MODELLING AND SIMULATION OF THE CMS TIER0 INPUT BUFFER WITH YASPER

L.H.J.M. Wijnant[1,2], A.T.M. Aerts[1,2], N. Sinanis[1], I.M. Willers[1]

[1]CERN, Geneva, Switzerland
[2]Eindhoven University of Technology, Eindhoven, The Netherlands

wijnant@gmail.com (Arnaud Wijnant)

Abstract

The CMS (Compact Muon Solenoid) experiment at CERN will produce large amounts of data in short periods of time. Because the data buffers at the experiment are not large enough, the data needs to be transferred from the experimental area to the multi-tier computing system for storage and processing. The first tier is the CMS Tier0, an enormous job processing and storage facility at the CERN site. One part of this Tier0, called the Tier0 input buffer, will have the task to readout the experimental data buffers. It has to make sure that no data is lost. This paper describes the modelling and simulation of the Tier0 input buffer to compare different scenarios involving a set of disk servers that can accomplish the Tier0 input buffer tasks. To increase the performance per disk server, write and read actions on the same disk server take place in separate phases. A critical issue then is to determine when a disk server should change from accepting and writing items to supplying items to other tasks. The combination of various parameters, such as the usage of a particular queuing discipline (like FIFO, LIFO, LPTF and SPTF) and the state of the disk server has been studied. We have used Yasper for modelling and simulation of the various scenarios. Yasper uses Petri Net models as its input. We find an LPTF (Largest Processing Time First) based queuing discipline to give the best performance.

Keywords: Read/Write buffer system, time extended Classical Petri Net, modeling, simulation

Presenting Author's biography

Dr. Ad Aerts holds a doctorate in Mathematics and Science from the University of Nijmegen, the Netherlands (June 1979). After working as postdoctoral researcher at the international laboratories in Los Alamos, and Brookhaven, USA and as a senior fellow at CERN, Geneva, Switzerland, he joined the Department of Mathematics and Computing Science of the Eindhoven University of Technology in 1985. His research interests include methods, tools and techniques for engineering information systems. Recent research activities focus on performance of information systems, adaptive Web-based systems and Semantic Web technology. He is (co) author of more than 100 scientific publications.

1 Introduction

The CMS (Compact Muon Solenoid) experiment is one of the experiments that will be run at the Large Hadron Collider (LHC) facility at CERN, Geneva. Once the CMS experiment is running, it will produce data in several streams with a total throughput of 225 MB/second. To store, distribute and analyse the data from the CMS detector, the LHC Computing Grid (LCG, [1]) is used. The LCG is a distributed computing system built to support the physics community. The LCG consists of a number of cooperating computer farms located at computer centres spread around the world. While some centres are more directly connected to the experiment's data source than others (and mostly offer more system resources to the LCG), the proposed service architecture of the LCG is hierarchical. At the top of this LCG service architecture is the computer farm that is directly connected to the experiment's computer farm, named the Tier0. The Tier0 is connected through high-speed connections to several Tier1-centers that are part of the second layer of computing farms in the LCG. These Tier1 centres are connected again to other elements in the LCG network of equal or smaller size than the Tier1 centres themselves, and so on. Every TierN element in the CMS LCG structure has its own responsibility in the global CMS-data physics analysis tasks. A Tier5 element, for example, is just a desktop computer running physics analysis software.

The Tier0 input buffer system has to read out the detector data buffers and to supply this data as input for the other Tier0 processes such as the reconstruction software. It has the structure described in Figure 1.

The data in Figure 1 flows in different streams of 2GB files from the CMS on-line computing farm (the computing farm located directly at the experimental site) to the Tier0 input buffer with a total volume of on average 225 MB/s [2]. This means that the Tier0 input buffer must be able to write data coming from different streams with at least a total write speed of 225 MB/s. At the same time that data must be read as well. It was decided to study a buffer implementation with RAID5 disk servers (a particular kind of Redundant Array of Inexpensive Disks that is resilient against crashes of single disks). The writing procedure uses a FIFO scheduler to decide which file will be written first. The reading procedure uses a First Available Item First Out scheduler (a FIFO scheduler that skips items that are not available for reading at the moment they were supposed to be read). This paper reports on a project that was set up to uncover the issues involved in using disk servers to implement the Tier0 buffer system and compares several possible solutions.

2 Possible Tier0 input buffer disk server scenarios

2.1 The modelling problem

The first task of the Tier0 input buffer is to accept all data from the on-line computing farm and to store this data. A second task is to transfer the experimental data to the other processes. For the first task, a disk server setup that gives the highest priority to writing is most suitable. This means that the write speed is much higher than the read speed when both actions take place at the same time. The proposed disk server has this property [3]. In the same benchmark [3] it was also concluded that the maximum disk write and read speed of the proposed disk server configuration are not sufficient to meet the Tier0 input buffer requirements. For this reason more than one disk server is needed for the Tier0 input buffer. We will consider situations in which there are several disk servers available, which may be reading, or writing data, or both.

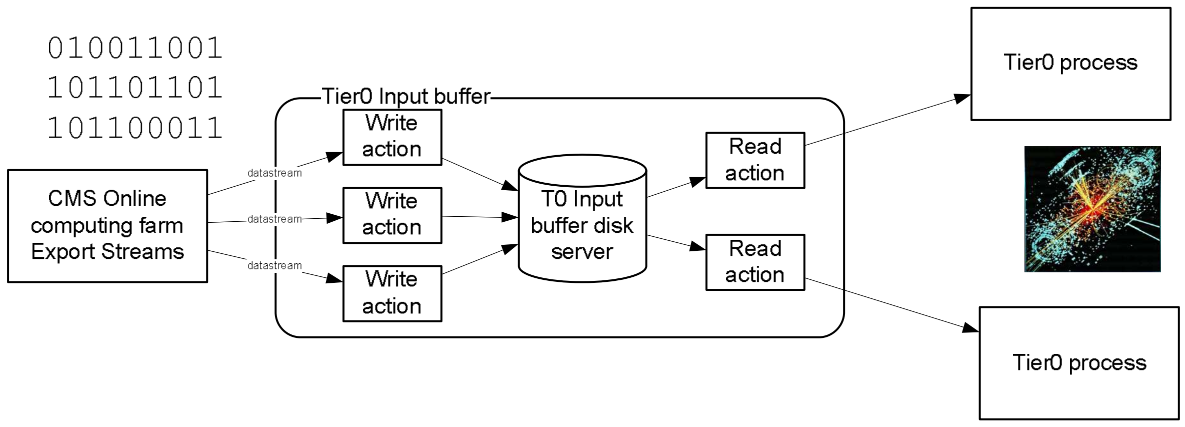

Figure 1: Tier0 input buffer dataflow. The number of read and write actions and data streams is an example in this model. The Tier0 processes are jobs in the Tier0 job processing machines. The bits on the left (input) are reconstructed to physics events on the right (output).

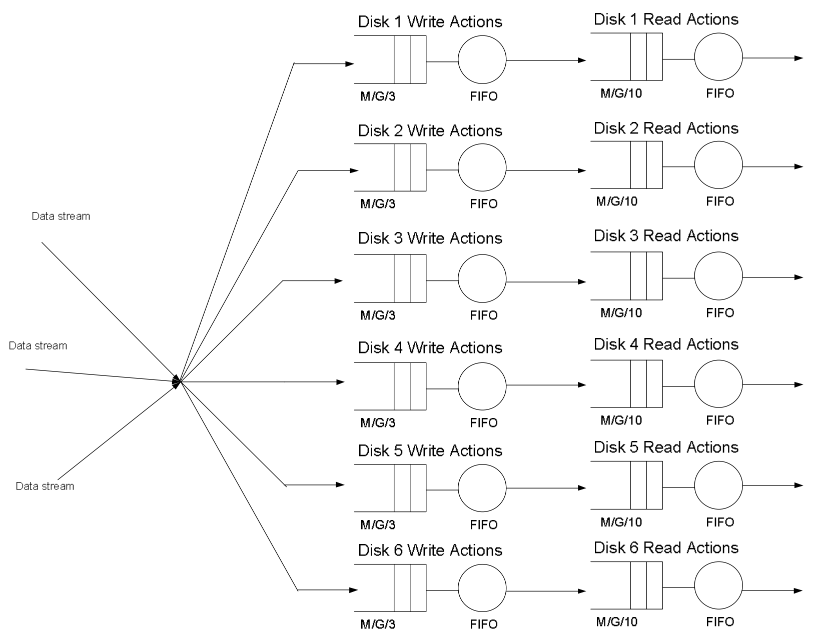

Figure 2: A queuing network representing the naïve Tier0 input buffer problem

The number of simultaneous write actions influences the total write speed of a disk server (according to [3]). But the number of read actions on a disk also has an influence on the total write speed. Disk servers perform better when they exclusively read for an extended period of time, followed by a period of exclusively writing, and so on. The question then is what the length of these periods should be. Write operations to a full disk will lose data, which is not acceptable. And reading from an empty disk waists time that could better be spent writing. Therefore, the alternation of reading and writing periods has to be chosen carefully. We studied 3 different types of triggers to start the alternation of these periods. These are the OnEmpty trigger, which changes a reading period to a writing period, when all items on a disk server have been read out, the OnNItemsAccepted trigger, which ends a writing period after N items have been written to one disk server, and the Fixed Time Change trigger, which alternates after N seconds.

It is not sufficient to change the I/O mode of one disk server, since one has to make sure that on average as much data will be read as is written. Another server will have to change mode to compensate. To select this disk server, any of a number of queuing disciplines can be used. Literature (like [4]) provides some possibilities to apply to this situation. The expectation is that different queuing disciplines will result in different properties of the Tier0 input buffer.

A first queuing model of this situation is given in Figure 2. Data coming from different streams is queued to be written on one of the disk servers after which it is queued to be read. This model immediately raises the question how many servers there should be at the minimum. The number of disk servers in this model is 6. At most three write streams are served, and ten read streams.

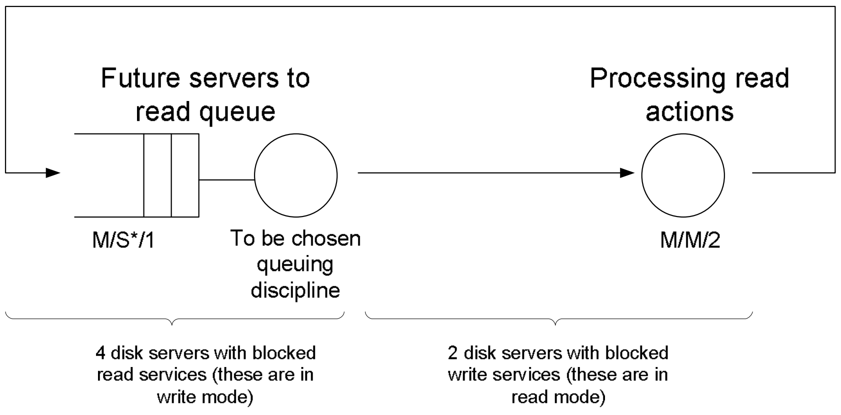
Future servers to read queue
Processing read actions
M/S*/1
To be chosen queuing discipline
M/M/2
4 disk servers with blocked read services (these are in write mode)
2 disk servers with blocked write services (these are in read mode)

Figure 3: The queuing network of the Meta scheduler for an onEmpty, or a Fixed Time Change trigger scenario

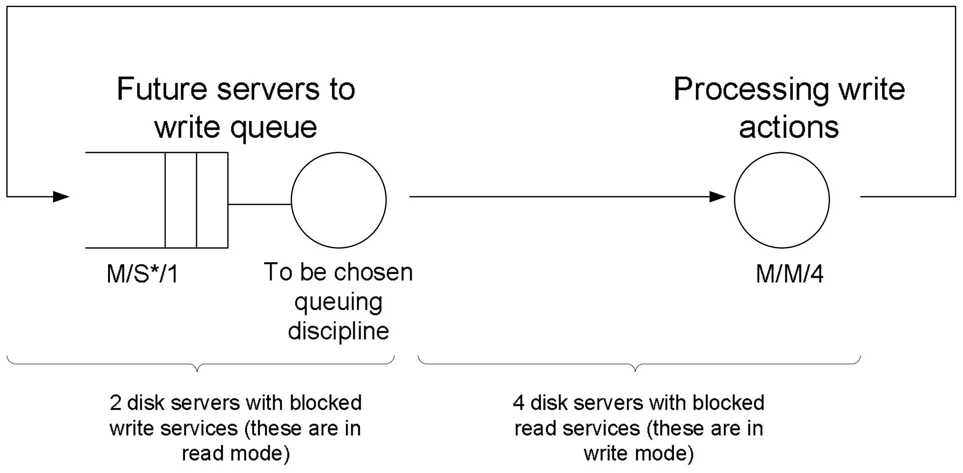
Future servers to write queue
Processing write actions
M/S*/1
To be chosen queuing discipline
M/M/4
2 disk servers with blocked write services (these are in read mode)
4 disk servers with blocked read services (these are in write mode)

Figure 4: The queuing network of the Meta scheduler for an NItemsAccepted change trigger scenario

The naïve model in Figure 2 describes the fact that each data item that is written is queued for reading. It does not show how the disk server that will be written to next is chosen, and it allows intermingling of read and write actions on a particular server. A Meta scheduler is needed to assign the read and write modes. This means that particular "Disk read actions" and "Disk write actions" services in Figure 2 are disabled or enabled according to the discipline enforced by this Meta scheduler. Since different disk server mode change conditions have been used, two different Meta schedulers have been used. These are presented in Figure 3 and Figure 4.

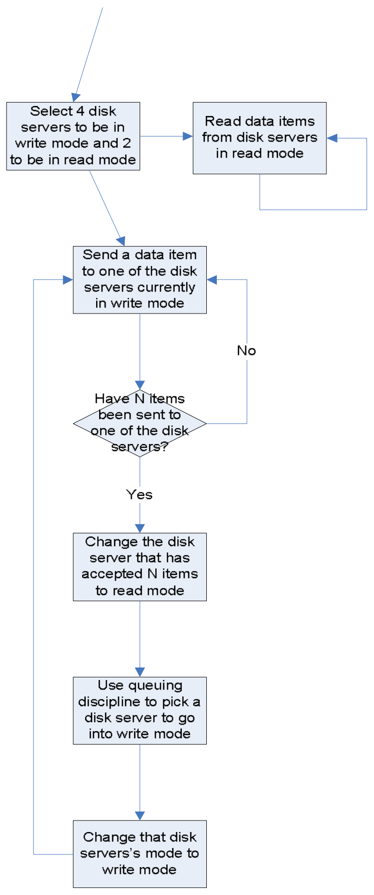
Select 4 disk servers to be in write mode and 2 to be in read mode
Read data items from disk servers in read mode
Send a data item to one of the disk servers currently in write mode
No
Have N items been sent to one of the disk servers?
Yes
Change the disk server that has accepted N items to read mode
Use queuing discipline to pick a disk server to go into write mode
Change that disk servers's mode to write mode

Figure 5: Flowchart of the OnNItems-Accepted scenario

The service time distribution S* in the "Future servers to read/write queue" is not a regular distribution. In order to let the number of disk servers in write and read mode stay constant, synchronization between the two services has to be performed. This means that as soon as the "Processing read/write actions" services processed their queued item, the "Future servers to read/write queue" service has to process an item as well. For this reason, the notation S* for the service time distribution abbreviation has been used.

This special service time distribution combined with the usage of the two layers (the normal naïve layer and the Meta layer) of queuing network to represent the problem makes the Tier0 input buffer problem complicated. The mutual dependence of a lot of the parts increases the overall complexity. We did not find much queuing theory literature about this kind of situation. This led to the decision to use simulation as the research method over an analytical queuing theory method. A third alternative, to build a prototype buffer system, has been rejected because it is too expensive in this preliminary stage of the research.

2.2 Overview of different scenarios

There are many scenarios one can make by combining one of the three triggers with a discipline for changing a pair of servers from reading to writing and vice versa. In a first exploration we limited ourselves to: FIFO, LIFO, LPTF (Largest Processing Time First), SPTF (Smallest Processing Time First), and Random. In combination with the OnEmpty trigger (all items on a server have been read), FIFO means that the next disk to be in read mode is the one that went into write mode the longest time ago. The empty disk server then goes to write mode. LIFO selects as next disk server in read mode the one that went into write mode the shortest time ago. LPTF selects the server that has the largest number of items stored, since all items are of roughly an equal size, whereas SPTF selects the server with the smallest number of items stored. The random discipline selects a server at random, and is included to see what improvement the use of knowledge for the selection of a disk server gives, if any. The disciplines are given a matching interpretation for the other triggers. Besides the scenarios mentioned above, where reading and writing is controlled, a scenario where the disk server reads and writes on demand, mixing read and write mode, has been studied to investigate how large the impact of controlling the I/O mode of a disk server is, if any.

The diagram in Figure 5 illustrates the implementation of a scenario for a server for the case where the trigger is fired after X items have been accepted.

3 Simulation of the Tier0 buffer problem with Yasper

The method of choice to perform an initial investigation of the behaviour of the Tier0 input buffer system and identify the important issues is simulation. The behaviour of the different parts and their dependencies is modelled and the result is fed into a simulation engine. The outcome of these simulations gives information about the influence on the overall Tier0 input buffer performance of different set-ups in the Tier0 input buffer.

The tool selected to do this simulation is Yasper [5]. Yasper is a tool for both modelling and simulating stepwise processes, developed at the Computer Science Department of the Eindhoven University of Technology. The technique used to model these processes is a time extended Classical Petri Net. This modelling technique is suitable for our purpose because the dataflow in the Tier0 can be modelled as a workflow process. Yasper can run manually or automatically. In the former mode, one can follow the files (workflow cases) step by step through the network and so gain insight in the dynamics of the network. This answers questions such as “are there any bottlenecks or pile-ups of items?”, and “what is the effect of the particular trigger or queuing discipline on the processing speed?”. The run mode is meant for gathering statistics.

The results from the Yasper simulations are interpreted in two ways. First, one can interpret the results according to the case-view. This means only properties of individual cases are used to draw conclusions about the simulated dataflow scenario. This interpretation provides statistics about running times of items in the system, and about tracking information of items in the system.

The second possibility to interpret the simulation results focuses on the number of items present in the

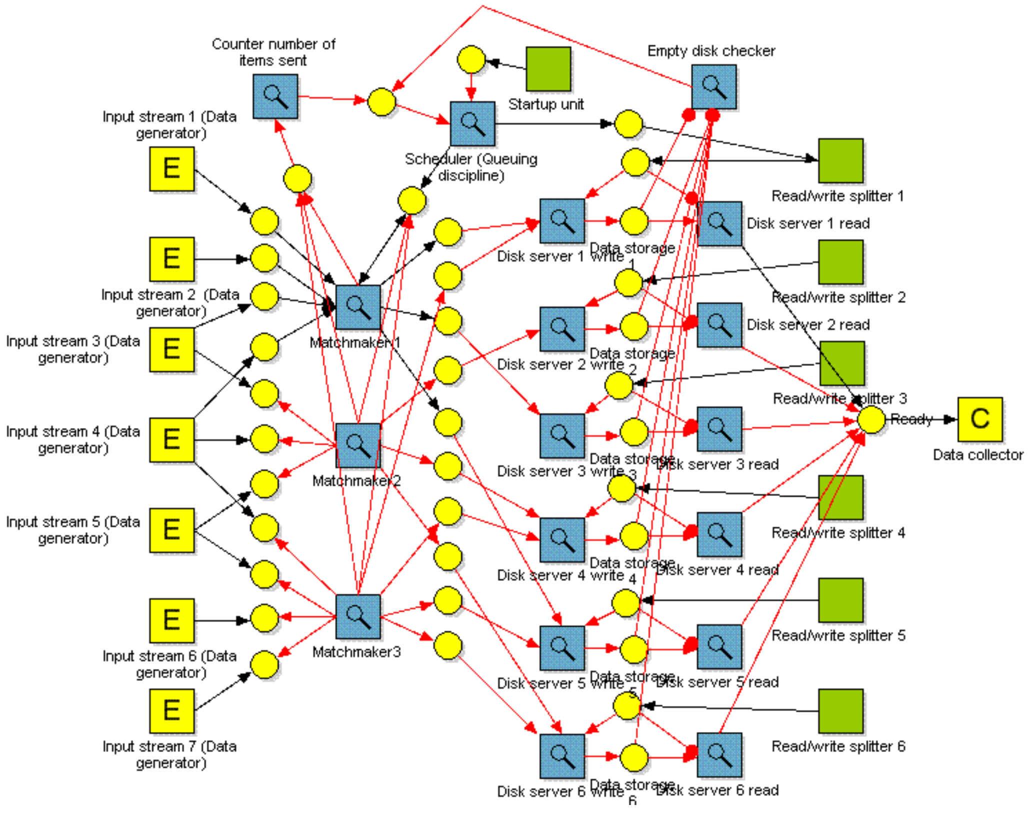

Figure 6: The connections between the different components of the Petri Net models used

simulated system over time. To use this interpretation the Yasper simulation results have to be parsed to a new format. The results of this interpretation method provide us with statistics about disk buffer usage in the simulated situation. Also congestions in the systems are easy to detect with this method of interpretation.

3.1 The construction of the models for the different scenarios

As mentioned before, the Yasper tool is used to run the simulations. This tool uses Petri Net models as an input to specify the simulated situations. For this reason the selected scenarios need to be specified in Petri Net models. A representative network is show in Figure 6.

The network in Figure 6 also demonstrates the modularity of the models. Parts of a network that occur more than once can be captured in a component, which is a Petri Net model in its own right, and which can be reused (instantiated) more than once in the same global model. Components are represented by boxes with a magnifying glass. The modularity arises because all of the selected scenarios have many elements in common, and the Petri Net models of these elements can be reused for different scenarios.

Once one has designed the components, only the connections needed for coordination need to be added to obtain a model for a particular scenario, in addition to the boxes that produce (labelled with an E) and consume (labelled with a C) items.

3.1.1 A Petri-net component representing a RND scheduler

Figure 7 shows a Petri Net model for the random scheduler component. The RND scheduler model receives an item in the “request to select disk server to read” place whenever a change of state is imminent in one of the servers. 11 tokens from one place can be consumed (this changes the mode of the disk server that is connected to this place from write mode to read mode), and any of the places can get 11 new tokens (and change the mode of the disk server that is connected to this place from read mode to write mode), even the ones that already contain some. The last action is non-deterministic. The last category is sorted out with the help of boxes that detect the appearance of more than 11 tokens in one place.

In reference [6] a detailed description of the other Petri Net components used is given. A model for each of these components is presented as well as a table that gives an overview of the inclusion of the various components in the different scenarios.

3.2 Comparison criteria and model parameters

To compare the different results to each other, criteria are needed. Criteria that are interesting from the point of view of implementation of a high performance buffer system are:

[reqprop1]: number of disk servers used

[reqprop2]: total capacity of the disks used

[reqprop3]: total throughput of the system

[reqprop4]: sojourn-times of items in the system

In Yasper, a disk server is modelled as a structural component of the net and cannot be changed dynamically (see Figure 6, where one can see 6 servers in write mode and 6 servers in read mode). The throughput of the system is a design parameter of the experiment, and therefore should be kept fixed. Therefore, [reqprop1] and [reqprop3] are input parameters of the simulation models, and have been chosen the same for all models. Only [reqprop2] and [reqprop4] have been used to compare the different simulation results from the simulations. Since

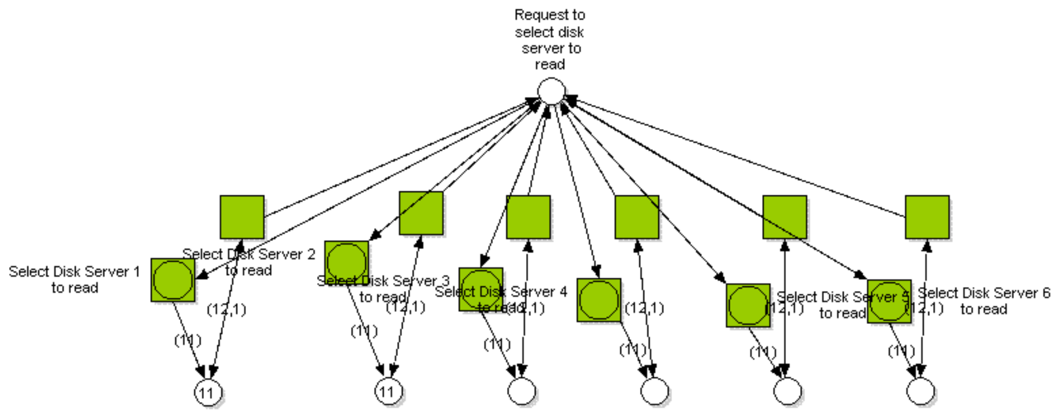
Request to select disk server to read
Select Disk Server 1 to read
Select Disk Server 2 to read
Select Disk Server 3 to read
Select Disk Server 4 to read
Select Disk Server 5 to read
Select Disk Server 6 to read
(12,1)
(11)
11

Figure 7: A model of an RND scheduler

[reqprop1] and [reqprop3] are constant factors for the compared scenarios, these parameters have no influence on the overall performance figures in this section.

A question that needs to be answered before running the simulations is what the appropriate number of items to be simulated per scenario is. Because Yasper has been designed to run simple process models instead of the complicated Petri Net models needed here, the simulations require much processing time per item. This makes it necessary to keep the number of simulated items as small as possible. On the other hand too small a number may not be sufficient to see the specific behaviour of the different scenarios. The number of items should be such that every disk server in every scenario is switched from read to write state and back again at least once. Tests with different scenarios showed that 6000 completed items was a reasonable number to run the simulations with.

Also the amount and properties of the hardware need to be defined before running the simulations. The number of disk servers has been chosen so that it is minimal. Given the required throughput, and the favouring of writing over reading, this number turns out to be 6: 2 servers for writing and 4 for reading at one time. Because of the latter choice, the number of data input streams is also chosen to be 4. However this does not imply that every disk server in read state will receive data from exactly 1 input stream during a period of time.

The total rate that items will come into the system is 225 MB/s. This means that the 4 data input streams also need to create data items at this rate. Because every item in the simulation represents a file of the size 2048 MB, the average item creation time for every stream is 2048/225*4 = 36.4 seconds.

4 Outcome of the simulation

As indicated above, the simulations produce two kinds of output: total disk usage over time, and the sojourn time distributions of the simulated data items. Figure 8, as an example of the first kind of output, shows the number of items that are present in the simulated system as a function of time. Because the generation of new items stops after 6000 cases have been completed, more than 6000 items will have been generated to achieve this. The incomplete cases will get an end-time that does not correspond to the time they will be read from the system, but to the time they reach the last-reached place.

Because of this the number of items in the system is only representing a real system until the time the 6000th item completes. In the graph this 6000th item has been marked by a red, vertical line. The graph to the right side of this line is not representing the behaviour of the simulated scenario.

Figure 8 shows the influence of the disk server state change trigger on the number of items in the system (reqprop2) for a FIFO based scenario. It shows that the OnEmpty trigger outperforms the other two triggers that are less sensitive to the state of the system. As explained above, the simulation is only realistic on the left hand side of the vertical line.

Figure 9 shows a graph for the number of completed items with a sojourn time in a given interval. The

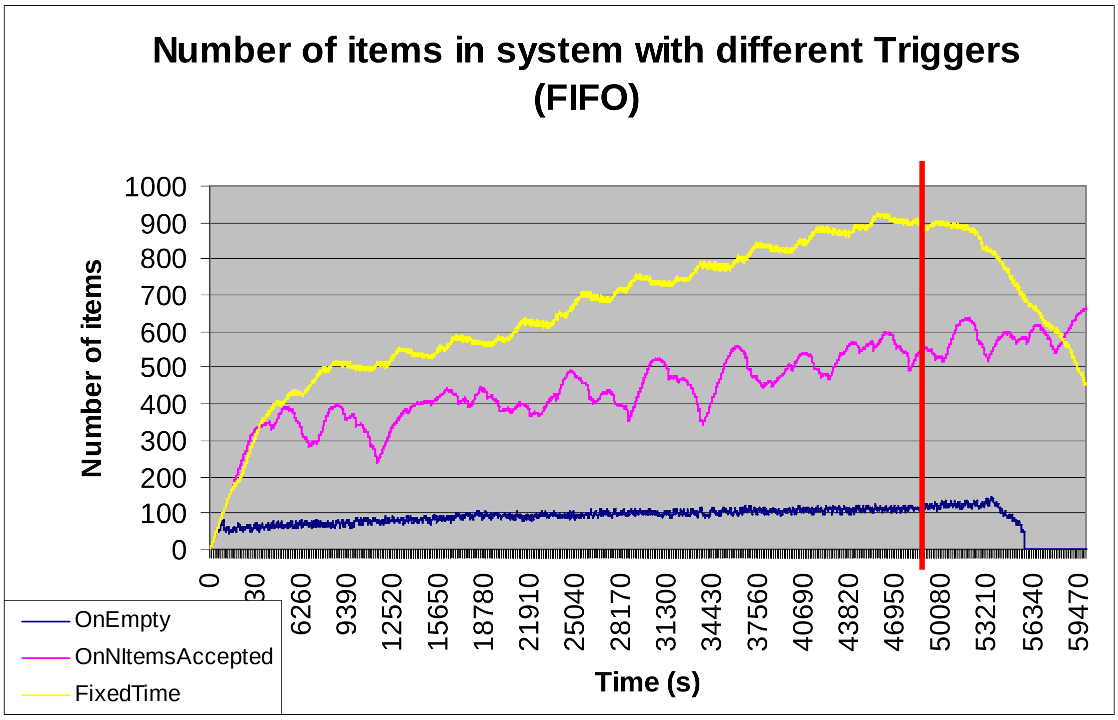

Figure 8: Example of a graph that shows the number of items in system over time

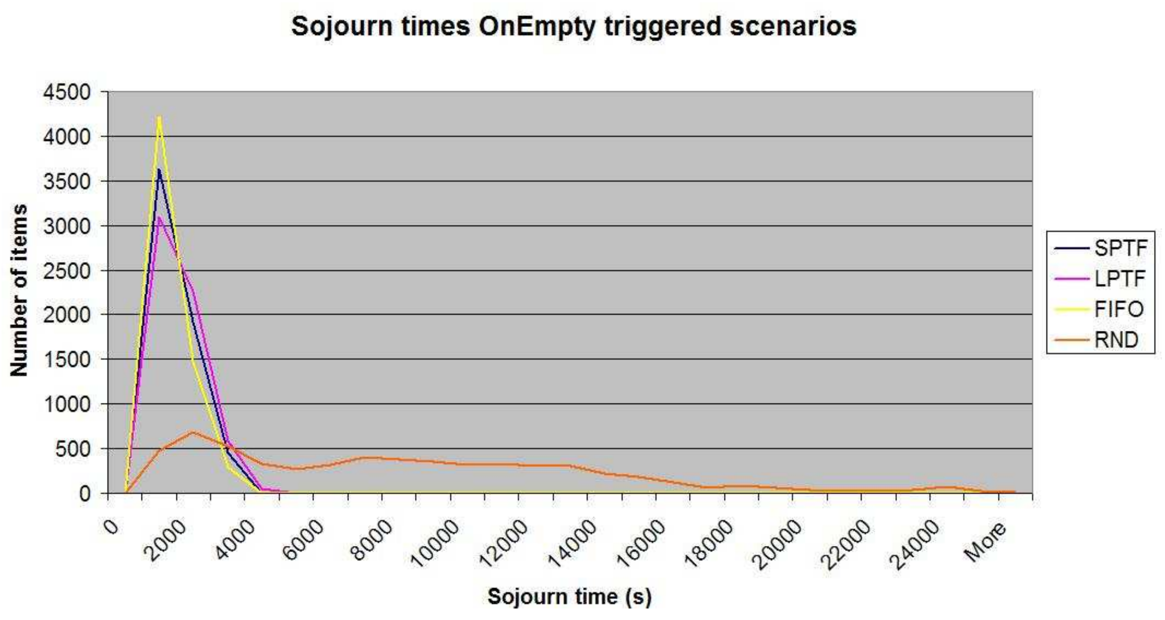

Figure 9: Sojourn times of items in OnEmpty trigger scenarios for different schedulers

values for a particular scheduling discipline have been connected by a line to ease comparison between the disciplines. Figure 9 illustrates the influence of the different schedulers on the performance of a scenario. The graph shows the sojourn-times of items in systems using different schedulers in combination with an OnEmpty trigger. It is clear that the uncontrolled random choice leads to a very wide spread of sojourn times.

Note that Yasper produces output in a textual form that has to be processed further to produce the graphs. To provide support for systematic simulation and analysis of simulation data a shell for Yasper, Yassim has been used. The procedures to do this are described in [6].

5 Comparison

This section relates the results to the problem definition. The possible scenarios have been grouped to compare the different schedulers, the different read/write modes and the different disk server state change triggers we compared. Also a comparison with the expectations of the different scenarios will be made.

5.1 FIFO scheduling

This type of scheduling is vulnerable to (temporary) fluctuations in the input streams. If one server receives more data than it can handle, sojourn times will be penalized with at least 1 cycle time (the time it takes for a server to go from read state to write state and back to read state).

An exception to the last mentioned cycle time penalization is if an OnEmpty trigger has been used in the scenario. This causes huge difference between the sojourn times of the different trigger scenarios used. The results are shown in Table 1.

One of the expectations of a FIFO scheduler is that there is not much variance in the sojourn times of the files in the system. This is partly true, the graphs of the simulated sojourn times show patterns that make it possible to predict the new sojourn times. But the variance on the items themselves can be high, because there are more influences on these sojourn times than just the scheduling discipline.

Table 1: Performance parameters for the FIFO scheduled simulations

number of disk servers used:	> 6
total size of the disks used:	OnEmpty: > 280 GB OnNItemsAccepted: > 1320 GB Fixed time change: > 1800 GB
total throughput of the system (with 6 disk servers for the first 6000 simulated files):	OnEmpty: 227 MB/s OnNItemsAccepted: < 200 MB/s Fixed time change: 261 MB/s
sojourn-times of items in the system:	OnEmpty: > 870 seconds OnNItemsAccepted: > 5490 seconds Fixed time change: > 6765 seconds

For example for the FIFO OnEmpty scenario there is the choice of which disk server in read mode will receive the item from a data stream. If this is the server that will be the next server in read mode, the sojourn time will be shorter than if it was sent to a disk server in write mode which will be the 4th disk server that will go to read mode.

Another expectation was that the disk usage is minimized if the OnEmpty trigger is used. This is confirmed by the simulation results. An expected disadvantage of this scenario is that there will not be any items in the system with short sojourn times. This was also true according to the simulation results.

5.2 LIFO Scheduling

This discipline was only simulated at a high-level. It is very vulnerable to variations in input streams since servers that receive the highest load get the shortest time to read out their buffers. Better performance will be achieved if time outs are possible on items [7] (since every data item has to be processed, this is not possible for the Tier0 dataflow scenario)

This was also confirmed by the simulation. The only advantage that was mentioned for this scenario [8]is that there will be a high percentage of jobs with short sojourn times. The simulation will give a bias in this case, since there will be many jobs that have apparently an acceptable sojourn time because they did not finish during the simulation.

5.3 LPTF Scheduling

The shortest average sojourn times are achieved with LPTF scheduling, especially if the system runs for a long period of time. In addition, the slope of the regression function is rather flat. This discipline has some interesting features:

The number of items in the system is already stable with a small number of files on disk.

This scheduler performs best in combination with the OnEmpty change trigger. If some disk servers receive more files than others, comparatively more time will be allowed to these servers to read out their buffers (because the chances that these disk servers contain the highest number of items are larger). The OnEmpty LPTF scheduling scenario thus reduces the number of disk server state changes. This is beneficial because a disk server state change incurs some overhead, because ongoing work has to be finished.

In combination with the OnNItemsAccepted trigger also reasonable performance can be obtained. The number of items that have to be sent before a change is accomplished has to be reasonably high to get the best performance. This increases the risk of idle disk servers that are supposed to be read, and the disk buffer sizes need to be larger, because write periods on one disk will be longer according to the number of items sent before changing the disk server mode. The optimal number of accepted items before a trigger is sent out lies, according to the simulation results, between 200 and 400.

Table 2: Performance parameters for the LPTF scheduled simulations

number of disk servers used:	> 6 (or 6 if mixed read/write will be used)
total size of the disks used:	OnEmpty: > 360 GB OnEmpty with mixed r/w: 270 GB OnNItemsAccepted: > 2000 GB
total throughput of the system (with 6 disk servers for the first 6000 simulated files):	OnEmpty: 259 MB/s OnEmpty with mixed r/w: 224 MB/s OnNItemsAccepted: < 231 MB/s
sojourn-times of items in the system:	OnEmpty: > 615 seconds OnEmpty: 740 seconds OnNItemsAccepted: > 7300 seconds

The prediction for the scheduler is that less disk space was needed. This is confirmed by simulation results. The LPTF scheduler needs the smallest amount of disk space of any disk server change trigger combination. The same holds for the variation on the sojourn times. These are the lowest for the scenarios which use a LPTF scheduler.

5.4 SPTF Scheduling

This discipline is characterized by a large variation of the sojourn times. This arises because a part of the files get very long sojourn times, whereas another part of the files have very short sojourn times.

This discipline suffers from the frequent changes between read and write modes. The number of items in the system is therefore rising more over time than in case of the other scenarios. The idle time of disks in read mode is moreover high when using an OnNItemsAccepted trigger scenario.

One of the expectations for this scenario is that there will be a high percentage of the jobs that get very short sojourn times. This only holds when the trigger is not the OnNItemsAccepted disk server change trigger. When this trigger is used, the percentage of jobs with short sojourn times is not that big. This is caused by the disk servers that host the files with the shortest sojourn times, frequently are idle because there are no items to be read anymore on their disks (the reading of items with short sojourn times is replaced in this scenario by being idle).

Table 3: Performance parameters for the SPTF scheduled simulations

number of disk servers used:	> 6
the total size of the disks used:	OnEmpty: > 320 GB OnNItemsAccepted: > 5000 GB
total throughput of the system (with 6 disk servers for the first 6000 simulated files):	OnEmpty: 232 MB/s OnNItemsAccepted: < 315 MB/s
sojourn-times of items in the system:	OnEmpty: > 980 seconds OnNItemsAccepted: > 22642 seconds

5.5 OnEmpty change

The OnEmpty change trigger has the nice characteristic that for disk servers that contain many items, the relative share of disk server read/write time is increased by postponing disk server state changes.

The best performance of the OnEmpty change trigger scenario is achieved in combination with the LPTF queuing discipline. In this case, the LPTF scheduler picks the disk to be read with the largest amount of available files and the OnEmpty change trigger keeps this disk in read mode until no items are left to be read anymore.

The expectation is that the sojourn times of items in a scenario with an OnEmpty disk server state change trigger have more deviation. This has been confirmed in the simulations.

5.6 OnNItemsAccepted change

Although the number of items in the system is higher than in the case of the OnEmpty change trigger scenario, the regression function of the results of an OnNItemsAccepted trigger scenario is not showing significantly worse results. This can be explained by the fact that disk servers that store fewer items are not distinguished from the ones that store (too) many items. This results in disk servers getting more resources per accepted item to buffer than other servers. The result of this uneven distribution is that there will be a lot of files that have relatively small sojourn times in the system and complete, and a set of files that do not have such a small sojourn time and do not complete during the simulation run. The last group does not contribute in the sojourn time regression function and causes this regression function to be too optimistic.

Although the expectation is that disk servers get equal loads, this is not borne out by the simulation results. The cause of these inequalities is that the input streams did not send items to the different disk servers in an equal way (variations in the arrival process). The expectation of the long sojourn times if items that are left on a disk server, even after a read phase has been ended can also been seen in the simulation results.

5.7 Fixed time change

Because the server state change moments are more predictable, the number of items in the system is rising more evenly over time (with less unexpected fluctuations).

Sojourn times get easily penalized with multiples of the cycle time (the time it takes for a disk server to go from the read state via the write state back to the read state). This is caused by the lack of synchronization between the disk state controller and the disk servers.

The predicted low variance of the sojourn time of the items in a system with fixed time disk server state change triggers was higher than expected. This was because of the penalized items mentioned before that stayed on a disk server, (even after a readout phase has ended), which have remarkably higher sojourn times compared to other items.

5.8 Mixed read and write mode

Although having simultaneous read and write actions on the same disk server in general increases read and write times, a special variant of mixed read and write modes surprisingly has the best performance. This better performance, in case of the OnEmpty trigger LPTF discipline combination, can be achieved by allowing read actions on disk servers in write mode. To protect the write performance, the write actions are given top priority so that the write actions take place without any influence from simultaneous read actions. The performance of the read actions on these disk servers, of course, is low, but it enhances the overall performance of the read actions, because the actions on the disk servers in read mode are limited exclusively to read actions. This behaviour is also according to the expectations.

6 Conclusion

The best performance with the 6 disk server setup is obtained by using an LPTF scheduler, an OnEmpty change trigger and a mixed read/write mode. It best takes into account that the read/write characteristics of disk servers by maximizing the bulk operation periods on the disk servers. This setup thus is recommended for the Tier0 input buffer system.

In the simulation this setup (with 6 disk servers, each with a total capacity of 1.9 TB) was sufficient to buffer the data coming from 4 different input streams with a total input of 225 MB/s. The dominant issue here was not the storage capacity but the reading and writing speed. The simulations, however, deal with the simplified situation that the read actions are pushed from the input buffer disk servers to the other parts of the Tier0. In the real Tier0 system, it is conceivable that there will be a pull from the downstream parts on the Tier0. Such high priority read requests for specific data will trigger the read actions on the input buffer disk servers. For this action a scheduler must be capable of stopping a write phase on a disk server in order to read on that disk server at a high speed. Since 4 disk servers can be in a write phase at the same time, there must be 4 extra disk servers that can replace the acceptation of the write actions of the disk servers mentioned before.

This makes the final proposed Tier0 input buffer system a 10 disk server systems controlled with an LPTF scheduler, an OnEmpty change trigger and a mixed read/write mode.

This paper has presented the results of a first analysis of the Tier0 input buffer system and has been focused on the modelling of the problem in order to discover which questions are the relevant ones to ask. The modelling phase was followed by simulation to improve insight into the behaviour of the various models. To more certain that the models represent the real situation, some model parts have been remodelled in a tool called GPSS. This tool uses queuing models described in a GPSS specific language. The results have been found to be consistent within the error margins.

The modelling effort also reconfirmed the fact that classical Petri nets become large very quickly. This tendency can in part be counteracted by using hierarchy in the form of subsystems, but this does not reduce the intrinsic size of the models that has an impact on the duration of the simulation. More powerful tools such as CNP tools [9] will be required to allow the extension of the description, for instance to accommodate high priority read requests.

Another interesting direction in which to continue the work of this paper is to investigate the more general aspects of the modelling problem from a queuing model point of view, as discussed in section 2.1. It is an open question whether one can formalize the concept of a meta-scheduler in a queuing framework. This would allow one to construct and analyse two level queuing models in which the queuing network at the lower level can be changed dynamically by a scheduler at the higher level. In our case we looked at a pair of servers that changed role in a compensating way specified at the higher level. A related example from the transportation domain would be a train operator that has a number of trains and carriages in active service, transporting people, and another number in maintenance or repair. The maintenance can be scheduled, the repair is incident based. Both involve changing the role of the equipment from 'in service' to 'out of service' and back again.

7 References

[1] J. Knobloch et al., "LHC Computing Grid Technical Design Report", CERN/LHCC 2005-024.

[2] "The computing project: Technical Design Report", Ed. L. Taylor, CERN/LHCC 2005-023

[3] B. Panzer-Steindel, *"Disk server benchmarks V2,* CERN/IT Report 10.03.2006

[4] Leonard Kleinrock, "Queueing Systems. Volume II: Computer Applications"

[5] Yasper: *"Yet Another Smart Process Editor"*, http://www.yasper.org/ (Jan 2007)

[6] LHJM Wijnant, "A comparison of CMS Tier0-dataflow scenarios using the Yasper simulation tool", Master thesis, Eindhoven University of Technology, Jan 2007.

[7] Naresh Singhmar, Vipul Mathur, Varsha Apte and D. Manjunath: *"A Combined LIFO-Priority Scheme for Overload Control of E-commerce Web Servers"*, Proceedings of the International Infrastructure Survivability Workshop (affiliated with the 25th IEEE International Real-Time Systems Symposium), Lisbon, Portugal, 2004.

[8] I. Adan and J. Resing. "Queueing Theory", Lecture Notes, Department of Mathematics and Computing Science Eindhoven University of Technology, Feb 2002.

[9] CPN Tools: http://wiki.daimi.au.dk/cpntools/cpntools.wiki (June 2007)