Paper 3275-2015

Great Performances: SAS® Visual Analytics Performance Monitoring and Enhancement

Jonathan Pletzke, University of North Carolina at Chapel Hill; David Franklin, SAS Institute Inc.

ABSTRACT

At the University of North Carolina at Chapel Hill, we had the pleasure of rolling out an enterprise-wide SAS® Visual Analytics environment in 10 months, with strong support from SAS®. We encountered many bumps in the road, moments of both mountain highs and worrisome lows as we learned what we could and could not do, and new ways to accomplish our goals.

Our journey started in December of 2013 when a decision was made to try SAS Visual Analytics for all reporting, and only incorporate other solutions if and when we hit an insurmountable obstacle. We are still strongly using SAS Visual Analytics and are augmenting the tools with additional products. Along the way, we learned a number of things about the SAS Visual Analytics environment that are gems, whether one is relatively new to SAS or an old hand.

Measuring what is happening is paramount to knowing what constraints exist in the system before trying to enhance performance. Targeted improvements will help if measurements can be made before and after each alteration. There are a few architectural alterations that can help in general, but we've seen that measuring is the certain way to know the problems and then whether the cures were effective.

INTRODUCTION

There are many things that can result in a well performing system and a happy user base. The responsibilities are shared by everyone who touches the system in some way. A system can be well architected for a particular need, but if users are not prevented from hurting the system or not properly trained, the system can be a failure. For example, Figure 1 shows a dialog that is presented to the users of the SAS Visual Data Builder portion of SAS Visual Analytics. It seems an innocuous enough warning, and depending on how one views the word "large" can give the user pause, or let the user blast straight through without further thought or notification.

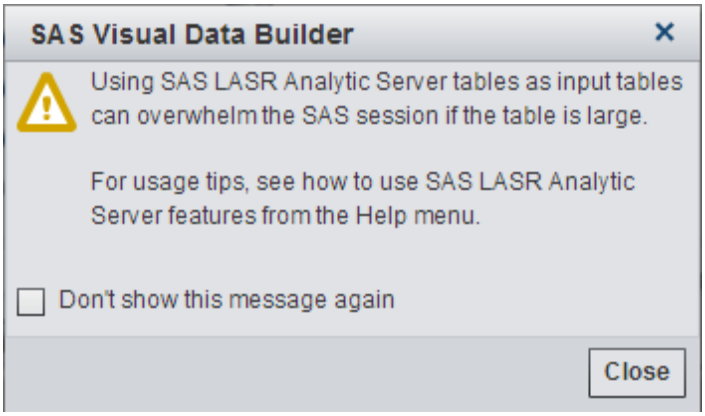


Figure 1. A simple dialog box that can destroy performance on SAS Visual Analytics even if built according to sizing

When we first started our implementation, we thought we would bring in each of our component tables separately and then work with them in memory. With that strategy and the thought that none of our tables, on its own is all that large, many developers passed straight through this warning. And the system got very, very slow. The online web interface became unresponsive and anywhere from 5 to 25 tables were joined together, exercising the standard SASApp service and using the sizing-default single channel RAID hard disk.

What we have learned and hope to share in this paper are some ways, certainly not the only ways, to measure performance and resolve some of the issues that we have seen over the last year primarily in SAS Visual Analytics 6.4. We will also introduce an approach that we are using on SAS Visual Analytics 7.1 and anticipate continuing for some years to come.

SIZING

Since we didn't know more than an order of magnitude how much data we'd end up with, we were given two sizing documents and found a way to balance the risks and costs between the two extremes. We had two estimates, one for our largest table being 5GB and one for the largest being 50GB. Our estimates of data size were related to what we knew of our existing database table sizes, and we assumed that we would join the tables the same way we would in the database. We did not want to spend more on infrastructure than we thought necessary, so we went with the lower estimate of 6 machines at 16 cores, 256GB of fast RAM, and 600GB disk. We hedged a little by going with the newer chipset with 20 cores, and 320GB of RAM along with a larger 900GB disk that we set up in a RAID mirrored pair to avoid downtime and emergency calls on the disk.

We learned through our experimentation that our head node was underutilized in CPU and RAM, but the SAS Visual Data Builder part of SAS Visual Analytics could interfere with the web server as large jobs were running to local disk. Separately, we noticed that our worker nodes appeared to be moderately busy in bursts, going from no use to 80% use and then back to zero within seconds, which is what we believe to be relatively healthy behavior when trying to multiplex multiple users viewing basic reports.

We also identified that our approach to row-level filtering (aka row-level security) within Visual Analytics requires excess CPU as each row in the table is checked for visibility to each user. Our current algorithm, under review for improvement, searches each row for a UserID to appear anywhere in one or more 4KB text fields and is shown in Figure 2. A second factor relevant to sizing is the current limitation in joining multiple in-memory tables to one key, in the SAS Visual Analytics "star schema", and our inability to readily create a composite key for each row that represents the joins coupled with what we understand to be a slowdown with an in-memory "view" that is more CPU intensive. This has resulted in tables that have gotten extremely large, over 500GB, in comparison to the 5GB component tables that would be joined together in a relational database. A second sizing exercise resulted in the recommendation that we increase our size by 50 additional servers, which is out of the question for our purposes. Instead we have relaxed the row-level security requirement and narrowed the user base for the largest tables, as well as swinging the underutilized servers to augment the production servers and increase the size with modest expense.

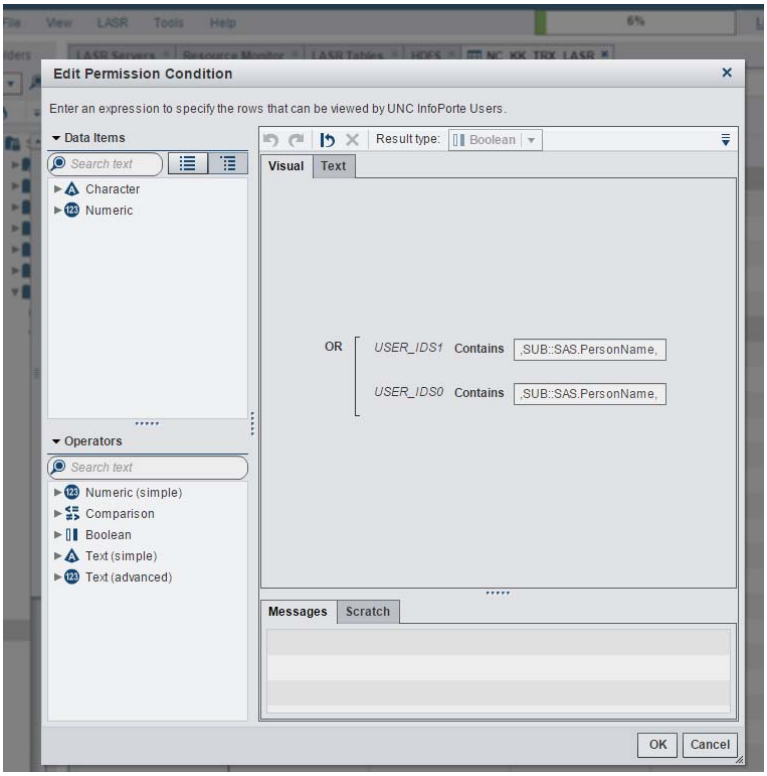


Figure 2. A resource intensive row level security implementation

CPU, MEMORY, STORAGE, NETWORKING

We started thinking we'd be able to do our work within a few small virtual machines and ended up knowing that we're happy in a separate bare-metal environment that rivals a VMware farm. We learned about the difference in storage and the layout of the software on the SAS Visual Analytics "head node" versus "worker nodes" and crafted a plan to be able to enhance our infrastructure rapidly if necessary. This included ordering all of our infrastructure in the same 20 Core CPU, with compatible 900GB RAID storage. RAM varied between the roles of the machines - this was because we would not know the exact data sizes nor the usage patterns that might necessitate the need for the machines, and waiting for an order to come in would take months. As you'll see in the sections below, we started to see CPU saturation because of the oversized data, needed some storage tweaks, and continue to have adequate RAM and network bandwidth.

SECOND RAID CHANNEL OR A SEPARATE SASAPP SERVER

Different from the default configuration and the configuration provided by the SAS EEC sizing effort, we identified a solution to offload the work from the SAS® Visual Data Builder as necessary for the "creative" use of the tool by our power users.

We had a choice between simply providing a second RAID channel for disk to overcome the IO needs of the traditional SAS processing that takes place, but chose instead to engage a separate server to meet this need. This decision was based on the observation of the disk I/O levels during SAS Visual Data Builder runs versus our benchmark that was identified during initial installation.

We re-deployed hardware that we had earmarked for other purposes in our SAS configuration. For our Test environment, we used like hardware to the other nodes, but with half the RAM, and augmented it with a second RAID channel on 2 additional drives. In Production, the server had the same CPU/RAM specs, but had 16 drives, across two RAID channels, which we split for executables/logging versus SASWork.

We knew we had to do something because the SAS Visual Analytics applications became less responsive and finally unresponsive during peak runs of the SAS Visual Data Builder queries against in-memory tables. We primarily used the Linux utility iostat to diagnose this problem by watching the IO during problem times and also during quiet times, and also comparing load using iostat while running the SAS provided utility iotest.sh, which we ran as "iotest.sh –l 5 –t /sasdata/ -b 2179072 –s 64". We were unable to use other means, such as SAS Environment Manager, because the server was getting non-responsive on the web side while running large transformations (refer back to Figure 1).

We also tested the secondary storage on the new app server to benchmark the best storage configuration. We don't want just raw mirrored disks, we want some redundancy so that we increase up-time and reduce emergencies. So our team tested RAID 10 and RAID 5 to see which would work best given our exact hardware. We could have gone with just a recommendation, or the theoretical differences, but thought that a test would be certain, and since it is easy with our selected hardware to re-provision the drives in different configurations, it would not be too much work. We found that the RAID 5 outperformed the RAID 10 in our 8-drive configuration.

As we refined what we were doing and how it was being done, the user population reduced their load by stopping pervasive use of the in-memory table transformations, and we are in the process of reconfiguring our hardware. We will end up moving off the second SASApp server, back to the first, but augmenting with a second RAID channel because of the observed (and projected) minimal CPU and RAM needs, and the occasional need for the SASApp server to run.

SAS ENVIRONMENT MANAGER

We developed our own best practices for this tool, including initial setup to go beyond the default installation, dashboards, alerts, and guidelines on how to interpret the data to spot trends in utilization.

The main task was to understand the configuration options and the proper way for us to use the tool. We focused on interactive use rather than alerts, knowing that we may want to selectively add alerts later.

In order to get the most value from the tool, we installed the SAS Environment Manager Agent on all of the machines. Not knowing how easily this might be accomplished with the install tool, we went the route of copying the directory install to all of the nodes with a simple rsync command. The path necessary to copy on our infrastructure was the /Lev1/Web/SASEnvironmentManager/agent-5.0.0-EE/ directory and all children. Once that was copied, you simply edit the /Lev1/Web/SASEnvironmentManager/agent-5.0.0-EE/conf/agent.properties file to change the agent.setup.agentIP to match the local host name. One thing to remember is to ensure that you run the program as the 'sas' user rather than root, which caught us many times as we were bouncing around the machines and setting up autostart. If you accidentally run it as root, the logfiles in /Lev1/ will save as root, and then when you correctly run it as 'sas', it cannot write the the logfiles in /Lev1/Web/SASEnvironmentManager/agent-5.0.0-EE/log and you need to chown and chmod the files back to 'sas' (or simply delete them from the root account which is what we did). The specific commands that we used are provided in the figure below:

```
Making more agents on the additional VA nodes

 mkdir /opt/config/Lev1

 mkdir /opt/config/Lev1/Web

 mkdir /opt/config/Lev1/Web/SASEnvironmentManager

 rsync -avz sasva0p:/opt/config/Lev1/Web/SASEnvironmentManager/agent-5.0.0-EE
/opt/config/Lev1/Web/SASEnvironmentManager/


 mkdir /opt/sashome/SASPrivateJavaRuntimeEnvironment/

 mkdir /opt/sashome/SASPrivateJavaRuntimeEnvironment/9.4

rsync -avz sasva0p:/opt/sashome/SASPrivateJavaRuntimeEnvironment/9.4/jre
/opt/sashome/SASPrivateJavaRuntimeEnvironment/9.4/
```

```
Edit the configuration file:

vi /opt/config/Lev1/Web/SASEnvironmentManager/agent-5.0.0-
EE/conf/agent.properties

Change the agent.setup.agentIP to the local host name, consider adding the
following line (and adjusting memory parameters) if out of memory errors:

# Memory settings to increase memory footprint (from vmware page "reduce
agent memory footprint")

agent.javaOpts=-XX:MaxPermSize=128m -Xmx4096m -Xms4096m -
Djava.net.preferIPv4Stack=true -XX:+UseConcMarkSweepGC

2


Remove the data directory:

rm -f /opt/config/Lev1/Web/SASEnvironmentManager/agent-5.0.0-EE/data


Start it all up:

On a single host:

/opt/config/Lev1/Web/SASEnvironmentManager/agent-5.0.0-EE/bin/hq-agent.sh
start


For the Grid:

 /opt/TKGrid/bin/simsh /opt/config/Lev1/Web/SASEnvironmentManager/agent-
5.0.0-EE/bin/hq-agent.sh start
```

Once the agents are all running, the next step is to ensure that they are delivering data to the SAS Environment Manager server. There are troubleshooting steps that can be found both within the SAS documentation and also by searching the Internet for ‘Hyperic troubleshooting’ and so forth. One of the most common fixes for each agent is to stop the agent, delete the /Lev1/Web/SASEnvironmentManager/agent-5.0.0-EE/data directory (or rename to data.bak or such), then start the service again from the command line and see that it is able to configure and connect.

For us, the two most useful areas are the customized dashboard and the Resources tab. We created a customized dashboard, and synced in a few users from the SAS Metadata server to see it. We setup the additional users as Guest role and made all dashboard edits to the Guest dashboard. Our monitoring of resources in this tool is focused on the SAS specific needs to have SASWORK available with plenty of disk space, along with RAM, CPU, and both Disk I/O and Network I/O. We monitor it through the central Xenoss monitor as well, but the SAS Environment Manager provides access to a different, broader set of users, with the basics as well as the SAS specific service monitoring.

Figure 3 shows the List View of the resources. You see this by clicking on the Resources menu tab and clicking browse. You may alternately see Figure 4 which is the same resources, but in the Chart View. You can alternate between the two by clicking the button on the right that is sandwiched between the white background that starts with Platforms and the server names.

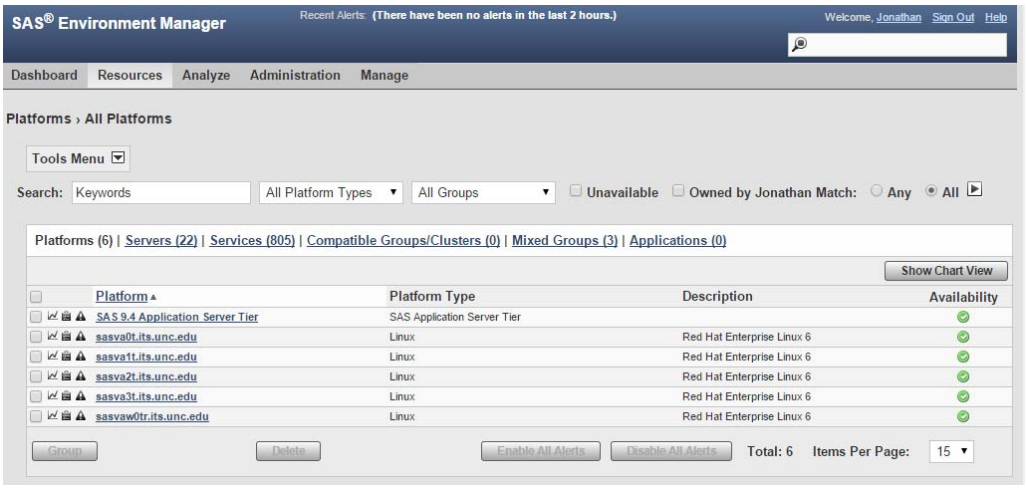


Figure 3. The List View top level resource tab in SAS Environment Manager

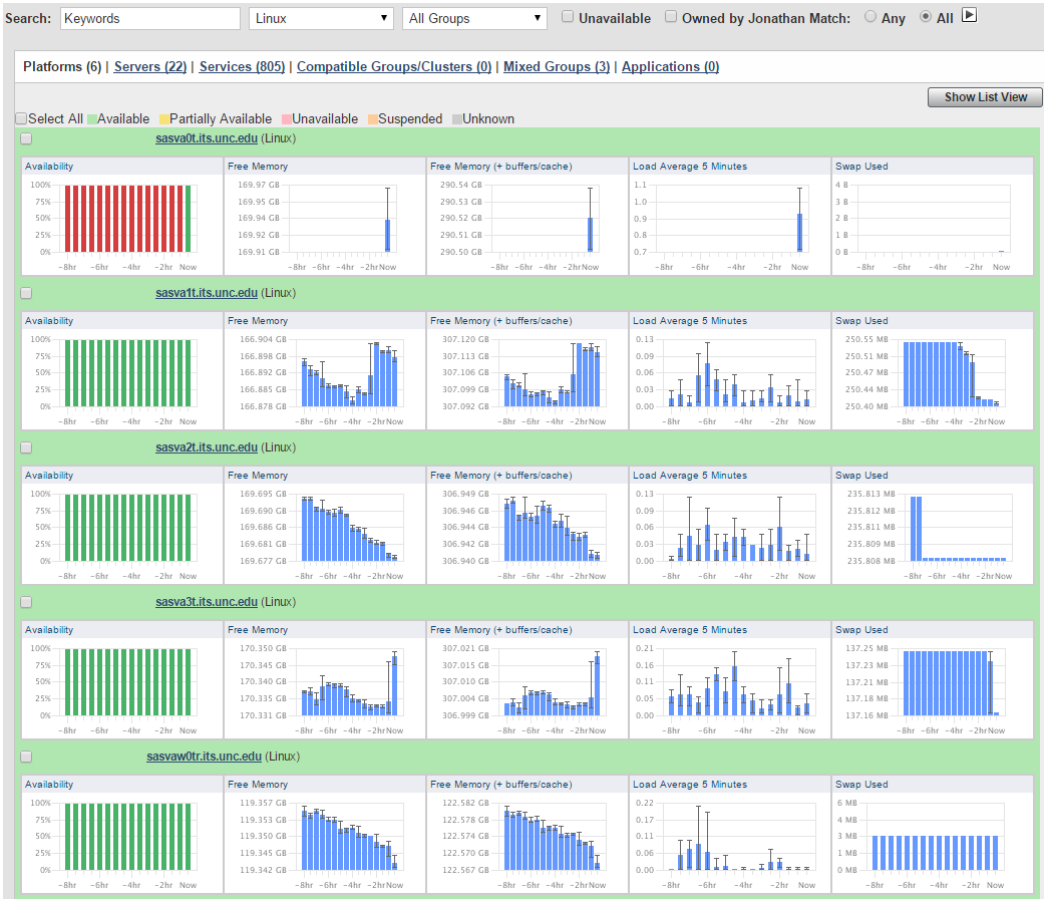


Figure 4. The Chart View of Platforms in SAS Enviornment Manager

Our Guest dashboard appears in the several figures below and is built simply by adding a few charts to the left side, and the appropriate metrics on the right, with the maximum number of entries sorted by most constrained to least constrained. Figure 5 shows the box where you select the Guest Role for the displayed dashboard, which you can also set as the default for your user account.

The rest of the dashboard components were added by selecting the drop down menu “Add content to this column” in the right hand column, and choosing the Metric Viewer option. The various resources were selected as shown below, with the highest percentage or number shown first, then the others in

descending order.

Figure 6 shows the Disk Use percentages from most used to least used, which are not instantaneous, but rather captured at defined intervals that are configured in SAS Environment Manager (we used the defaults). We saw high use in our test environment, and then the users were complaining that there were errors, which turned out to be their desire to load each table twice, and we will be installing additional disk to hold that data in HDFS so that it is pulled on-demand into RAM for the smaller population using our test environment.

Figure 7 shows the RAM, and since we are using the HDFS storage to hold tables, we see RAM percent use decrease as more activity is occurring in the environment, then it pulls back overnight as tables are reloaded into HDFS.

Figure 8 shows the CPU load average, which based on the bursty nature of the load, appears as a lower number than peak, unless of course the environment is completely CPU bound. Figure 9 shows the Disk IO Reads, which we don't see very high, since the data set segments are loaded as needed, and our RAM exceeds the size of data sets on disk. We have seen this number go up momentarily while the largest data set (500GB+) is being loaded into memory, which takes a minute or two. Figure 10 shows the Disk IO Writes, which for the Hadoop nodes picks up while they are being written, and for the second SASApp server while significant jobs are running. If a more detailed set of data over time is required, a click on any particular line will take you to the graph, and example of which is shown in Figure 11 for free memory which shows memory use in a busy day from morning onward. Figure 12 shows a busier view of the disk writes as significant processing is occurring, driving SASWORK use on the second SASApp server.

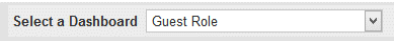


Figure 5. The Guest Role dashboard

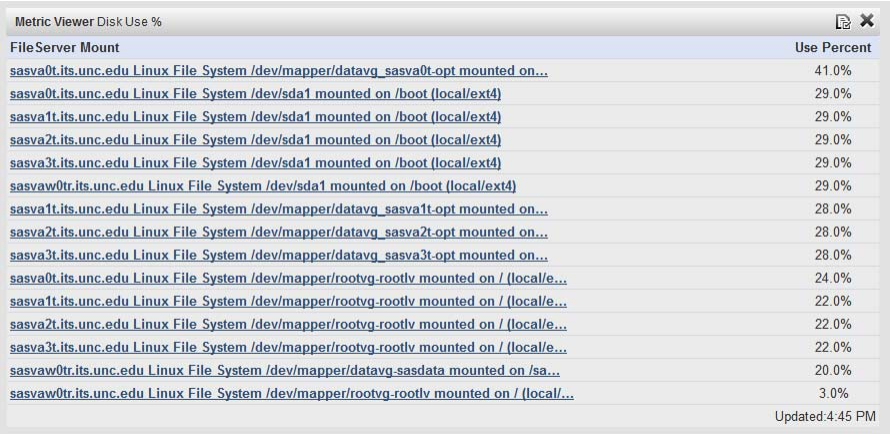


Figure 6. Disk Use percentages in Guest Dashboard

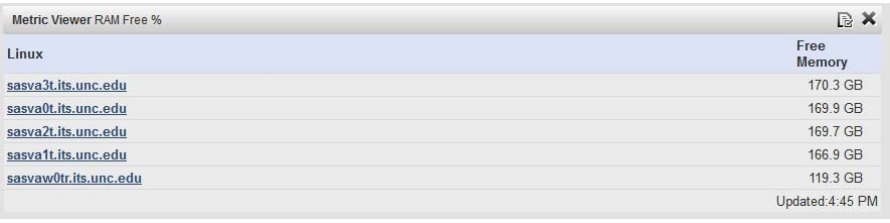


Figure 7. RAM Free % in Guest Dashboard

Metric Viewer CPU Load Average

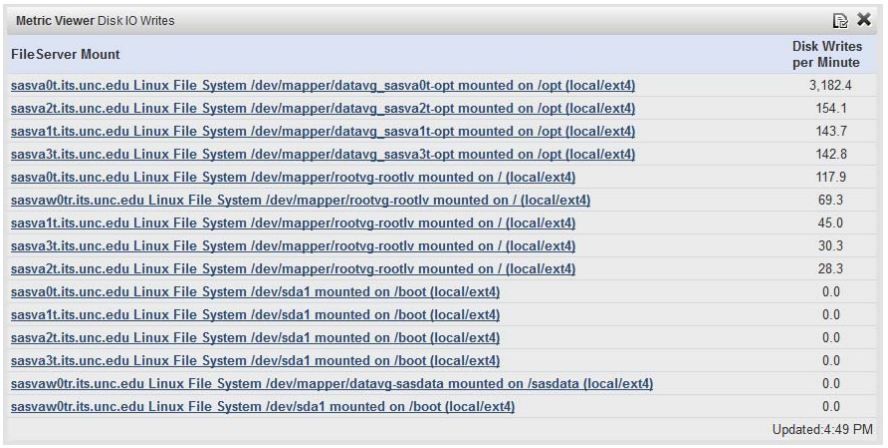
Linux	Load Average 5 Minutes
sasva0t.its.unc.edu	0.3
sasvaw0tr.its.unc.edu	0.0
sasva1t.its.unc.edu	0.0
sasva2t.its.unc.edu	0.0
sasva3t.its.unc.edu	0.0
	Updated:4:47 PM

Figure 8. CPU Load Average in Guest Dashboard

Metric Viewer Disk IO Reads

File Server Mount	Disk Reads per Minute
sasva0t.its.unc.edu Linux File System /dev/mapper/datavg_sasva0t-opt mounted on /opt (local/ext4)	0.0
sasva0t.its.unc.edu Linux File System /dev/mapper/rootvg-rootlv mounted on / (local/ext4)	0.0
sasva0t.its.unc.edu Linux File System /dev/sda1 mounted on /boot (local/ext4)	0.0
sasva1t.its.unc.edu Linux File System /dev/mapper/datavg_sasva1t-opt mounted on /opt (local/ext4)	0.0
sasva1t.its.unc.edu Linux File System /dev/mapper/rootvg-rootlv mounted on / (local/ext4)	0.0
sasva1t.its.unc.edu Linux File System /dev/sda1 mounted on /boot (local/ext4)	0.0
sasva2t.its.unc.edu Linux File System /dev/mapper/datavg_sasva2t-opt mounted on /opt (local/ext4)	0.0
sasva2t.its.unc.edu Linux File System /dev/mapper/rootvg-rootlv mounted on / (local/ext4)	0.0
sasva2t.its.unc.edu Linux File System /dev/sda1 mounted on /boot (local/ext4)	0.0
sasva3t.its.unc.edu Linux File System /dev/mapper/datavg_sasva3t-opt mounted on /opt (local/ext4)	0.0
sasva3t.its.unc.edu Linux File System /dev/mapper/rootvg-rootlv mounted on / (local/ext4)	0.0
sasva3t.its.unc.edu Linux File System /dev/sda1 mounted on /boot (local/ext4)	0.0
sasvaw0tr.its.unc.edu Linux File System /dev/mapper/datavg-sasdata mounted on /sasdata (local/ext4)	0.0
sasvaw0tr.its.unc.edu Linux File System /dev/mapper/rootvg-rootlv mounted on / (local/ext4)	0.0
sasvaw0tr.its.unc.edu Linux File System /dev/sda1 mounted on /boot (local/ext4)	0.0
	Updated:4:47 PM

Figure 9. Disk IO Reads in Guest Dashboard



Metric Viewer Disk IO Writes

File Server Mount	Disk Writes per Minute
sasva0t.its.unc.edu Linux File System /dev/mapper/datavg_sasva0t-opt mounted on /opt (local/ext4)	3,182.4
sasva2t.its.unc.edu Linux File System /dev/mapper/datavg_sasva2t-opt mounted on /opt (local/ext4)	154.1
sasva1t.its.unc.edu Linux File System /dev/mapper/datavg_sasva1t-opt mounted on /opt (local/ext4)	143.7
sasva3t.its.unc.edu Linux File System /dev/mapper/datavg_sasva3t-opt mounted on /opt (local/ext4)	142.8
sasva0t.its.unc.edu Linux File System /dev/mapper/rootvg-rootlv mounted on / (local/ext4)	117.9
sasvaw0tr.its.unc.edu Linux File System /dev/mapper/rootvg-rootlv mounted on / (local/ext4)	69.3
sasva1t.its.unc.edu Linux File System /dev/mapper/rootvg-rootlv mounted on / (local/ext4)	45.0
sasva3t.its.unc.edu Linux File System /dev/mapper/rootvg-rootlv mounted on / (local/ext4)	30.3
sasva2t.its.unc.edu Linux File System /dev/mapper/rootvg-rootlv mounted on / (local/ext4)	28.3
sasva0t.its.unc.edu Linux File System /dev/sda1 mounted on /boot (local/ext4)	0.0
sasva1t.its.unc.edu Linux File System /dev/sda1 mounted on /boot (local/ext4)	0.0
sasva2t.its.unc.edu Linux File System /dev/sda1 mounted on /boot (local/ext4)	0.0
sasva3t.its.unc.edu Linux File System /dev/sda1 mounted on /boot (local/ext4)	0.0
sasvaw0tr.its.unc.edu Linux File System /dev/mapper/datavg-sasdata mounted on /sasdata (local/ext4)	0.0
sasvaw0tr.its.unc.edu Linux File System /dev/sda1 mounted on /boot (local/ext4)	0.0
	Updated:4:49 PM

Figure 10. Disk IO Writes in Guest Dashboard

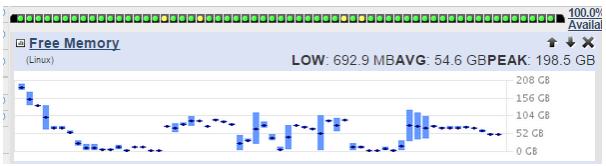


Figure 11. Drilling into detail on Free Memory

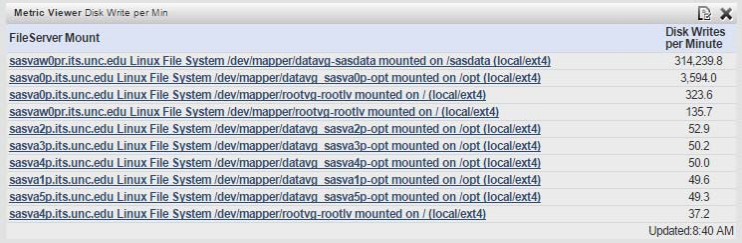


Figure 12. Disk writes per Minute in Guest Dashboard

SAS VISUAL ANALYTICS ADMINISTRATOR

A broader audience lives by these tools that let us look at the current status of data, memory, disk, network, and CPU load. We also mix in use of the gridmon tool. It is important to note that while the two tools provide similar information, the gridmon provides it with less delay than the SAS Visual Analytics Administrator and does not depend on the Flash interface. However gridmon does require an XServer to be running on the local client, and the user must login to the linux shell, have appropriate permissions, and start the tool from the command line. They share much of the same information. Because of the delay difference, and timing sensitive performance testing is better served by using gridmon.

Figure 13 shows nice, short, bursty usage, consistent with short jobs that are very user responsive. This is likely what we see when there is one user, or several users, but no strong overlap between them, essentially time-slicing the grid.

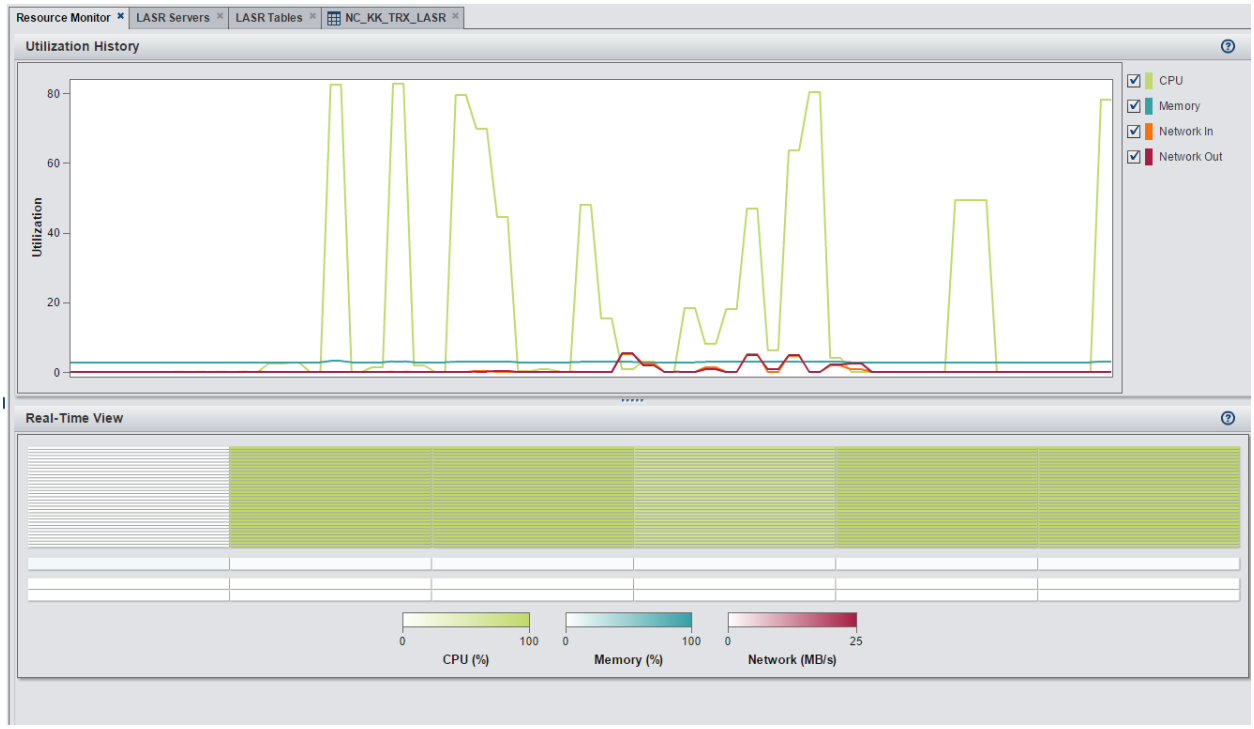


Figure 13. Resource monitor showing nice short spikes of CPU

Figure 14 shows something that you really don't want to see. The orange and red spikes show network load, likely an import from a relational database, at the same time that multiple users are trying to run reports. You can also see that in the leftmost area, only one strong green line shows, indicating strong use of our single WebAppServer. We intend to create more instances to spread the load to multiple CPUs which in the future would be seen as more horizontal green lines in the leftmost server.

Figure 14. Resource Monitor showing excessive CPU from reports while network is importing significant data

Figure 15 shows a more desirable graph, with multiple users running smaller queries across the grid, but when this went on beyond business hours, we realized that something was not right. A few days passed, and it was still happening, and we could not find the cause. Other operations continued normally but this persistent floor that did not appear to cause any great harm only went away when we restarted the LASR. At the time we were loading from the database into the LASR, and clicking the “staging” box on those queries, so the restarts were more impactful as we waited for the data to stream in from the databases. We have since switched our Visual Data Builder queries to load only to HFS, and then it is a matter of a few minutes to completely reload all of the data from HDFS into LASR.

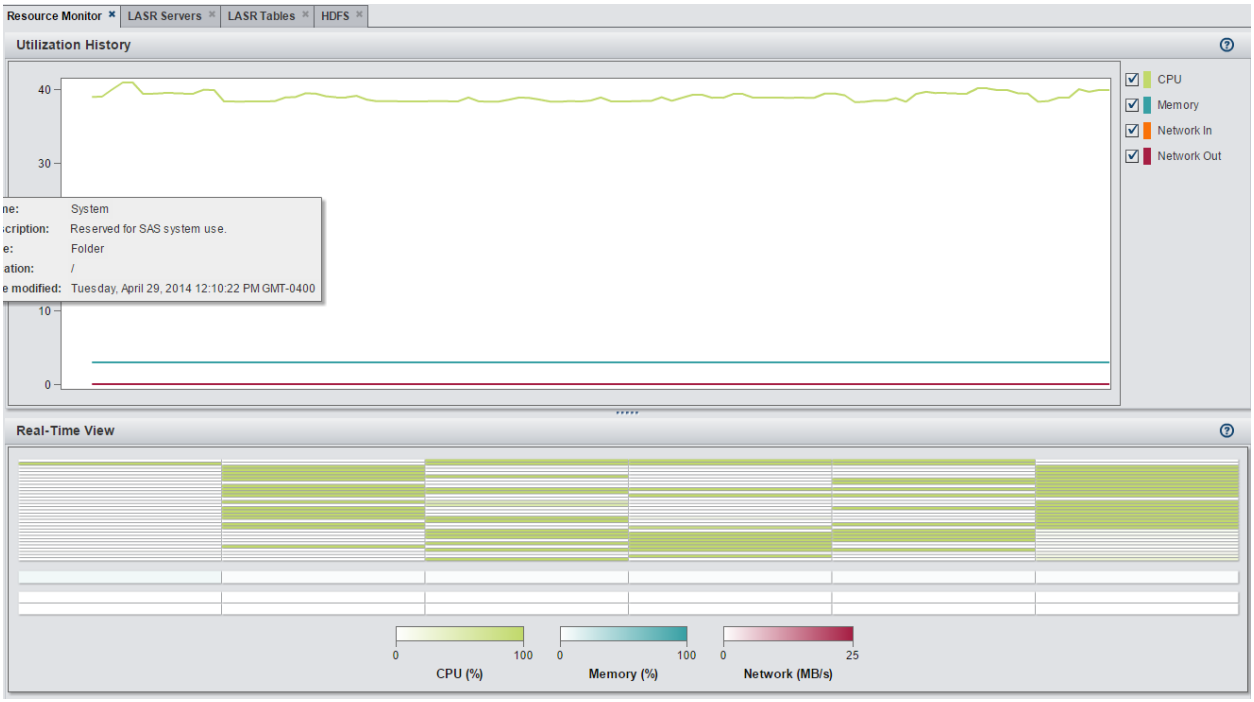


Figure 15. Resource Monitor showing steady use across CPU rather than spikes. Notice that the head node has a single heavy CPU. This state was only resolved by restarting the LASR and the origin is unknown.

Figure 16 shows what might be a more reasonable balance between report running and loading data. It would still be preferable to schedule data loading during off hours to prevent any sort of delays. It appears that when the report running occurs during data loading that the report running takes precedence, which makes sense since the online interactive experience ought to be at a higher priority. The consequence is that data loading may take longer.

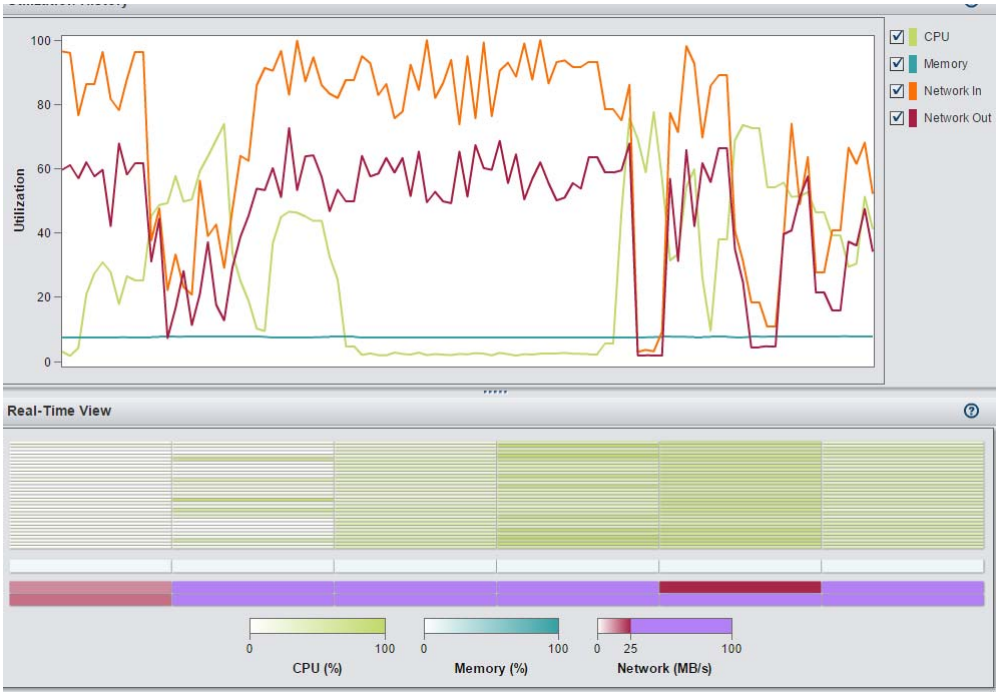


Figure 16. Resource Monitor showing reasonable CPU spikes from reporting while loading data

Figure 17 shows report spikes that are getting wider and wider. Since some of our reports are doing full table scans on multiple elements in the report, plus row level security, we estimate that we are doing trillions of operations just to bring up a report on our 55 million row table. Things start to slow down for the users when the number of simultaneous users increases. At that point we see all green with no breaks between runs and can no longer tell from this view how many processes are running or stacked up, and need to switch to the gridmon utility to see the count of running jobs.

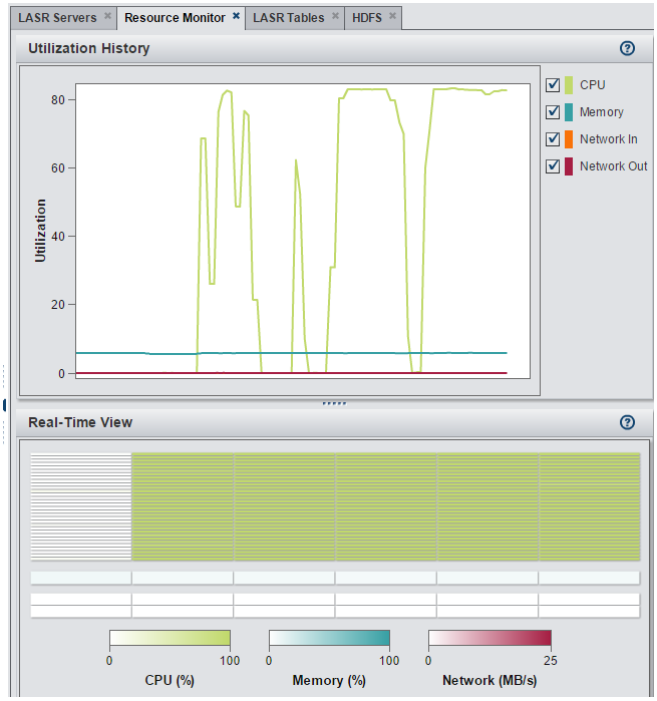


Figure 17. Resource monitor showing spikes that are too wide

There is one more thing of primary interest to look at inside of the SAS Visual Analytics Administrator. It is the HDFS browser, which is very convenient. There is more detail provided elsewhere in the Hadoop utilities, but this one is very convenient and helps us to see what is going on with loading to Hadoop and HDFS. Figure 18 shows an at-rest set of tables, and Figure 19 shows a load in progress. Note that the table name is followed by the IP address of the individual worker nodes. We use this view to monitor the incoming tables to see how quickly they are loading. It appears that the amount of data loading to each node totals up to the overall size of the data set, so the duplication that occurs inside of Hadoop for redundancy is factored out of this display, and if you know the overall size of your dataset you can estimate progress. Frequently we see data trickling in based on constraints in the source system, as can be seen in Figure 20. Only when loading directly from disk can we get the maximum speed available from either the disk or the network, and it is usually our disk that constrains the load speed. We look at the network monitors and the read speed of the disks to come to this conclusion.

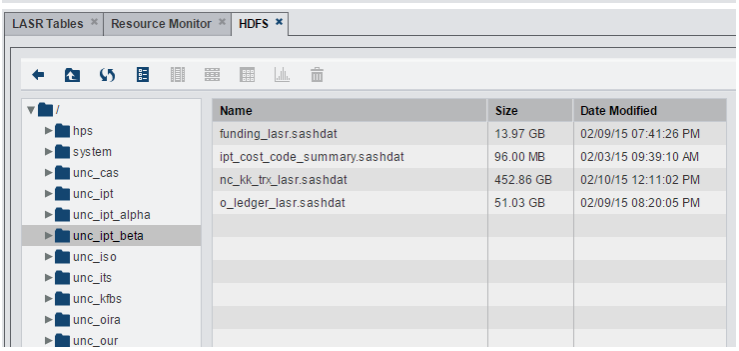


Figure 18. HDFS browser showing tables loaded in Hadoop.

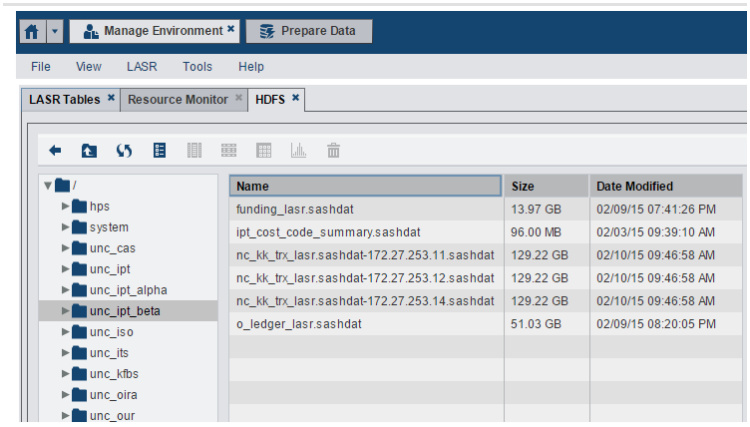


Figure 19. HDFS browser showing data being loaded to 3 Hadoop nodes

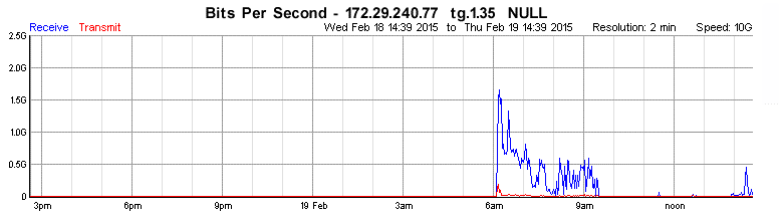


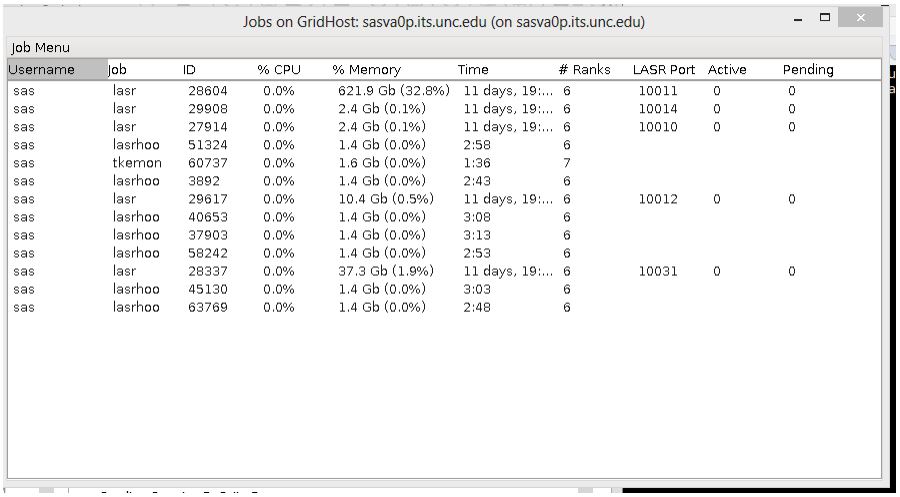
Figure 20. A network utilization graph from our network switch showing the busiest and typical bandwidth.

GRIDMON

The gridmon utility helps us to see what is happening further inside of the system, but is only available to a more sophisticated super user or administrator. Getting setup with an x-terminal session, and finding the directory is required, though we have not seen any user specific restriction once that is done. When we first started up gridmon on a 28 server grid, we thought it was broken. It was, but once we fixed it, it still took a bit of time to initialize all the nodes in the display. Our 6 server environment takes a few seconds to start up.

We have heard from other administrators that use gridmon to watch the environment, in addition to any enterprise server monitoring software. The gridmon shows the status closer to real-time, and has some useful additional details, like the specific % of CPU, Memory, # Ranks, Active, and Pending.

Figure 21 shows the gridmon of one of our environments with a few things of note. The % of memory shows that we have loaded much of our current data into memory based on our understanding of our table sizes. We see a few lasr processes, the first one is the main data warehouse set. We also see a few lasrhoo jobs that correspond to loading of tables. While the lasr processes are long lasting, 11 days at this capture, the lasrhoo jobs are ideally as short as possible. We have seen some stuck ones when the data source job disappears, but the corresponding lasrhoo survives. A little grooming helps to head off any collisions, such as trying to load the same set twice.



Username	Job	ID	% CPU	% Memory	Time	# Ranks	LASR Port	Active	Pending
sas	lasr	28604	0.0%	621.9 Gb (32.8%)	11 days, 19:...	6	10011	0	0
sas	lasr	29908	0.0%	2.4 Gb (0.1%)	11 days, 19:...	6	10014	0	0
sas	lasr	27914	0.0%	2.4 Gb (0.1%)	11 days, 19:...	6	10010	0	0
sas	lasrhoo	51324	0.0%	1.4 Gb (0.0%)	2:58	6			
sas	tkemon	60737	0.0%	1.6 Gb (0.0%)	1:36	7			
sas	lasrhoo	3892	0.0%	1.4 Gb (0.0%)	2:43	6			
sas	lasr	29617	0.0%	10.4 Gb (0.5%)	11 days, 19:...	6	10012	0	0
sas	lasrhoo	40653	0.0%	1.4 Gb (0.0%)	3:08	6			
sas	lasrhoo	37903	0.0%	1.4 Gb (0.0%)	3:13	6			
sas	lasrhoo	58242	0.0%	1.4 Gb (0.0%)	2:53	6			
sas	lasr	28337	0.0%	37.3 Gb (1.9%)	11 days, 19:...	6	10031	0	0
sas	lasrhoo	45130	0.0%	1.4 Gb (0.0%)	3:03	6			
sas	lasrhoo	63769	0.0%	1.4 Gb (0.0%)	2:48	6			

Figure 21. Gridmon Jobs view showing loading jobs in progress (lasrhoo)

Figure 22 shows a CPU use across one of our grids, at what I'd guess is about 50% from how it looks. Hovering over it shows more details about more than CPU, which are to the far right of the green bars in a different color. Figure 23 shows the page you can get to from the Jobs view if you right click on a particular job. Again the green represents CPU and the blue RAM and if you hover over the graph more details are shown. Figure 24 shows the disk use across all nodes under the entry of / (/dev/mapper/rootvg-rootlv) and we have plenty of disk available in this environment. Figure 25 shows the gridmon history view, which is in many was a summary of the current and past activity and the only gridmon that shows non-instantaneous measures. Without these strong gauges to observe behavior in the grid, knowing what to fix where to improve performance is shooting in the dark.

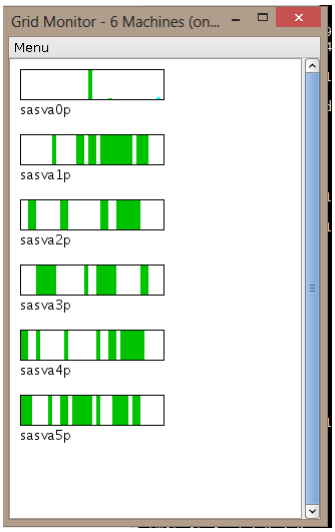


Figure 22. Gridmon Server view across servers with moderate distributed use.

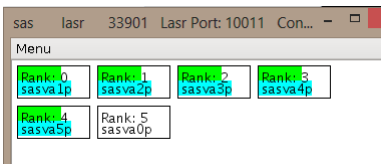


Figure 23. Gridmon view showing CPU and RAM in use across nodes

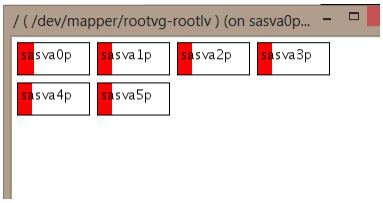


Figure 24. Gridmon view of consumed disk

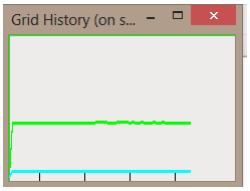


Figure 25. Grimon summary view

LINUX COMMAND LINE TOOLS

These tools and their use are invaluable in identifying and remediating issues that reduce performance. These include the sas disk io test tool, and Linux tools like top, iostat, iotop, df, du, ps, and kill.

We use “ps –ef | grep sasexe” regularly to check for runaway processes. If there is a performance problem, sometimes these processes can be executing long after the user has given up, or tried it several times. Be careful when using kill on these processes – educate yourself by trying it in a safe environment or time, because killing a parent process could result in bringing your LASR process offline and necessitating a reload of all that data.

In conjunction with the above, top –m is another favorite to find anything that looks out of the ordinary. By observing the system after startup and during a few controlled operations, you’ll get a sense as to what is ordinary.

We use iostat to figure out whether we are disk constrained. Running this when performance issues occur, and knowing your baseline by running something like iotest.sh utility provided by SAS can help to determine if you are hitting a bottleneck here.

Sometimes running the iotop utility can shed some light on the processes that are consuming the most IO. It must be run as root, and can help focus the search on which processes to tune.

A neat little utility, iftop shows an ascii graph of utilization for each network connection in addition to the numbers. It is interesting to see the input percent from the data source split n-ways to the Hadoop node along with bandwidth in gigabits/second. We use “iftop -P -i bond0 -N –B” frequently.

We use df and du depending on the situation to see if we are running low on disk anywhere, and where disk files are increasing, such as in the SASWORK directory.

You can also check your data block balance with “/opt/hadoop/hadoop-0.23.1/bin/hdfs balancer -policy blockpool”. If you see that any one node is more utilized than another, you can check and then rebalance the Hadoop nodes. If you’re not using Hadoop, then this won’t help. Other strategies may be to use the partitioning in the SAS Visual Data Builder to encourage data to be contained on nodes to be intentionally unbalanced, but with an overall performance improvement by removing the overhead of the network distribution.

CONCLUSION

You have seen in this paper that there are many ways and places to measure. Without good measurement and identification of performance issues, there is now way to know whether a fix is effective, and how effective it is. It is important to measure when idle, when in use, and then compare those measurements to measurements taken when there are problems.

However, there are a handful of sure-fire ways to improve performance that we have discovered. These include the use of a second RAID channel for SASWORK on the head node or a second SASApp server entirely. Also the way in which data is loaded – to HDFS via SAS Visual Data Builder, with a SQL View selected, and partitioning (if you know your data, queries, and test this approach), followed by a separate load to LASR (we wrote a script rather than doing it by hand). Creating multiple WebAppServer instances if your head node has additional CPU's and memory available, enabling you to restart any instance rapidly if it gets stuck, without having the full 20 minutes a WebAppServer requires impacting users.

What's next for us? We are doing an environment consolidation to bring our Windows based SAS Business Intelligence tools, including Data Management tools, into the SAS Visual Analytics environment, involving hardware reallocations. We will look at the TKGrid resource settings that are part of the SAS High Performance Infrastructure support SAS Visual Analytics version 7.1. We are also looking at a larger grid for our research computing needs having deployed a test of 28 servers where we will alter that configuration to do IP over Infiniband for even higher performance. We also plan to listen and learn from you in order to try things that you have learned. Please feel free to contact us.

ACKNOWLEDGMENTS

There were so many people at UNC, SAS, and our various contractors who have contributed a great deal to the success of this project and the information contained in this paper. We wish to thank all of them for their efforts, because a project of this scope and size would not be possible without the efforts of the hundreds of people involved in this project.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Jonathan Pletzke
University of North Carolina at Chapel Hill
Pletzke@unc.edu

David Franklin
SAS
David.Franklin@sas.com