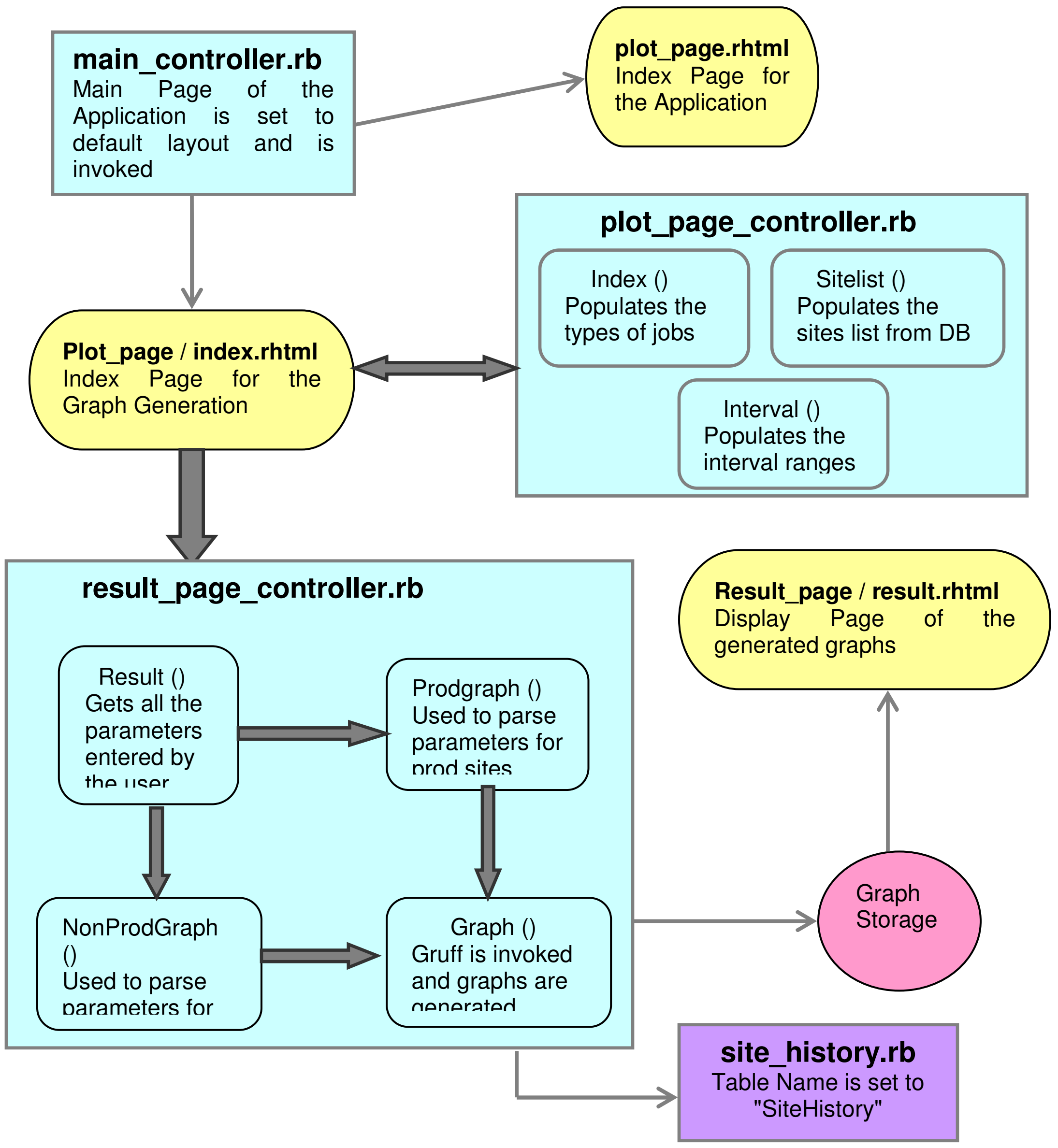

main_controller.rb
Main Page of the Application is set to default layout and is invoked
plot_page.rhtml
Index Page for the Application
plot_page_controller.rb
Index ()
Populates the types of jobs
Sitelist ()
Populates the sites list from DB
Interval ()
Populates the interval ranges
Plot_page / index.rhtml
Index Page for the Graph Generation
result_page_controller.rb
Result ()
Gets all the parameters entered by the user
Prodgraph ()
Used to parse parameters for prod sites
NonProdGraph ()
Used to parse parameters for
Graph ()
Gruff is invoked and graphs are generated
Result_page / result.rhtml
Display Page of the generated graphs
Graph Storage
site_history.rb
Table Name is set to "SiteHistory"

Figure 7.8: Flow of controls between functions in the new architecture

In Figure 47, the entity (site_history.rb) in purple is the model for this graph generation module. This model is used only when the code contacts the database. The entities in yellow are the views. These views have fillers which are filled in by their respective controllers, when they invoke the views to be displayed to the user. The entity in pink denotes a specific physical location in the system, where the generated graphs are stored before they are displayed to the user. The entities in light blue are the controllers. These controllers form the crux of the whole new architecture. They manage the control flow and also aid in providing the necessary back end support for their respective views.

In this case, main_controller.rb controller aids in displaying the index page for the whole application. The view plot_page.rhtml is the index page that is initially displayed. From this index page, links are provided which enable us to access the different features of the Panda monitor. The view plot page / index.rhtml will be invoked, if the graph generation module is chosen in the index page. This view is aided by its controller plot_page_controller.rb; this controller has functions which populates the types of categories in the graph generation page. After the user chooses the necessary categories for the graph generation, the controller result_page_controller.rb is invoked. The parameters entered by the user are accessible in this controller. These parameters are formatted and categorized in the same controller. The function graph () takes in these formatted parameters and uses gruff tool to generate the necessary graphs. These graphs are stored in a temporary location before they are picked by the result_page / result.rhtml view to display it to the user.

From the above sequence of actions it is apparent that the number of context switches is less compared to the old architecture. This is the advantage of using the Rails framework over the Python code base.

7.5 Future Work

The future work in the monitor could involve in making the response times even faster by bringing the whole monitor into the AJAX paradigm by directly using the inbuilt libraries in the Ruby on rails framework. This will make the monitor feel more like a desktop application. New applications need to be developed, which will clearly marking out between analysis and production jobs in separate views. The newer version can also include a better navigation scheme for the interface since the current system has more than necessary information in one single page.

These views have to be more carefully designed to make sure they do not contain more information than what the user is looking for. This is the case with the current monitor a whole page with excessive information is built in one generation request. The enhancement could involve a development of a memory and network usage monitoring as an integral part of panda monitor itself, rather than the separate monitoring that is now provided by Nagios. The Monitor should also include a secure session management for the user who logs in from the same system. Currently the monitor does not include any security model, the versions that are to be later shipped at a later stage should include a user login and session tracking model.

REFERENCES

[1].The first commercial computer http://en.wikipedia.org/wiki/UNIVAC_I

[2].Paul Laskowski, John Chuang-Network Monitors and Contracting systems: Competition and innovation

[3].IBM: Red Books (Introduction to Grid Computing)

[4].Dieter Wybranietz and Dieter Haban- Monitoring and performance measuring distributed system during operation.

[5].http://en.wikipedia.org/wiki/Grid_computing

[6].Ian Foster, Carl Kesselman, Steven Tuecke: The Anatomy of Grid-Enabling Scalable Virtual Organizations

[7].LHC Website. http://lcg.web.cern.ch.

[11]. Emir Imamagic, Dobrisa Dobrenic - Grid Infrastructure Monitoring System Based on Nagios.

[2].Larry Roberts, Multiple computer networks and inter computer communication (ACM Symposium on Operating System Principles. October 1967)

[3].Jean Francois Roche: Grid and Cluster Monitoring-Universite Libre De Bruxelles, Universite D'Europe

[12].Nagios -http://nagios.sourceforge.net

[13].The Monalisa Monitoring system-http://monalisa.cacr.caltech.edu/monalisa.htm

[14].The Lemon monitoring system- http://lemon.web.cern.ch/lemon/index.shtml

[15].Abishek.S.Rana- A Globally Distributed Grid monitoring system to facilitate high-performance computing at D∅/Sam grid.

[16]. GridIce. http://gridice.forge.cnaf.infn.it.

[17]. EGEE - Enabling Grids for E-science. http://public.eu-egee.org.

[18]. Andreozzi, S., Bortoli, N. D., Fantinel, S., Ghiselli, A., Tortone, G., and Vistoli, C. Gridice: a monitoring service for the grid, 2003.

[19]The Monalisa Monitor-http://monalisa.cacr.caltech.edu/img/

[20] Relational Grid Monitoring Architecture. http://www.r-gma.org.

[21] Al-Shaer, E. S. High-performance monitoring architecture for large-scale distributed systems using event filtering, 1997.

[22]. Cottrell, R. L. Passive vs. active monitoring, 2001.

[23].CPU Time- http://www.hlrn.de/doc/performance/glossary.html

[24].NASA Software Engineering Page: http://satc.gsfc.nasa.gov/assure/agbsec4.txt

[25] Baranovski, A., Bertram, I., Garzoglio, G., Lueking, L., Terekhov, I., Veseli, S., Walker, R. *SAM-Grid: Using SAM and Grid middleware to enable full function Grid Computing*.

[26] The AMANDA-http://butler.physik.uni-mainz.de/amanda/homepage/index.html

[27] The SAM GRID- http://www-d0.fnal.gov/computing/grid/

[28] The Panda Twiki- https://twiki.cern.ch/twiki/bin/view/Atlas/Panda

[29] Panda: US ATLAS Production and Distributed Analysis System-Xin Zhao Brookhaven National Laboratory

[30] Context Switching- http://www.linfo.org/context_switch.html

[31] The ATLAS PANDA-Production and Distributed Analysis system- Torre Wenaus- http://www.rhic.bnl.gov/RCF/UserInfo/Meetings/Technology/Archive/Mar-13-2006/200602-panda-tech.pdf

[32] The PANDA DDM integration-Torre Wenaus ,Tadashi Maeno - https://twiki.cern.ch/twiki/bin/viewfile/Atlas/PanD-ddmbnl-panda-ddm.pdf

BIOGRAPHICAL INFORMATION

Prem Anand Thilagar received his Bachelors in Computer Science and Engineering from the Anna University, Chennai, India in 2005. After working at a software company in India for 6 months; he started his graduate studies at the University of Texas at Arlington, USA in spring 2006. He received his Masters of Science in Computer Science in December 2007.