MUSE KNOWLEDGE

Muse Server Sizing

18 June 2012

Document Version 0.0.1.9
Muse 2.8.0.0

Notice

Disclaimer

Trademarks

museknowledge.com

260 Madison Avenue, 8th Floor
New York, NY 10016, U.S.A.
+1 (646) 740-4968 SUPPORT@MUSEKNOWLEDGE.COM

Table of Contents

1.0

Introduction

This document provides guidance on sizing a server to run the Muse Search environment. Suitable computers change all the time and the whole process of sizing is more of an art than a science. Thus this document must be read as providing guidelines, rather than specific recommendations.

Suggestions on server sizes can be found in section 4. It is not necessary to read the rest of the document before this section is looked at.

This document addresses the factors which need to be taken into account when trying to measure performance, and a proposed way of determining the suitability of a particular server for a particular workload.

1.1 Architecture

Muse is entirely a server-based system; there is no client software residing on the user's computer, other than a standard browser. Thus sizing is concerned solely with the server machine.

Muse needs three pieces of ancillary software to function, in addition to a suitable operating system. These provide the environment to run Muse and make its functions available across a LAN or the Internet.

Muse is written in Java and thus requires a Java Runtime Environment (JRE) to function. The JRE runs the whole time Muse is active and it is used every time a Muse instruction is actioned. Thus it is part of the direct overhead of Muse. Since Muse is a Java application, it needs Application Server software to start various Muse components. This server is active when Muse sessions are started or closed, so is not a continuing overhead. If Muse is interacting with users through Web Browsers, then it needs Web (http) Server software to format its output and send it to the user's browser. This runs continually while Muse is active.

The JRE is a third party system. Currently a version compatible with Sun's JDK 1.6 is required. The Application server and web server may be third party, or the ones included with Muse may be used

directly. Using the included servers ensures compatibility, provides slightly better performance, and makes installation and maintenance a whole lot easier.

2.0

Workload

To arrive at sizing figures it is necessary to define some standardized workloads. To do that we have made the following assumptions and definitions:

A logged on user is one who has started a Muse session whether s/he is actually submitting a request to Muse or not. Logged on users, in fact, spend most of their time reading results and deciding what to do next. From our experience we make the assumption that a logged on user is only active for about 10%–15% of the logged on session time.

An active user is one who has just submitted a request to Muse and is awaiting results. From the 10% figure above it follows that, at any one time, only 1 user of 10 logged on users is active (similarly if a 15% figure is used the number is 1 active from 7 logged on). Since it is the active users that place a load on the server it is those we consider as the base for a workload for server sizing.

However it is the number of logged on users that is usually of interest to the organization as these are the number of users "using" the system at any one time, and hence a measure of the service offered to those users. As the number of logged on users increases we increase our active figure from 10% to 15% to compensate for 'collision effects' where users contend for resources and to provide a more conservative estimate to accommodate usage fluctuations. To adjust for peak loads within a user population we adjust all these numbers upwards by 2x for sizing purposes. So the sizing use assumption is that from 20% - 30% (near enough one third) of the user population is active. Or, put another way, three times as many user will be logged on as are actually using the system.

2.1 Standard user workload

To allow repeatability of testing and to represent what real users do when interacting with Muse, we have devised a script which represents the type of work we have found is generated by an average user during a session.

This set of actions is the 'standard workload' we perform to measure an average responses time. There are 11 actions and the times for each action are averaged to give a 'Response Time' figure.

Alternatively for special circumstances we will measure the Response Time for just a single action (usually a search) with given constraints as long as that action adequately reflects the type of workload for which the server will be used.

2.2 Response times

Times for each action are measured from the time the user presses the key or clicks the button that initiates the action until the first data starts to display on the screen. All times are measured in seconds. A second time of interest is the 'Completion time' which is the time for the action to complete. This is easily recognized in a situation where the user is working through a browser, as the time until the word "done" (or equivalent) appears on the browser's status bar.

It is impossible to determine an absolute time (in seconds) for any response times as the searches in particular are dependent on the performance of the external systems being searched. Thus it is necessary to run the tests with one user to obtain a baseline response time and then repeat the tests with multiple users to determine the degradation as more users are added.

The actual number of seconds it takes to get results from a search can vary within wide limits even from the same installation and using the same search and database Sources. Network performance both local to the organization and remotely on the Internet, and the load on the target search engine can vary second by second, thus providing response times which can differ by 2:1 when the tests are performed one after the other. This is why the whole thing has to be done statistically to ensure the good and the bad are included as far as possible, and the, often much smaller, variation due to number of users can be found.

2.3 "Acceptable performance"

We assume that the response time performance for a number of simultaneously active users is considered acceptable when it is no greater the 125% of the response time for a single active user. That is: if the response time for a single user using a particular installation of Muse was 10 seconds, an acceptable response time for multiple users would be 12-13 seconds.

A more relaxed definition would be at 150% (or even 200%) of the single user value. Thus these would give acceptable times of 15seconds and 20 seconds on the base of 10 seconds for a single user taken as an illustration above.

It is, in reality, up to the organization to determine what an "acceptable performance" time is for its maximum user load. However it must be realized that Muse is providing extra processing on top of the

search response times of the external search engines, and it is impossible to achieve response times less than theirs, and impractical to require response times equal to theirs. Muse utilizes sophisticated parallel processing techniques to return results as quickly as possible, but it cannot do work in no time at all, specially with high user loads.

In testing Muse is able to determine the split of the time between the remote Source and its own processing. This allows the overhead of the Muse processing to be calculated and this figure is often used to determine the acceptable load on the system.

2.4 Maximum user load

The maximum number of simultaneously active users that a Muse server can support when the average of their response times is less than or equal to the acceptable performance time is considered the maximum user load.

As discussed actual response times in seconds cannot be used here as they depend not only on the Muse server and its workload, but also on external factors on the local area network, the Internet and the remote Source servers. All of these will not be affected by any change in the capabilities of the Muse server computer. Thus a base response time must be set for a particular installation and set of Sources for a single active user, and the acceptable performance time calculated from that. Because the external conditions change over time (e.g. the Internet transmission time changes because of other traffic), it is necessary to continually monitor the base response time during any performance testing. Thus tests involving larger numbers of simultaneous active users must be interspersed with tests involving only a single active user, and the average of those single user tests be used as the base response time.

Experience has shown that running a single user test for every 3 or 4 tests with larger numbers of users is adequate to ensure the base is an accurate measure.

2.5 Throughput

In particular circumstances with a constrained set of conditions (such as the type of actions performed for users, and the actual scope of the Sources used for searching) it is possible and sensible to measure throughput on the system and to measure the Response Time for particular throughput levels.

For these numbers to be meaningful the environment in which they are obtained, must closely measure the working environment and both must be quite severely constrained. Thus these numbers are not appropriate where the end user has control of the actions performed at any time.

3.0

Typical Results

We ran the tests on a number of different servers on a LAN measuring both single user performance and also the average response times for different numbers of active users. The users were simulated by a set of scripts developed for just this purpose which allow a specified number of users to be emulated by a single PC, and the response times to be measured.

In this particular set of tests the single user response time (the baseline response time) was 3.5 seconds. Thus the 125% time is 4.4 seconds and the 150% time is 5.25 seconds. So, depending on the Acceptable Performance criterion (125% or 150%) chosen, the Muse server will be considered to have reached its maximum user load when the average response time just does not exceed either 4.4 or 5.25 seconds.

We tested in increments of 5 active users (except for the smallest machine) and the table below records the number of active users and the response time which was closest to the target response times. The next increment of 5 pushed the response time over the target times.

For each type of computer the Acceptable Performance level is reached at the following number of users:

	125%	150%
P4/1.0GHz	4.5 sec 3 users	5.1 sec 4 users
P4/2.0GHz	4.0 sec 5 users	4.8 sec 10 users
P4/3.0GHz 2x	4.1 sec 10 users	4.8 sec 15 users
UltraSPARC III	4.0 sec 15 users	4.8 sec 25 users
Xeon/2.66GHz 2x	4.3 sec 20 users	5.2 sec 30 users
Xeon/2.66GHz 4x	4.1 sec 25 users	4.9 sec 35 users
Sun Niagra 1.0GHz	4.2 sec 30 users	5.2 sec 40 users

Table 1.

From the above it is clear that even at a more 'relaxed' Acceptable Performance level of 150% of single user response, the P4/1.0GHz machine is nearing its limit at 4 active users. At an extrapolated 20 active

user loading, its response time will be nearly 20 times that of a single user (i.e. about 70 seconds for a single user to get their response on average, some will be much longer).

The 3.0GHz machine obviously is well capable of handling 10 active users, and still gives very acceptable performance for 15.

The SPARC based computer is easily able to handle the projected workload for 25 active users, as is a dual core Xeon machine. The quad core Xeon and the Sun Niagra based machines are the current preferred choice for servers.

Multi-processor machines are capable of handling larger loads as would be expected, and they have larger amounts of memory to go with the extra processors. Of course they must be running an operating system which is able to utilize the multiple processors (Linux in this case, or Solaris).

However conditions and computers change and these figures must be taken as a guide only.

4.0

Recommendations

When using these recommendations, please carefully consider the assumptions behind them and determine that they are consistent with your experience and situation.

This document has explained the architecture of Muse and how its performance is tested. The following are Suggested types of servers for various workloads, assuming an unpredictable mix of work from the users:

Logged on Users	Active Users	Server Type	Memory (RAM)
5	1	P4/1.0GHz	512MB
10	2	P4/2.0GHz	512MB
25	5	P4/3.0GHz 2x	1GB
50	10	Xeon 2.66GHz 2x	2GB
125	25	Xeon 2.66GHz 2x (two CPUs)	4GB
200	50	Xeon 2.66GHz 4x Sparc IIIi (two CPUs)	8GB
350	100	Sparc IVi Sparc Niagra AMD Opteron Xeon (4 CPUs and up) (speeds equiv to (3.0GHz Xeon and up)	8GB to 16GB

Table 2.

The server types are given in terms of popular processors and their speed. Any similar processor of comparable performance can be used. These CPU brands are not specific recommendations, just

examples.

In a large load environment multiple machines can be used as long as there is load balancing available to spread the load across the multiple machines. This has the advantage of providing redundancy as well. All other factors being equal (cost mainly) multiple smaller machines are recommended over a single bigger one. Muse has been installed and used with the Resonate load balancing software, and also in an 'rsync' based mirroring environment, and utilizing the Coyote Point load balancer hardware. All set ups perform load balancing on a 'user session' basis.

Muse is multi-threaded software and is thus able to take advantage of multiple cores in a single CPU efficiently. A dual core (Intel or AMD) CPU can be expected to handle about 1.6x the load of the single core equivalent. Multiple thread CPUs such as the Sun Niagra (capable of handling 32 simultaneous threads) are an advantage and show about a 7%-10% performance advantage against similar cost x64 ar chitecture machines.

The server memory is an important factor. As long as it is in line with the above guide, it is not the limiting factor, but it is an important part of expansion for Muse capabilities, and a computer with the ability to add memory should be preferred to one which cannot.

Processor speed has a major effect on Response Time as might be expected, and a lesser effect on throughput, which is often limited by network bandwidth (see below) and other factors. Thus increasing the power of the CPU(s) in a computer may well reduce Response Time but have negligible effect on the number of users supported. Increasing the number of computers (or, more accurately, the number of cores) will have the effect of increasing the number of users supported, but have little effect on Response Time. This assumes there are not other limiting factors.

Server hard disk capacity is used mostly by storage of user Muse WorkRoom result sets, and the amount of space allocated to a particular user can be controlled within Muse. A suggested WorkRoom capacity is 1Megabyte per user. This means a 120GB disk can handle approximately 100,000 users. This number is the total user population, not those logged on or active at any one time, as it stores data for all user of Muse. Disk speed is not important as Muse makes little use of a hard disk (unless the operating system is heavily using its swap space to compensate for lack of main memory).

Networking reliability is critical for Muse. As all user actions will generate multiple network messages, a reliable (no dropped packets) connection is essential. Messages are typically small so bandwidth is not usually an issue, but a broadband connection (256Mbps or greater) is necessary for reasonable performance for a 20 user installation. Increase it pro rata for higher numbers of active users. Remember that this is the network connection for the server to connect to the sources.

As an aid to calculating the necessary bandwidth we have noted that the average response from a

Searched Source is a page of about 50KBytes. Thus a single user searching 5 Sources will generate 125KB in traffic. The outbound traffic is generally small (at about 10KB) in comparison. However with Ajax clients the amount of traffic to the client is increased quite dramatically, even though special small traffic messages are used. It is suggested that about 50KB be allowed for this traffic.

Thus the number of users and the frequency with which they are searching (which is effectively the number of Active Users) and the number of Sources searched and the response time of the Sources will allow a calculation of the necessary bandwidth. As an average "rule of thumb" figure it is reasonable to assume 10 Active Users will need about 200KB/sec = 1.6Mbps, adding a factor for peak loads and collisions the 10 Active Users will need about a 3.5Mbps connection for sustained use. These numbers vary considerably depending on the assumptions made (like the average number of Sources searched and the average response time of those Sources – 5 Sources responding in 10seconds for the figure quoted above) and the above MUST be taken as a rough guide until actual data about usage can be obtained.

The largest actual workload recorded (not necessarily achieved, that is higher) is with 500 user sessions running on a 4 processor Sun Sparc machine with 8GB of RAM with little performance degradation.

The highest throughput measured in our test environment for single CPU machines is as follows (repeated random searches of 3 Sources per search with average 1.8 sec response times):

P4 @ 3.0GHz	216 searches/min	@ 13.0 sec response	1GB memory
Xeon @ 3.0GHz	213 searches/min	@ 3.5 sec response	1GB memory

Table 3.

This shows the advantage of the more powerful CPU in reducing response times with little effect on the throughput.

When using these recommendations, please carefully consider the assumptions behind them and determine that they are consistent with your experience and situation.

