TERADATA ASTER

Aster MapReduce Platform Workload Management

Revision History

Version	Authors	Date	Comments
1.4	Bruce Frank	12/14/2012	First draft
1.5	Bruce Frank	12/17/2012	Closed
1.6	Bruce Frank	12/20/2012	Closed
1.7	Bruce Frank	12/28/2012	Closed
1.8	Bruce Frank / Dileep Kumar	01/08/2013	Closed
1.9	Bruce Frank	01/11/2013	Closed
2.0	Bruce Frank	01/15/2013	Closed
2.1	Bruce Frank	01/15/2013	Closed
2.2	Bruce Frank	02/12/2013	Closed
2.3	Bruce Frank	02/21/2013	Out For Review

Contents

Introduction

Today's computer world provides users with many tools to access a database system. As tools become popular more access is provided to these tools and consequently performance of the underlying system may be impacted. Many methods are available to control access to a system; they range from limiting access physically to automatically controlling resources. Through the years computer operating systems have had different methods of controlling resources. One way that has been successful is workload management and most systems today have some flavor of workload management.

There are many reasons to use workload management. The main reason is to make sure the system is utilized properly; other reasons are to control usage of the system by time of day or user groups.

The perception of a product's performance is important for the successful use of that product.
The benefits of using workload management are many:

- increased productivity,
- the ability to provide different SLAs (Service Level Agreement) to different departments or ensure the SLA for a specific department,
- the ability to provide optimal performance,
- the ability to run multiple workloads in an efficient manner,
- and many more.

It is important that you manage access to Aster MapReduce Platform in order to provide optimal performance for the groups that benefit the most from using its features.

1 Overview

Aster MapReduce Platform provides workload management by letting the user or DBA (Database Administrator) create **service classes** which are associated with **workload policies**. Service classes control the priority, CPU resources and memory associated with a work policy. Each service class has a **priority**, a **weight**, a **soft memory limit** and a **hard memory limit**.
Workload policies control how the resources are allocated to queries, users, time of day or many other available predicates.

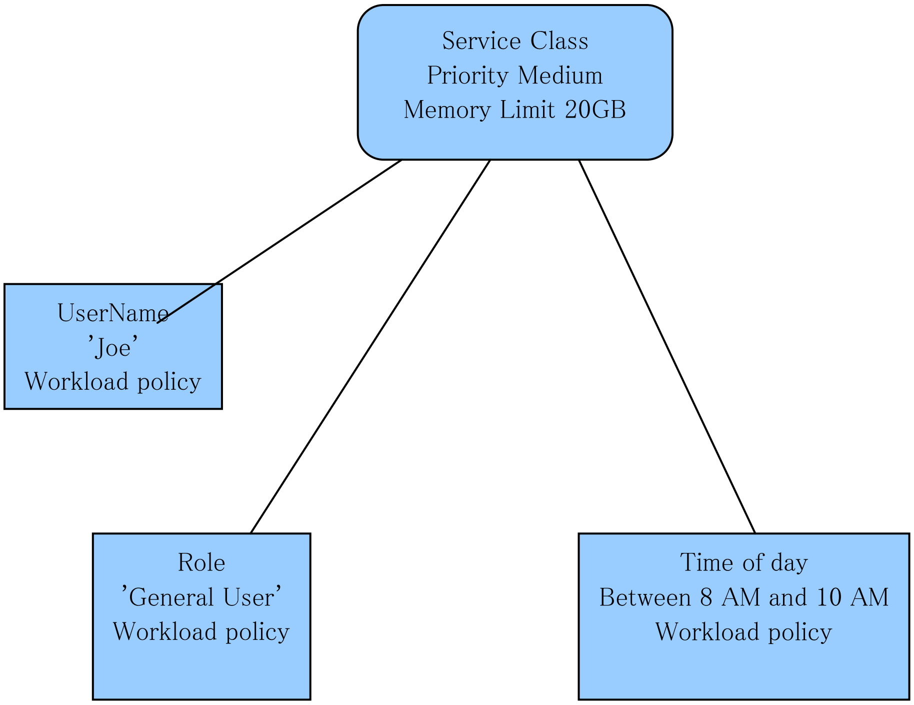


Figure 1 **Service Class Workload Policies relationship**

The above diagram shows the relationship between service classes and workload policies. We have a service class with a medium priority and a memory limit of 20GB, this service class is applied to user “joe”, any user in a role defined as a “General User” and also to any query running between 8 AM and 10AM.

1.1 Service Classes and Workload Policies

Services classes are used to allocate resources which will be associated with a work policy. The resources associated with service classes are related to the priority assigned to the service class, which can be High, Medium, Low or even Deny (does not allow the query to run) and weight assigned to the service class.

Weight is the second-level setting that governs admission to the queue for processing. That is, incoming requests are first evaluated and prioritized by their priority, and then by weight. These two settings also map to a per-node CPU share control and a per-node IO priority.

The weight value is an integer between 1 and 100, inclusive, that establishes how important a job is, relative to other jobs executing *with the same priority value*. A higher weight value indicates a job of higher importance. You set a service class's weight when you create the class in the Admin: Configuration: Workload: Service Classes tab of the AMC. This is stored in the `weight` column of the `nc_qos_service_class` table.

Within a priority level, the weight value dictates the ratio of resource allocation. For example, if two statements execute with the same *priority*, but with *weight* values of 80 and 20, the system will aim to allocate resources in a 4:1 ratio, with most of the resources allocated to the statement with higher weight. Allocation of I/O-related resources is less accurate than allocation of CPU shares, so in this example, the CPU share ratio would be very close to 4:1, while the disk I/O shares cannot be guaranteed as precisely.

Workload policies are used to control query execution. This is done by creating rules that apply to a set of predicates, for example username, time of day, database name, table name, etc. For more details of rules supported refer to Aster MapReduce Platform User Guide.

1.2 Memory Management

The memory component of Aster MapReduce Platform workload management is designed to avoid memory abuse by a query or group of queries that can cause a cluster to hang or fail. With the design of this safety feature comes the advantage of being able to group queries so they can have different memory limits or thresholds.

1.2.1 Memory Limits

Two limits are used by Aster MapReduce Platform:

- "Soft Limit" is the current limit. The soft limit is the value that the kernel enforces for the corresponding resource.
- "Hard Limit" is the maximum limit. The hard limit acts as a ceiling for the soft limit.

For Aster MapReduce Platform configuration we define them the following way;

Soft Limit is defined as the lower bound to the amount of memory used by a service class on each physical node.

This means that once the queries in a service class have reached the limit we will be looking at the queries in this service class to manage memory utilization.

Hard limit is defined as the upper bound to the amount of memory used by a service class on each physical node.

Once this limit is reached the queries will be canceled.

When memory is needed, the system will strive to reduce the memory used by a service class to its soft limit, but the success of this operation is not guaranteed. Even when resources are available and the system is not under memory pressure, a service class will not be allowed to consume more memory than defined by its hard limit.

1.3 Usage

Let’s take a look at an example of how we would use workload management in Aster MapReduce Platform.

Suppose you want executive management to be able to run a report and not be impacted by other user’s queries. You would set them up in a work policy associated with high priority and no memory limits. For example;

- Service Class name **CEOCLASS**
- Priority High with weight 90 and no memory limits

By setting a service class and work policy for executive management with these parameters, you enable their queries to run with almost the highest priority you can assign. This means as long as there are resources available, they will be allocated to these queries.

Table 2-1 shows the service classes defined. “CEOCLASS” for upper management queries and “USERCLASS” for queries that can be associated with users. The last class is “ANYCLASS” which will be associated with the “WLP_DEFAULT” Work policy.

Name	***Priority***	***Weight***	***Memory soft limit %***	***Memory hard limit %***
CEOCLASS	**3 (high)**	**90**		
USERCLASS	2 (medium)	30	30	35
ANYCLASS	1 (low)	50	25	30

Table 2 - 1: Service Class definitions

Table 2-2 shows the workload policies “WLP_JOE” for user “joe” with association to the service class “USERCLASS” and “WLP_CEOCLASS” for anyone in a role of “ceo_role” with association to “CEOCLASS”. The last policy is the “WLP_DEFAULT” policy which will be used if none of the other policies apply.

Name	***Predicate***	***Service class***
WLP_CEOCLASS	**'ceo_role'=ANY(roles)**	**CEOCLASS**
WLP_JOE	username='joe'	USERCLASS
WLP_DEFAULT	True	ANYCLASS

Table 2 - 2: Example Workload Policies

When in use, workload management requires that at least one *default workload policy* be defined. A default workload policy is one which matches any activity in the system and it is defined by a predicate of 'true'. This ensures that any activity in the system can be mapped to at least one workload policy. Note that the default workload policy should appear last in the evaluation order, so that it will be applied only if none of the other workload policies apply.

2 Aster MapReduce Platform Workload Management Architecture

The queen is responsible for managing the system configuration and coordinating queries.
The queen accept, optimize and plan SQL or SQL/MR statements from clients, distribute work across the cluster, aggregate results from workers, send results back to client and performs system management to ensure that replication levels of the data in the cluster are maintained.

Worker machines are responsible for storage of and computation on distributed data. As your data volume grows, you can add more worker nodes to ensure query execution times scale linearly with data volume. Each worker machines hosts 1 or more (typically 4-6) vworkers (virtual workers) which are individual slices of data and computation that are performed on each worker. The vworkers receive phases of the execution plan that are relevant to the slice of the data they are responsible for. vworkers have the capability to perform inter-vworker communication in case the plan requires shuffling of data between worker machines. Once the computation is complete, the results are sent back to the queen.

ARC, which is short for "**A**ster **R**elational **C**ompute Engine" each vworker will have an ARC Instance.

Aster MapReduce Platform workload management controls the usage of CPU and sets priorities for different queries running in the system. Also it controls memory utilization by queries to avoid a memory starvation that could cause a cluster crash.
You define these parameters to the system is by using service classes and workload policies. A service class defines the priority, weight for the queries, and the memory limits. These settings apply to all statements that map to the service class.

A workload policy defines a set of related activities (SQL queries being the most common), that share similar properties. Assigning these related activities to a service class allows them to be prioritized in a similar manner by Aster MapReduce Platform .

Aster MapReduce Platform uses the Deadline scheduler which is an I/O scheduler for the Linux kernel. The goal of the Deadline scheduler is to guarantee a start service time for a request.

I/O is not managed by Aster MapReduce Platform workload management because the deadline scheduler is quite efficient and Aster MapReduce Platform uses CPU shares to manage resource allocation for the different priorities set in workload management

The **Completely Fair Scheduler** (CFS) is a task scheduler available in Linux Kernel release 2.6.23. It handles CPU resource allocation for executing processes, and aims to maximize overall CPU utilization while also maximizing interactive performance. Aster MapReduce Platform takes advantage of the CFS and uses CPU shares as part of the implementation of quality of service (QoS).

Aster MapReduce Platform controls quality of service (QoS) with their QosManager. Aster MapReduce Platform uses a QosManager that runs on the queen and QosSlaves that run on each worker and on the queen. The QosManager is in charge of service class assignment and communicating with the QosSlaves. The QosSlaves are in charge resource allocation and managing the resources for queries running on workers.

Here is a summary of how QosManager and QosSlave interact.

- QosManager notifies QosSlaves of priority and weight settings for sessions on queen and worker nodes.
- QosSlaves map priority and weight to OS-level values
    - CPU shares
- QosSlaves change processes' priorities
- Subsequent statements generate new contexts
- Sessions may be mapped to different service classes

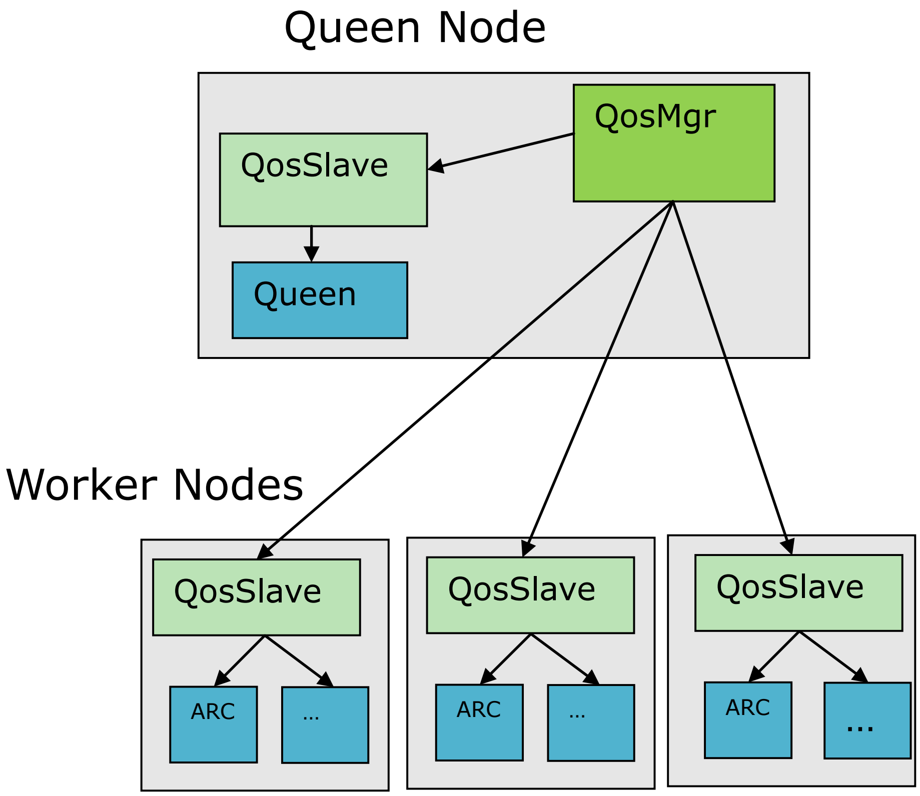


Figure 2 – Aster MapReduce Platform QoS Architecture

3 Policy Classification

Aster MapReduce Platform workload management uses service classes and workload policies, in this chapter we will look at workload policies classification by examining the different type of policies and the predicates used.

User-based policies
Many customers may rely mostly on username and or roles.

A sample business case for using **userName** would be when you have users with limited experience writing SQL commands. You would put them in a service class that would limit the amount of CPU as well as the amount of memory they can use. Without workload management, novice users might write “Unconstrained” SQL statements that consume excessive resources, especially memory.

Example of user based policy:

username=”dba”

Time-based policies
The variable **currentTime** can be used to implement different policies for business hours and off hours.
In this category you would create workload policies for different times of the day, running some resource intensive reports during off hours so as not to affect the ad hoc queries and reports needed by upper management during business hours.

Example of time-based policy:

StmtElapsedTime < '2 mins'

Object-based policies
You can use table and database names to implement different policies by using the **tableNames** and **dbName** predicates.

You may want to map databases or tables to departments, and provide priorities to those departmental queries that are essential to the day to day business, while providing lower priority to those departmental queries that are run on a less critical schedule (e.g. A production database could be given a higher priority, while test databases could be given lower priorities).

Example of object- based policy:

dbName='reports'

IP-based policies

IP policies may be useful for large companies. For example, you could control queries after associating them with branches or office vs. home office by ip address. The predicate to use for this is **clientIpAddr.**

Example of IP-based policy:

clientIpAddr<< '192.1.1/24'

This example specifies client IP addresses contained within subnet 192.1.1/24.

Periodic Re-evaluation

You may choose to allow the same statement or activity to be mapped to a different service classes at different times.
Dynamic statement reprioritization Example: change workloads based on statement execution time

Let us take a look at a business that has a group of upper management that requires reports right after submitting them. They are not the only ones using the system. In the company there are many groups, Interactive users, Database Administrators, Upper management, ETL users, Regular Users and Backup. Each of these groups has different priorities and memory limits. The upper management group has the highest priority and memory limit. The priorities and memory limits will be assigned in the service class while the rules of execution will be governed by the workload policies. The tables bellow show the service class and workload policy definitions needed to satisfy the business requirements.

The *evaluation order* is the deciding factor when a given activity maps to more than one workload policy. The *evaluation order* is defined in the AMC (Aster Management Console) with AMC's Admin: Configuration: Workload: Workload Policies tab.

A workload policy governs only those queries that match its workload predicate as the first match in the set of workload policies. The workload management attributes and their associated values are assigned to the query when it is planned by the system.

Name	Priority	Weight	Memory soft limit	Memory hard limit
CEOCLASS	3 (high)	90		
ADMINCLASS	3 (high)	10	10	15
INTERACTIVECLASS	3 (high)	60	60	75
ETLCLASS	2 (medium)	50	50	65
DEFAULTCLASS	2 (medium)	30	30	35
BACKUPCLASS	1 (low)	10	10	20

Table 3 - 1: Service Class definitions

Name	Predicate	Service class
WLP_CEOCLASS	'ceo_role'=ANY(roles)	CEOCLASS
WLP_DATASCIENTIST	‘ds’=ANY(roles) and StmtElapsedTime < '2 mins'	INTERACTIVECLASS
WLP_ADMINOPS	StmtElapsedTime < '5 mins' and username='dba'	ADMINCLASS
WLP_ETL	‘etl’=ANY(roles)	ETLCLASS
WLP_BACKIPOPS	Activity=’Backup’	BACKUPCLASS
WLP_DEFAULT	true	DEFAULTCLASS

Table 3 - 2: Workload Policies definitions

4 Service Class and Work Policy Samples

Let us look at a company that has a requirement to control the usage of memory to avoid having unconstrained queries use large amounts of memory. The company also needs to set priorities for different queries according to role, user, time of day, elapsed time of query and combinations of these.

Table 4-1 **Service Classes definition** shows service classes with their settings. Table 4-2 **Workload Policies** shows workload policies and their associated service classes.

To set up workload management in Aster MapReduce Platform, create the service classes first and then create workload policies. When creating workload policies, you will associate the workload policy to a service class. See the **Aster Database User Guide** for more details.

Work policy “WLP_MSTR_USERS” in the first row of Table 4-2 has a predicate of “userName”. The workload policy definition also uses the “or” to associate it with more than one user. This workload policy is associated with service class “INSIGHT_USER” which is defined as Medium priority with a weight of 20 of the resources for that priority. The service class “INSIGHT_USER” has a memory limit of 70% for soft limit and 80% for hard limit.

Workload policies “WLP_SMITH” and “WLP_JOE” in rows 2 and 3 of Table 4-2 represent two different types of users. “WLP_SMITH” in row 2 is a general user and has a priority of High with a weight of 35 of the resources for that priority and memory limits of 60% for soft limit and 80% for hard limit. “WLP_JOE” in row 3 is an ETL user and it has a priority of Medium with a weight of of the resources for that priority and memory limits of 10% for soft limit and 20% for hard limit.

In row 4 of table 4-2 we define workload policy “WLP_ADMINOPS”. This workload policy has two predicates “stmtElapsedTime” and “userName”. The “stmtElapsedTime” predicate specifies the elapse time for the query. In this case specifies that if the elapse time is less than 5 minutes and the user is dba then apply service class “ADMIN” which provides High priority and 80% of resources available and no memory limits.

Work policy “WLP_ETL_NIGHT” in row 5 of Table 4-2 is for ETL queries submitted after 6:00 PM that have been running for less than 10 minutes. These queries will get a High priority with 60 weight, 30% memory soft limit and 40% memory hard limit.

Work policy “WLP_ANALYTICS“ in row 6 of Table 4-2 will apply to queries submitted by anyone with the role “analytics_role” that have been running for less than 10 minutes. These queries will get a priority of High with a weight of 15 of the resources and no memory limits.

The last row of Table 4-2 of is the WLP_DEFAULT workload policy which is required by Aster MapReduce Platform Workload management and is the policy that is used if none of the other policies apply.

Name	Priority	Weight	Memory soft limit	Memory hard limit
INSIGHT_USER	Medium	20	70	80
GENERAL	High	35	60	80
ETL_SLOW	Medium	10	10	20
ADMIN	High	80		
ETL_NIGHT	High	60	30	40
ANALYTICS	High	15		
DEFAULT	High	40	30	40

Table 4 - 1:   Service Classes definition

Workload policy name	Workload predicate	Service class name

WLP_MSTR_USERS	username='mstr_nc' or
	username='mstr_user'
		INSIGHT_USER

WLP_SMITH	username='smith'	GENERAL
WLP_JOE	username='joe'	ETL_SLOW
WLP_ADMINOPS	stmtElapsedTime < '5 mins' and username='dba'	ADMIN
WLP_ETL_NIGHT	stmtElapsedTime < '10 mins' and username='etl' and now()::time > '18:00:00.000000'	ETL_NIGHT
WLP_ANALYTICS	stmtElapsedTime < '10 mins' and 'analytics_role' = ANY(roles)	ANALYTICS
WLP_DEFAULT	true	DEFAULT

Table 4 - 2:   Workload Policies

5 Workload Management Test Environment

This section shows the results of experiments carried out at Aster Labs.

Let us take a look at a company that has three major groups using Aster MapReduce Platform for their day to day work:

- The first group is composed of upper management that runs reports to evaluate company performance and to balance budgets.
- The second group runs ad hoc reports and feed information to the first group.
- The third group is composed of scientists who develop different queries to be used by the other two groups.

The company requirement is that long running and resource intensive queries be controlled. This way the cluster can continue to perform and not have any memory problems.

This sample company was used as the basis of our testing as defined in the following sections.

5.1 Workload test

TPC-DS is an industry standard decision support benchmark that reflects a schema in the retail domain where sales are made using different channels. TPC-DS models several applicable aspects of a decision support system, including queries and data maintenance.

The test used a TPC-DS benchmark workload. Three groups of ten users each were used to run a set of queries. Each user ran the same queries, but in a different order, to ensure that each query was not run by more than one user at the same time. A total of 30 users were executing queries concurrently on a Teradata Aster Big Analytics Appliance 3.

The test proves the advantages of the use of workload management by providing more resources to users with workload policies associated to service classes that were assigned a high priority and allocated more resources. Users with workload policies associated with a service classes that had a lower priority got fewer resources. However, they were able to use more resources after the queries of the higher priority users had finished and more resources became available.

5.2 Memory control test

The memory control tests were conducted by running different queries with different memory limits to stress the system. Prior to Aster Database 5.0 the memory component of Aster MapReduce Platform workload management was not available and Aster MapReduce Platform clusters were vulnerable to run away queries that would consume all of the memory causing the cluster to crash. Aster Database 5.0 introduced memory control, this feature help protect the cluster and helps control cluster performance by assigning memory limits to different groups or users.

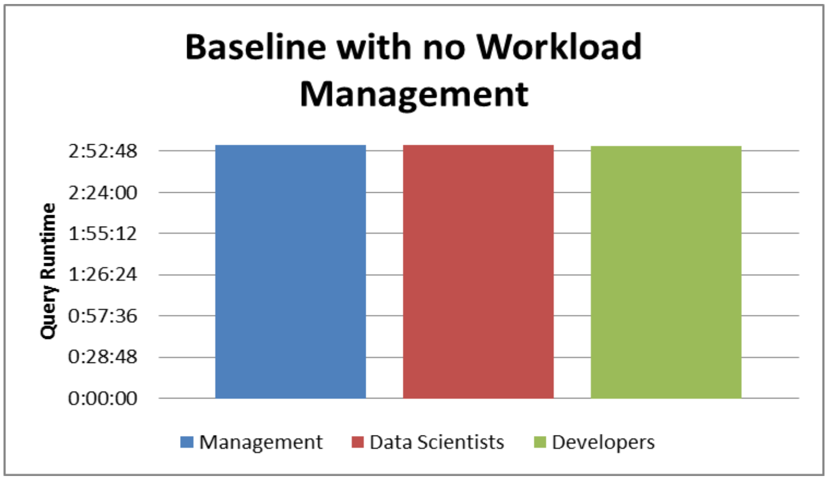
During the test, workload management provided consistent protection from overuse of memory. The queries that might have utilized more memory than allowed were associated to a service class with memory limits. These queries were canceled when the memory utilization was more than the memory hard limit.

6 Results

Two different tests were run: a TPC-DS like benchmark which we modified to add SQL-MR functions and a stress test with a combination of 1) queries that utilized large amounts of memory and 2) other queries with normal memory utilization. Following we will review the results for both tests.

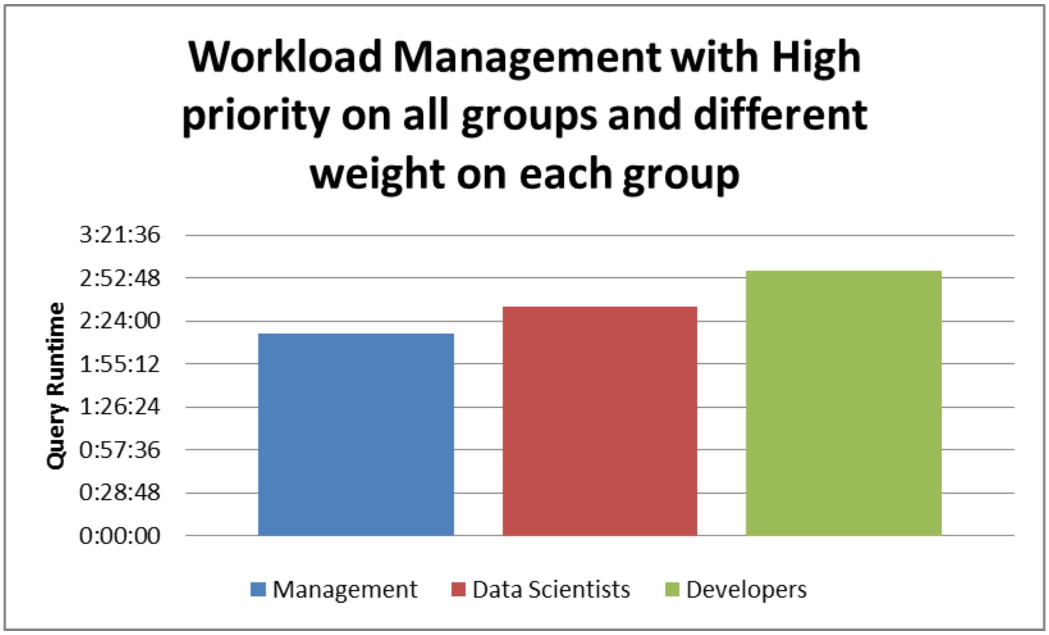
6.1 TPC-DS like benchmark results

This first graph shows the baseline numbers for our test. This run was done without any workload management defined and completed for all groups in the same amount of time.



Group	Query Runtime in HH:MM:SS
Management	2:57:28
Data Scientists	2:57:18
Developers	2:56:39

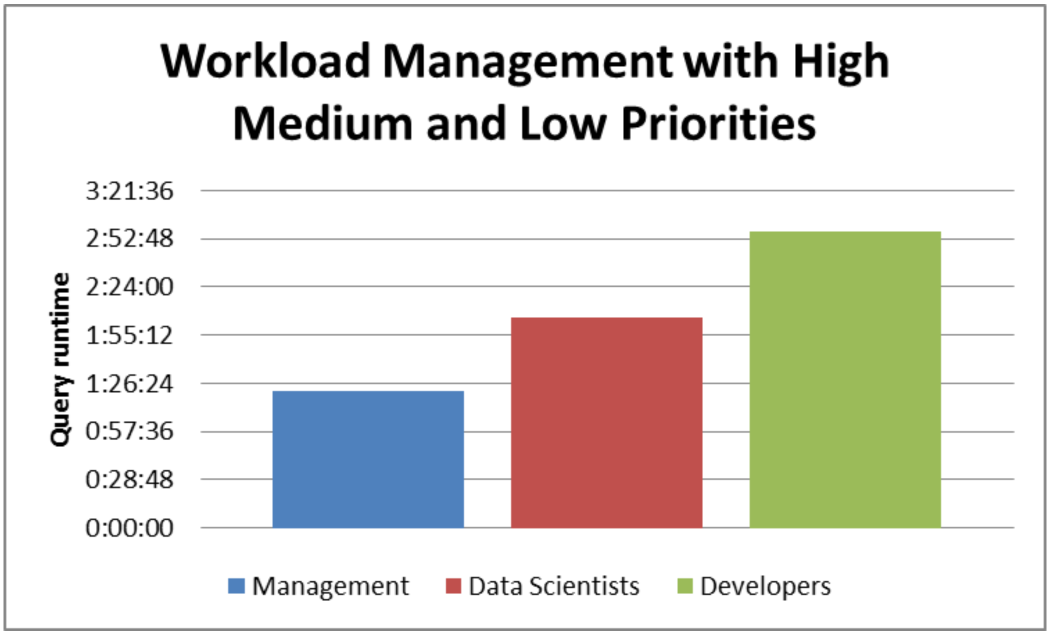
The next run was for all three groups with priority set to high and each group with a different weight. The users in group 1 (Management) had a weight of 75, while users in group 2 (Data Scientists) had a weight of 50 and users in group 3 (Developers) had a weight of 25. Since all three groups had a high priority, the resources had to be split between all users almost equality with a small variance for the weights. This is reflected in the results, which show a small advantage for users in group 1 and a smaller advantage for users in group 2, but no advantage for users in group 3.



Name	Priority	Weight
Management	3 (high)	75
Data Scientist	3 (high)	50
Developers	3 (high)	25

Group	Query Runtime in HH:MM:SS
Management	2:15:52
Data Scientists	2:33:52
Developers	2:57:53

In the following run all three priorities were tested, High, Medium and Low with a weight of 50%. As expected, the users in the group with High priority (Management) had the best response time, finishing the workload in less than half the time it took the users in the group with Low priority (Developers). Users in the group with Medium priority (Data Scientists) also benefited. Once the users in the High priority group finished running their workload, users in the medium priority group were assigned resources which helped them finish the workload faster than those in the Low priority group. The weight did not make much of a difference since it was consistent across all three groups.

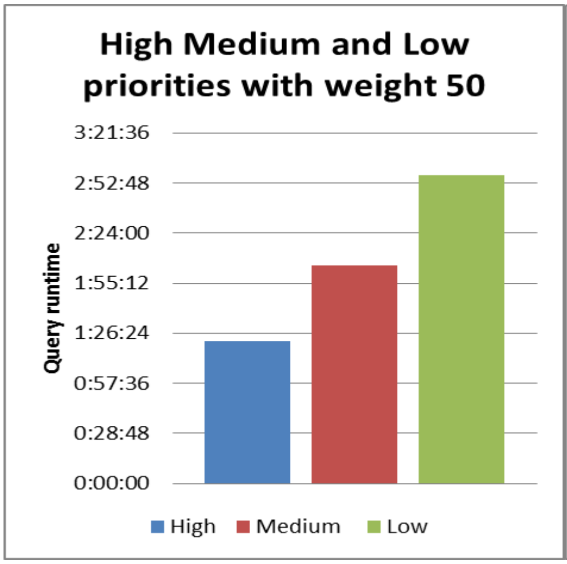


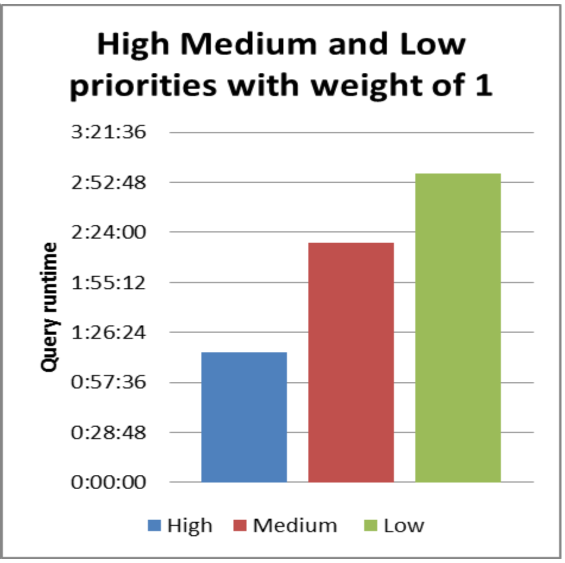
Name	Priority	Weight
Management	3 (high)	50
DataScientist	2 (medium)	50
IDevelopers	1 (low)	50

Group	Query Runtime in HH:MM:SS
Management	1:21:44
Data Scientists	2:05:31
Developers	2:57:17

The next graphs show a comparison between two runs with the same priorities but different weights. These graphs show similar runs. The only difference is the weight assigned to each priority. The graph to the left uses a weight of 50 for each priority, and the graph to the right uses a weight of 1 for each priority.

The High priority group with a weight of 1 shows an improvement of 8% over the High priority group with a weight of 50. This is because resources are not allocated to the Medium or Low priority groups when a weight on 1 was used. The group with the priority of High and 50 weight had to compete for resources with groups of priority Medium and priority Low with weight of 50.

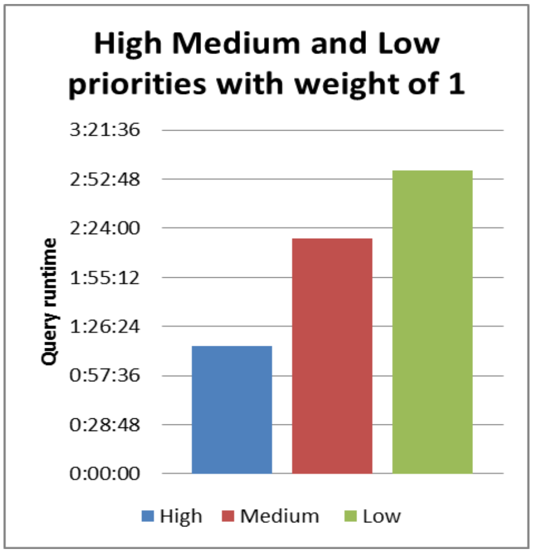


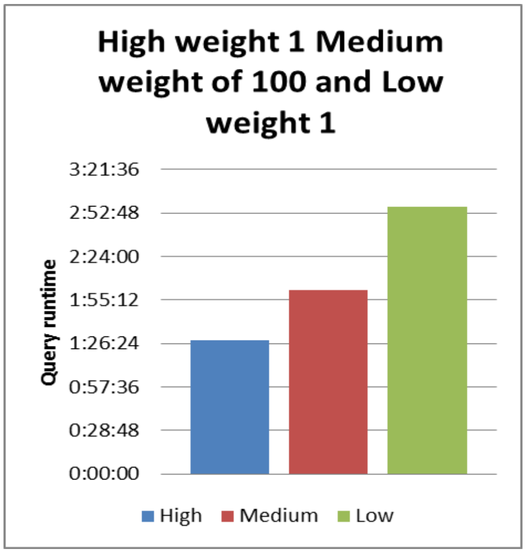


Priority with Weight 50	Query runtime
High	1:21:44
Medium	2:05:31
Low	2:57:17

Priority with Weight 1	Query runtime
High	1:14:58
Medium	2:18:04
Low	2:57:44

This set of graphs shows the difference made by the weight for the Medium priority users group (DataScientists). After assigning a weight of 100% to the Medium priority user group, the workload completed 11% faster. This caused the High priority user group to complete its workload 18% slower. When resources are limited, it is important to balance the usage by using setting the weight appropriately.





Priority with Weight 1	Query runtime
High	1:14:58
Medium	2:18:04
Low	2:57:44

Priority with Weight	Query runtime
High 1	1:28:42
Medium 100	2:01:46
Low 1	2:56:57

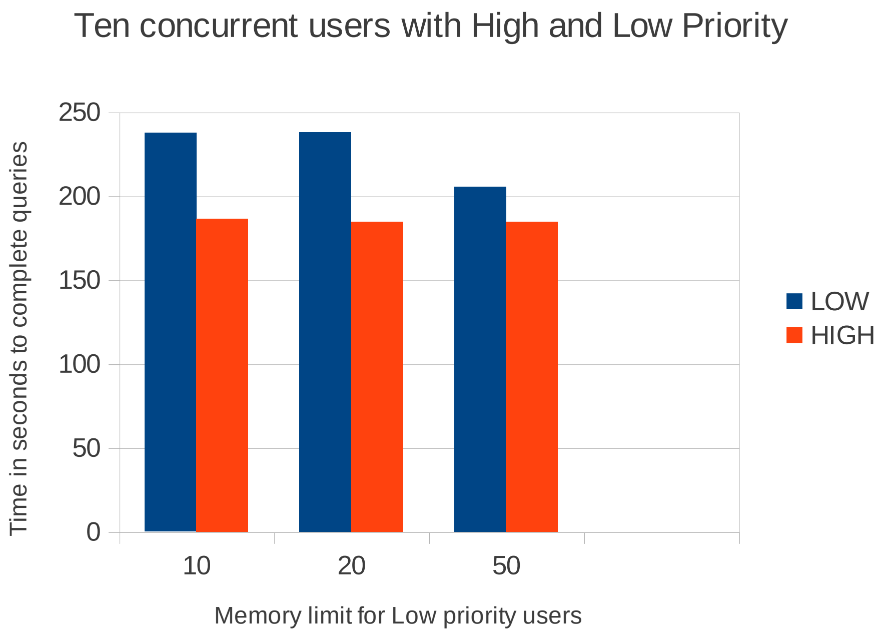
6.2 Memory Limit test results

Next we will review the advantages of using memory limits.

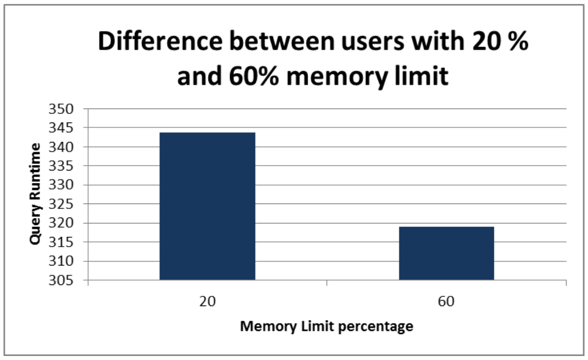
The following graphs show the test results when using a set of memory intensive queries to stress the system. The following graphs represent different memory limits, and show the time taken to complete a workload.

This first graph shows three different runs of ten concurrent users, five with low priority and memory limits of 10%, 20% and 50%, while the other five users have high priority and no memory limit.

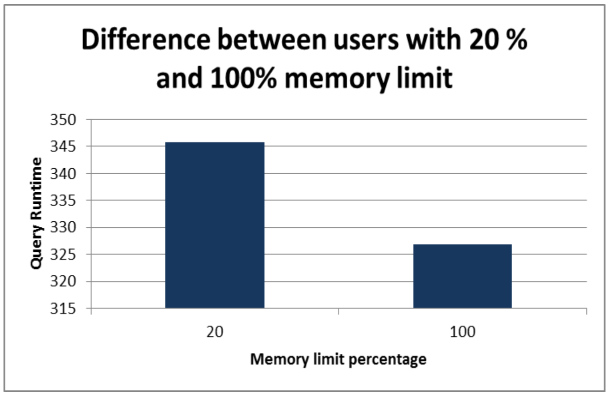
The results show that changing the memory limits produces different results in the time taken to complete the queries.



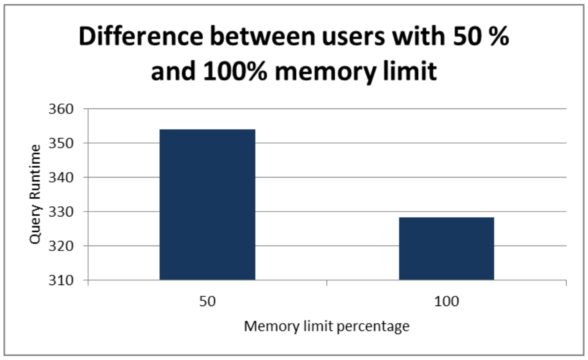
The next graph shows the difference between users running with a 20% memory limit against users running with a 60% memory limit.



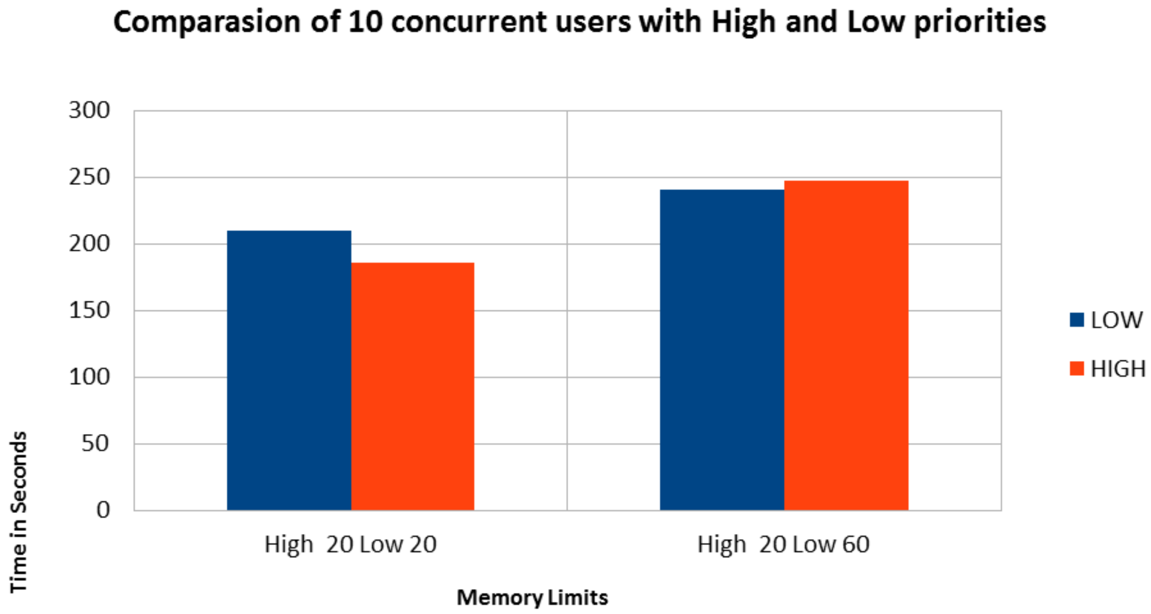
The next graph shows the difference between users running with a 20% memory limit against users running with a 100% memory limit.



The next graph shows the difference between users running with a 50% memory limit against users running with a 100% memory limit.



This graph shows two scenarios. First is a comparasion of High and Low priorities with a memory limit of 20% each. The second comparasion is of High priority with a memory limit of 20% and Low priority with a memory limit of 60%.The low priority with a memory limit of 60% was able to run the workload faster since this was a memory intensive workload.



7 Conclusion

Workload management provides a method for a business to prioritize their workloads and achieve predictable performance by utilizing the various workload management features available with Aster MapReduce Platform. It is evident from the above examples and tests that when these features are used correctly, better performance results, to suit customers' needs. It is very important for organizations and administrators to understand the relative importance of the queries and workloads in order to apply these features appropriately. The application of these features requires understanding of the data model, workloads and their impact on the business. Due to different organizational needs, there is no single boilerplate recommendation that would apply to all or most of the users of Aster MapReduce Platform System.

Workload management can help you attain the following goals:

- Prioritization of workloads
- Grouping of workloads
- Establishing predictable performance based on group and priority
- Improving user satisfaction

To achieve these goals the administrators need to understand the following;

- Grouping of users
- Allocation of resources
- Business Impact of the different groups
- Relative ordering of the workload
- When to constrain or eliminate non-significant business activity

Once the goals and workload management features are understood and implemented, the result is user satisfaction and a high business impact. As the above examples make clear, when workload management is used effectively, the Aster MapReduce Platform can finish the same workload more efficiently, with better end user satisfaction.

It is very important to understand the primary needs of the business and how critical one workload is verses another. Applying these policies without completely understanding the business needs prioritize less important business task over more important ones, which could have a disastrous impact on the business.

It is highly recommended to validate workload management rules on a test environment before introducing them into production.

Therefore, consider the following before creating your workload management policies:

- The needs of the business
- Business impact of workloads
- Utilization of system resources

Before making changes to a workload management policy:

- Consider any changes in the organization.
- Perform an impact analysis of any changes to workload management

Administrators will be faced with question while implementing workload management policies, including:

- Should I create a new policy group or assign different priority within a group?
- Should I change the CPU allocation or memory?
- What different priority levels should I define?
- How do I associate weights to different priority groups?

This paper presents some cases which answer some of these questions. Administrators should read the Aster Database User Guide to obtain a better understanding of workload management.

8 References

Aster MapReduce Platform user guide

Linux Man pages

Aster internal documentation