Data-oriented scheduling for PROOF

Neng Xu1, Wen Guan, Sau Lan Wu
University of Wisconsin, Madison, WI, USA
Gerardo Ganis
CERN, 1211 Genève 23, Switzerland
E-mail: {Neng.Xu, Gerardo.Ganis, Wen.Guan, Sau.Lan.Wu}@cern.ch

Abstract. The Parallel ROOT Facility - PROOF - is a distributed analysis system optimized for I/O intensive analysis tasks of HEP data. With LHC entering the analysis phase, PROOF has become a natural ingredient for computing farms at Tier3 level. These analysis facilities will typically be used by a few tenths of users, and can also be federated into a sort of analysis cloud corresponding to the Virtual Organization of the experiment. Proper scheduling is required to guarantee fair resource usage, to enforce priority policies and to optimize the throughput. In this paper we discuss an advanced priority system that we are developing for PROOF. The system has been designed to automatically adapt to unknown length of the tasks, to take into account the data location and availability (including distribution across geographically separated sites), and the {group, user} default priorities. In this system, every element - user, group, dataset, job slot and storage - gets its priority and those priorities are dynamically linked with each other. In order to tune the interplay between the various components, we have designed and started implementing a simulation application that can model various type and size of PROOF clusters. In this application a monitoring package records all the changes of them so that we can easily understand and tune the performance. We will discuss the status of our simulation and show examples of the results we are expecting from it.

1. Introduction
The Parallel ROOT Facility, PROOF [1], is a distributed analysis system optimized for I/O intensive analysis tasks of HEP data, addressing the (quasi-) interactivity needs of end-user HEP analysis. At LHC, PROOF is becoming a natural ingredient for computing farms at Department or University level (Tier 3), i.e. those dedicated to end-user analysis. End-user activities in the Tier3 farms consist of data mining, (toy)-simulation, fits, etc., with data coming mainly from the associated Tier2 or Tier1 sites. Typically the Tier3 sites cannot afford to provide only a PROOF service; in fact, as all interactive systems, a PROOF cluster works better when the resources are not fully loaded, so that, in average, the occupancy of a pure PROOF cluster is not 100%. In order to fill the occupancy gaps, in particular during non-working hours, a standard batch system is typically available, which runs less-chaotic jobs with lower priority. The combination of batch and PROOF services allows to use the computing farm more efficiently and also gives users more flexibility in design their computing strategy [2].

1 To whom any correspondence should be addressed.

Published under licence by IOP Publishing Ltd
1.1. Multi-layer storage systems

The typical configuration of Tier3 farms consists of $O(100)$ cores and will typically be used by $O(20)$ users. These farms have typically a multi-layer storage system, with a fair amount of fast local "storage" (RAM and possibly Solid State Disks, SSD), and $O(50)$ TB of standard storage, possibly connected to a local MSS and the ability to access or download any required data via the Grid. Figure 1 shows different prototypes of multi-layer storage system. The input data is stored or replicated at different layers. The fast storage layer can provide much better data stream but is limited in size. The slow layers have enough capacity for multiple copies of the data. In most of data analysis scenario, not all the files are needed for multiple copies of the data. In most of data analysis scenario, not all the files are needed for all user tasks, and the least used files can go on the cheap and slow storage layers. Files also have multiple copies at different layers so if they are missing at one layer they can always be found at lower layers. The fast access speed, integrity and availability of input data are very crucial for those data analysis jobs.

![Figure 1 Different types of multi-layer storage systems](image)

1.2. The problem of overloading resources

Multi-layer storage systems are promising solution for Tier3 facilities. However, to work properly in all situations, in particular when the size of the farm increases or the amount of input data becomes huge, a good coordination among all the active components of the system is required. Resource overloading may in fact occur at different levels:

- **Memory**
  Depending on the type of processing algorithms and activity, data analysis jobs may need a lot of RAM. Concurrent jobs may therefore force the operative system to heavily use the swap space making the whole system reacting very slow or even irresponsive.

- **Hard drives/RAID arrays**
  Jobs reading and/or writing concurrently can interfere negatively, making the effective bandwidth smaller than it is supposed to be, and the storage component irresponsive.

- **Network**
  The available network bandwidth may be saturated by the concurrent requests of data-analysis and file-movement jobs. This may slow down communications between processes, leading to timeout conditions that may be difficult to handle correctly, screwing up job completion.
1.3. Scheduling at Tier3

Scheduling is mainly need to avoid congestions and overloading, and to optimize resource usage. Batch systems have their schedulers, usually taking into account CPU and memory. The default policy is based on the First-Come-First-Served (FCFS) paradigm [3].

PROOF provides an abstract interface for scheduling with a plug-in mechanism. Currently, also in the case of PROOF, only an implementation of a FCFS policy is available. However, PROOF has dynamic scheduling embedded allowing to micro-optimize the usage of the assigned resources once the job is scheduled.

The question to be answered is if the available scheduling options are enough to cope with the requirements of the analysis farms. If we take the two approaches alone then we see that the situation is reasonably under control for what relates to avoiding congestions. Given the chaotic nature of short, prototyping jobs typical of analysis activities, resource optimization seems a difficult ask for batch systems, while the embedded dynamic scheduling available in PROOF is definitely better suited for that.

In mixed systems, however, the underlying batch activities may lead to severe congestions if PROOF and batch jobs are not controlled by the same scheduler. This was observed, for example, by the ATLAS group at Univ. of Wisconsin, where low-priority, long file transfer jobs were saturating the network and disk resources, significantly slowing down high priority PROOF jobs.

We believe, therefore, that there is the need for a new scheduling strategy. The remaining of this paper is devoted to describe the ideas we have in mind to improve on the current situation. In Sect. 2 we describe the main ideas of data-oriented scheduling, focusing in particular on the problem of overloading, the system components and their relationship, and the priority assignment. In Sect. 3 we discuss the current status of the simulation program and show some first results. In Sect. 4 we present briefly some ideas for an integration with PROOF. Finally in Sect. 5 we conclude.

2. Data oriented scheduling

Data oriented scheduling system is adding a higher-level control on top of PROOF systems. This system includes storage elements, network and their specifications, for example - the available ‘bandwidth’, to the scheduling metrics. Different from other scheduling system, we add priority for the data and storage elements. The file movement control is also part of the scheduling.

2.1. The main functions

The main goal of this scheduling system is to not overload the storage system and the network. The second goal is to minimize the data movement among the different storage components. In order to achieve its two goals, the main tasks for the scheduling system are following:
- for each waiting job, choose an adequate fast storage and preload the files to the chosen layer;
- pre-locate and validate the files for the waiting jobs to minimize job failures;
- reorganize the startup sequence for waiting jobs based on their priority;
- balance the cluster usage (both in CPU and storage) between users and groups;
- assign reasonable {CPU, storage, network bandwidth} to a job.

2.2. The components and the bandwidth

The components of the cluster taken into account by the scheduling system are the following:

| Component  | Description |
|------------|-------------|
| System     | The whole cluster, including GRID storage, if any; |
| Job        | Data-Analysis or File-Movement job |
| User       | Person submitting jobs |
| Group      | A set of users |
| Dataset    | A set of files to be processed |
| Storage    | Different storage layers (RAM, SSD, HDD, RAID, tape, GRID) |
| Job slot   | CPU core + RAM |
The bandwidth of some of the components is also taken into account. For example, the storage components provide a maximum bandwidth depending on the hard-drive controllers capabilities and the available network bandwidths and latencies. These determine the effective bandwidth between different storage layers. Each job has some optimal bandwidth requirements. By matching the job bandwidth requests with the component bandwidth, the system can avoid the overloading of the network and storage resources. When the load on certain component reached the maximum available bandwidth it, the scheduling system will stop starting more jobs needing access to the component.

2.3. The Priority
Batch scheduling systems usually assign priorities only to users, jobs and job slots. One of the main ideas in data oriented scheduling is to assign priorities to all components in the system. There is either a direct or indirect relationship between two components. For example, users belong to groups and jobs belong to users; so there is an indirect relationship between jobs and groups. Datasets belong to groups and storage components, but are requested by users, so their priority will indirectly depend also on the user priority. The groups also own job slots and have quotas on the different storage components: the priority of those components therefore depends also on the group priority.

Based on the main components and their relationship, the scheduling system adds the priority on top of them. Each component is consists of its own priority and the priority of the component which it belongs to or is in relation with. When the job is running or waiting, the priority of relevant components is increased or decreased with time. Ordered sequences can then be easily organized by comparing priorities.

2.4. Job management
The scheduling system manages the analysis jobs and staging jobs based on their priority and the load of the bandwidth. For those running jobs, including analysis jobs and file movement jobs, the system calculates the bandwidth they occupy on the disk and network and use this information to determine if those components are overloaded or not. It also calculates the available bandwidth and decides when to release the next job. In this way, the job with highest priority may not be released or partially released if there is no enough bandwidth available.

3. Simulation system

3.1. Purpose
Since the scheduling system is still under development, how to tune the system performance is an unclear issue, like how fast the priority should drop or increase. A simulation system was developed to study different cases from small cluster to big ones. From this simulation system, one can monitor the change of priority of each component to see if it moves as expected. It also monitors the traffic load on the disk/RAID, network and between the storage layers. The waiting time and running time of each job are also recorded for the study. The parameters can be tune in this simulation system for the optimization

3.2. Users’ activity studies
Based on our past experience from different universities and labs, some basic assumptions were made for the simulation of the user activity as follows:

- the number of jobs submitted at the same time by each user is small and close to unity;
- the time between two job submissions is similar to the execution time; we used 20 min;
- the size of the input datasets is not fixed a priori;
- jobs can be both CPU and I/O intensive, with very different CPU needs;
- jobs are rather short, in the range 0.5 to 10 minutes;
- input datasets are used repeatedly;
users in the same group read similar datasets;
- new datasets are used more often than old datasets;
- job submission happens only during working hours.

With these assumptions, we can define different scenarios to test and study the scheduling system.

3.3. Virtual clusters
The real tests are running on virtual clusters. Virtual clusters are defined in a text based configuration file. An example configuration is the following:

- Check period: 5 seconds
- Job length: 0.5 to 10 mins
- File size: 200 MB
- Dataset size: 10 to 1000 files
- Inter-submission time: 5 to 60 mins
- Number of datasets: 1000
- Number of servers: 5 x 8 job slots per server
- Size of RAM: 16 GB per server (2 GB / core)
- Size of HDD: 5 TB per server
- Space on tape: 500 TB
- NIC bandwidth: 1 Gb/s
- Group 1: 5 users
- Group 2: 7 users
- Required job bandwidth: 25 MB/s (in average)

The simulation system reads this configuration file and simulates the users’ activities with different system perimeters. In those virtual clusters, there are some details not yet added in at this stage. One of them is the file distribution on the file servers. In the system, we assume that each server contains same number of files in a dataset but this is not always the case in the real cluster.

3.4. System performance study
There are many indicators can be used to judge the system performance. The waiting time for the job is an important one especially for users’ interactive jobs because users always want their jobs to run immediately after the submission. The average waiting time is more concerned than individual jobs in the system. Some jobs may request big input dataset or many input datasets. In this case, they may have to wait long time for the file to stage-in. To reduce the average waiting time is a major task in this study.

The running time for the jobs is another important indicator to study. Without overloading the system, the running jobs should always maximally use all the available bandwidth. If there is a bottleneck in the system, the running time will be affected. Therefore, the average running time of the jobs has to be also well studied.

The data transfer load on the network among different layers is the indicator to judge the file movement control part in the scheduling system. The file movement uses the stage-in jobs. When those jobs are running, part of the network and disk bandwidth is occupied so some of the job slots have to be stopped because there is no available bandwidth for them. Therefore, frequent file movement will affect the system performance. The decision making for the file movement between storage layers has to be carefully studied.

We are still working on finding a reasonable balance point to control the raise/drop of the priority for each component. So far our study are mainly focus on three indicators for which we show examples in the next section.
3.5. Example results
The indicators that we have studied so far are the priority evolution of groups and users, the job waiting time and the job running time.
In Figure 2 the priority evolution with time of two groups of users is shown. These two groups have different priority and the users priority is affected by it. This test is only simulating the activities during the working hour therefore we see a continue priority decrease. The overall priority will move back during the none-working hours. We are also considering using a different priority control for those none-working-hour activities in order to provide better fair share of the resources.

![Figure 2 Example of time evolution of group and user priorities for two groups. See text for details.](image)

From Figure 3, one can see most of the jobs have a reasonable waiting time, about 30 seconds. Some jobs have longer waiting time because they were waiting for input datasets. In this test, we have treated new and old datasets at the same strength. Therefore we see much more data stage-in activities in this test. In the real case, since users are more interested in the new data than the old data, we shall see fewer jobs that are affected by the file movement.

![Figure 3 Example of job waiting time](image)

From Figure 4, we can see that all the jobs finish in an acceptable time range. This means there is no obvious bottleneck found in this system configuration. In this test, we did not allow job to use multiple datasets and this will be added in our future tests. We have foreseen the problems when some users submit jobs with large amount of datasets. This kind of jobs will need an alternate queue to deal with them.
The above results are just some examples how we study the system performance and how to find the potential problems. There are still lots of tests need to be done and more test methods need to be used.

4. Integration with PROOF
As mentioned above, controlling and optimising the usage of the available resources requires that all activities, pure batch and PROOF, are controlled by the same scheduler. In addition, the new data-oriented scheduler requires control of the file-movement jobs based on information available only in PROOF.

The integration with PROOF, therefore, needs to go via the abstract scheduling interface provided by PROOF. The plug-in will consist of two main components: the PROOF interface, following the PROOF abstract interface to scheduling, and an interface to batch staging system.

The full implementation of the new scheduling plug-in will require some addition in PROOF itself. In particular, an important ingredient is the possibility to check-point a running session and to resume a check-pointed session; this is needed to setup the right dependencies based on the dataset requirements.

5. Conclusions
By studying different cluster structures, we have seen many potential problems in the existing systems and those problems cannot be fully solved on the batch system side. We realized that management of I/O intensive farms in multi-group and multi-user environment requires to include bandwidth control in the job scheduling system. Especially for the cluster with multi-layer storage system, the file movement control can be done together with scheduling system. A simulation program is being developed to study the various cases and find the possible working points. The system is designed for small Tier3 site and also for big data analysis centre that is mainly used for those I/O intensive data analysis jobs. Although the current development is focused on PROOF-enabled farms, the concepts apply also to fully batch-based systems, provided that the batch system offers the possibility to define dependencies between jobs in a flexible way, like, for example, in Condor [4].

6. References
[1] http://root.cern.ch/drupal/content/proof
[2] Several examples exist of such farms. For example, the contribution #89 at this conference, PROOF on a Batch System, by W. Ehrenfeld et al., describes a system using SGE as batch system; the ATLAS group at Univ. of Wisconsin has a farm based on the Condor system [4].
[3] See, for example, Y.Etsion, D.Tsafrir, A Short Survey of Commercial Cluster Batch Schedulers. Technical Report 2005-13, School of Computer Science and Engineering, The Hebrew University of Jerusalem, May 2005.
[4] See, for example, D.Thain, T.Tannenbaum, and M.Livny, "Distributed Computing in Practice: The Condor Experience", Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, pages 323-356, February-April, 2005, and references there in.