Paper 898-2017

A CUSTOM METHOD TO AUTO-LOAD SAS® LASR™ TABLES AND LONGITUDINALLY REPORT ON ETL, DQ, AND LASR TIMINGS

Jessica Fraley, University of North Carolina at Chapel Hill

ABSTRACT

Automatic loading, tracking, and visualization of data readiness in SAS® Visual Analytics is easy when you combine SAS® Data Integration Studio with the DATASET and LASR procedures. This paper illustrates the simple method that the University of North Carolina at Chapel Hill (Enterprise Reporting and Departmental Systems) uses to automatically load tables into the SAS® LASR™ Analytic Servers, and then store reportable data about the HDFS tables created, the LASR tables loaded, and the ETL job execution times. This methodology gives the department the ability to longitudinally visualize system loading performance and identify changes in system behavior, as well as providing a means of measuring how well we are serving our customers over time.

INTRODUCTION

The Enterprise Reporting & Departmental Systems (abbreviated as ‘ERDS’) department of the University of North Carolina at Chapel Hill provides services to multiple schools within the University by supplying both data warehouse and reporting capabilities. Each night our department processes and surfaces over 1TB of reportable data to our university customers. Achieving a holistic view of such a large and complex system is a difficult but necessary challenge. This paper is a ‘real-world’ example of how we solved this puzzle.

To meet this challenge, our goals required 1) the ability to surface data about our systems in a simple format, 2) to make the information easy to understand and 3) provide visual interpretation of the data using charts.

Our purpose was not simply to show what was happening at that moment within the system, but to provide a simple visual guide path of how the system (both data and processes) has changed over time, and where the system is going.

In order to accomplish this goal ERDS required a longitudinal view our system’s state and the data being loaded into it. We needed to collect data on a daily basis consisting of 1) how our nightly batch job processes are operating, 2) how much data is being loaded into the LASR server, and 3) measurements of the resources that our nightly jobs are consuming.

With this data collected on a daily basis, viewing how the data changes longitudinally has provided us with a vision that allows us to match system behavioral changes to specific events, and also to see the magnitude of those changes while giving us signposts indicating how our processes may perform in the future.

SAS Visual Analytics platform is the perfect tool to visualize our system and data performance.

THE PROCESS

The process which our department uses involves code that performs several different tasks. These tasks can be broken down into 3 categories.

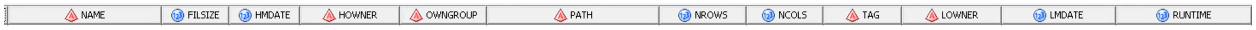
- Nightly Batch Job Processing including batch ETL and Data Query execution.
- Output of data from the Nightly Batch Jobs that provides performance information about the nightly job processes.
- Loading the results of the nightly job output into LASR servers including retrieving the metadata about the tables loaded into the LASR server.

This paper is primarily concerned with how we modified the LASR server auto-load code that served as our base, and what kind of reports we were able to create because of these modifications. As such, we will not spend time on the first 2 bullet points. Fortunately each piece of our process is easy to construct and most of the tasks to be performed are already well documented in other papers. References to the papers which we based individual pieces of our solution on are supplied in the bibliography.

OUTPUT TABLES TO BE CREATED

In order to report on our processes and data, we needed to create 2 tables:

- LASRandHDFSData – a table which will contain metadata (on LASR and HDFS tables) that is extracted as part of our LASR load process. The structure of the table is represented in Figure 1.
- JobStatusReportingData – a table which will contain 3 columns. Each scheduled job appends this table with that job's "starting date time", "ending date time", and return code.



NAME	FILSIZE	HMDATE	HOWNER	OWNGROUP	PATH	NROWS	NCOLS	TAG	LOWNER	LMDATE	RUNTIME

Figure 1. LASRandHDFSData table column list

These tables will be appended nightly and will serve as the data from which we will create our reports.

AUTOLOAD DATA TABLES INTO LASR AND RETRIEVE METADATA FOR REPORTING

The code we used to automatically load our HDFS tables into LASR servers was based on a paper written by Swetha Vuppalanchi of the SAS Institute. Swetha has placed a full code example in the appendix of her paper.

We modified her approach by inserting code which extracts metadata about our target LASR and HDFS libraries into the load process (via a combination of ODS and Proc Datasets). This code outputs the required metadata into the LASRandHDFSData table. The code we inserted into to the LASR table autoload process is listed below:

```
/* Extract Metadata on the tables in our HDFS library*/
ods output members=work.hdfsout;
   proc datasets library=hdfslib nodetails;
   run;
quit;

/* Extract Metadata on the tables in our LASR library*/
ods output members=work.lasrout;
   proc datasets library=lasrlib nodetails ;
   run;
quit;

/* Drop unneeded columns from the HDFS Metadata and rename columns to represent HDFS data*/
data work.hdfsout(drop=num memtype blksize cei perms rename=(mdate=hmdate owner=howner));
   set work.hdfsout ;
run;

/* Drop unneeded columns from the LASR Metadata and rename columns to represent LASR data*/
data work.lasrout (drop= num memtype cei rename=(mdate=lmdate owner=lowner));
   set work.lasrout;
run;
```

```
/* Merge the HDFS and LASR data based on name - SAS Requires HDFS and LASR tables to have the same name */
data work.combine;
   merge work.hdfsout work.lasrout;
   by name;
   runtime= datetime();
run;

/* Append our results to the LASRandHDFSData table */
proc append base= dbLib.LASRandHDFSData data=work.combine force;
run;
```

Both the “LASRandHDFSData” table and the “JobStatusReportingData” table are loaded into the departmental LASR server as the final step of the nightly job runs. The longitudinal growth of these 2 data tables gives us the ability to ask questions about our system’s performance, utilization, and growth which we were not able to ask before. We can now look at how individual jobs are performing, how fast our data is growing and the stability of our systems as a whole.

THE REPORTS

The ERDS department uses the gathered data to create 2 different types of reporting:

1. Operational – reporting which gives a clear view of the current state of our data and our system
2. Projective – providing data about how our system changes over time and gives us information about how our data and system may perform in the future.

REPORTS CREATED WITH DATA GATHERED FROM METADATA

Completion times of nightly jobs report (Operational):

Our first report examines the completion times of our nightly jobs. This includes everything from the start of the first job to the completion of all LASR tables being loaded and ready to be reported on (see Figure 2).

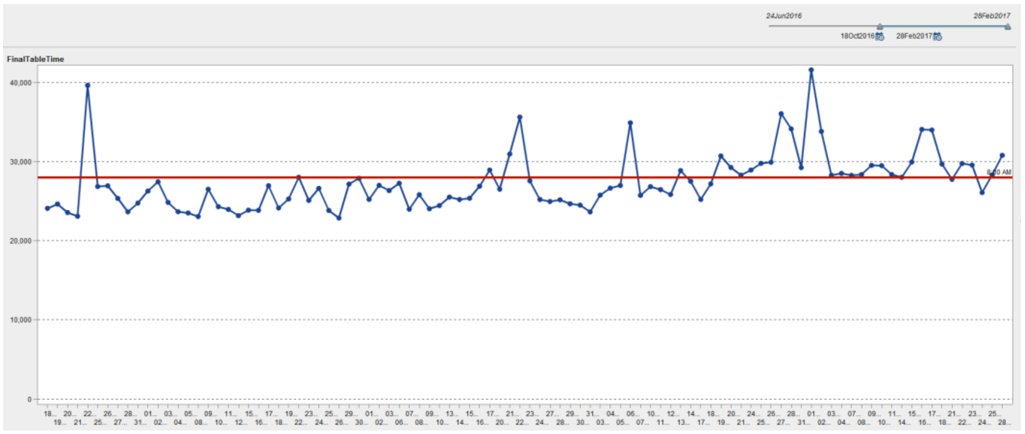


Figure 2. Longitudinal report displaying the completion times of nightly jobs

Each observation in Figure 2 represents the nightly job completion time for a specific date. This report is our most basic report. This longitudinal view gives us information about how the nightly batch job completion times change across days, weeks, and months. As you can see our completion times have recently been climbing past the 8AM mark (the red line). Previously our nightly runs regularly completed by 8AM. This may, or may not be something that we want to investigate. However, having this report gives us a general view into how our nightly jobs are executing and allows us to ask questions about how our nightly jobs are performing.

Row Growth per LASR Loaded Table report (Projective):

Data growth rate is another basic but very important report for ERDS.   This report (See Figure 3) graphically displays our data growth via row counts across time.

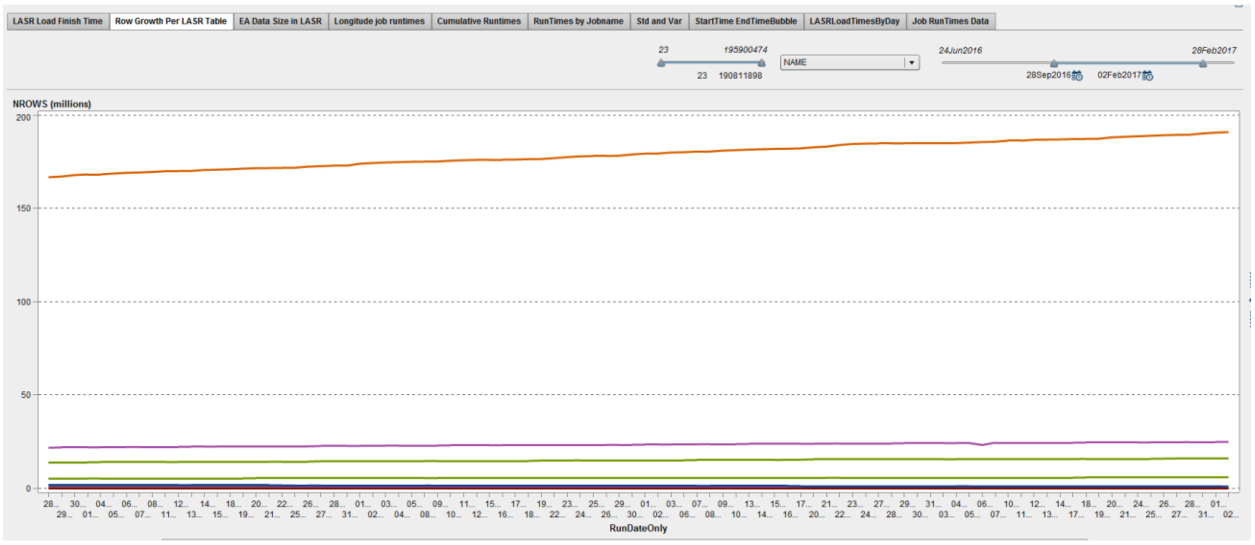


Figure 3. Report showing data growth rate as measured by row counts in each table.

Figure 3 reveals how many observations are being added to each data source across time. In this report, none of the growth rates appear alarming from a ‘growth slope’ perspective. However if we select the table represented by the top line, the actual numbers give us a different picture of the information (See Figure 4).

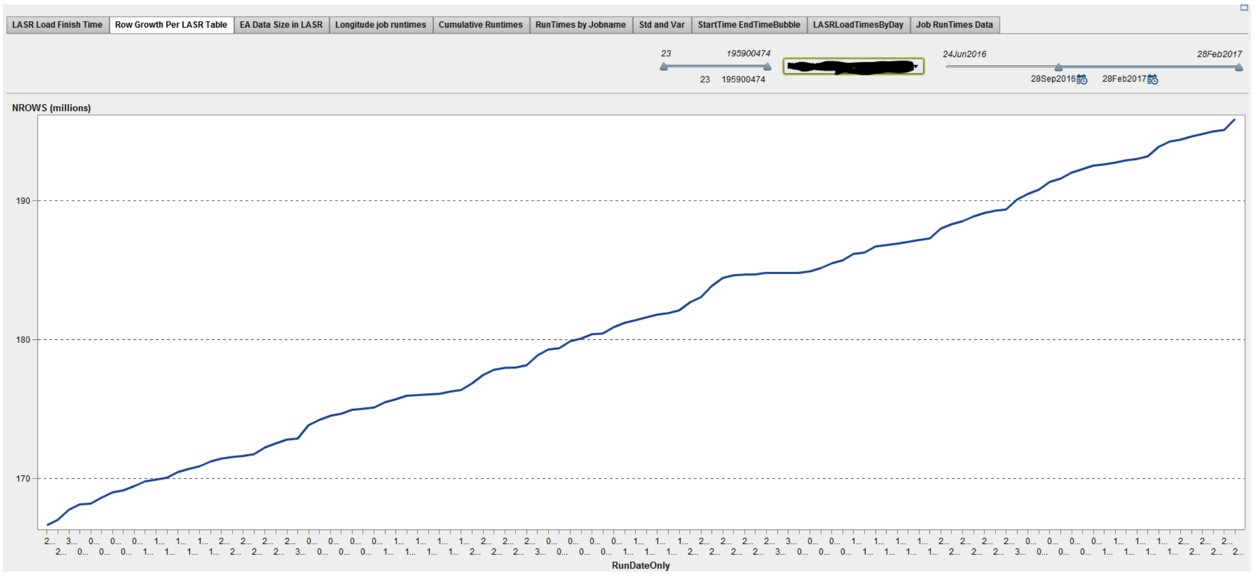


Figure 4. Growth rate as measured by row count for a single table

Figure 4 represents the data growth in rows of a single table from 28 Sept 2016 to 28 Feb 2017. Over a 5 month period the number of rows for that table grew from 165 million to 195 million. That is a growth rate of 30 million rows which is a 43% annual growth rate. Now we have hard data about how fast data is being added to this table, we can ask ourselves “what will happen to report and nightly job performance if that growth rate continues”?

Cumulative Total Growth of all ERDS data loaded into LASR server report (Projective):

Absolute data file size growth provides a crystal clear picture of how the demands on our system are changing (See Figure 5).

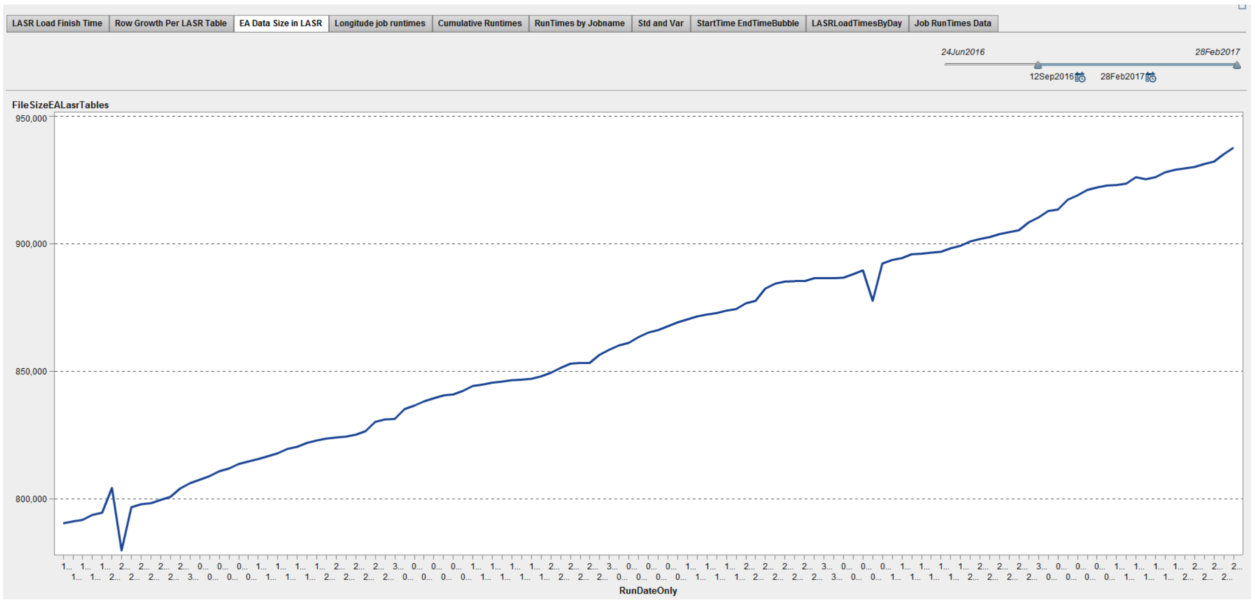


Figure 5. Growth rate of all ERDS data being loaded into our LASR servers.

Figure 5 visually displays the growth rate of the ERDS departmental data that is being loaded into our LASR servers and surfaced to customers for reporting on a daily basis. This knowledge gives us the ability to project the size of our data at a future date.

On 12 Sept 2016 we had a total data volume loaded into the SAS LASR Servers of 790 GB. On 28 Feb 2016 we had a total data volume of 937 GB. That is an 18.6% growth rate in under 6 months. Averaged over a 1 year period, our departmental data growth rate is approximately 40%. Having this knowledge allows us to pro-actively forecast system needs, and anticipate possible future performance issues.

REPORTS ON NIGHTLY JOB EXECUTION TIMES

Cumulative runtime of all jobs that are run at night report (Operational):

Completion times are one measure of how nightly jobs are performing. Another measure that gives us a greater capability to understand how our nightly jobs are behaving is to measure the aggregate of each individual job’s execution time. This measure shows us resource time that the nightly jobs are consuming as a whole. Figure 6 shows how we display this information:

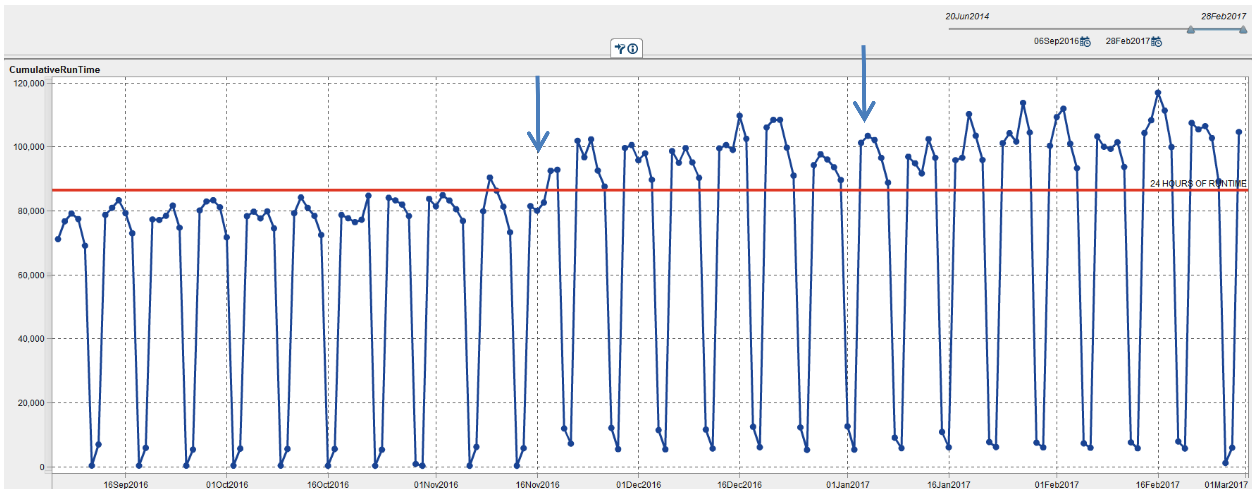


Figure 6. Aggregated runtimes of all nightly jobs

Each data point is a cumulative aggregation of the total execution time for each job in our nightly batch run. These data points represent not only run time, but indicate resources consumed during the nightly run.

As you can see from Figure 6, we have had 2 separate increases in resource consumption between 06 Sept 2016 and 28 Feb 2017. One of these increases in resource consumption (beginning around 15 Nov 2016) can be correlated to additional jobs that were added to the nightly runs at that time. However the cumulative runtimes have been increasing since then without clear explanation. Having this information displayed for us in an easily read visual format allows us to identify issues and potential fixes before they become problems.

Longitudinal runtime of individual jobs report (Operational):

Each nightly batch ETL job and deployed Data Query appends that job's "start date time", "end date time", and "return code" into the JobStatusReportingData table. Having these values allows us to calculate the actual runtime for each job on a daily basis.

Figure 6 visualized the fact that around 15 Nov 2016 we started consuming more computational resources every night. Given the increase in resources being consumed we needed to be able to view the longitudinal runtimes of each individual job in order to identify which jobs were consuming the additional resources.

A line graph provides the ability to plot the actual runtime of each job (Y axis) on a daily basis (X axis), giving us a clear view of how individual jobs are effecting nightly runtimes and providing clues on resource consumption changes (See Figure 7).

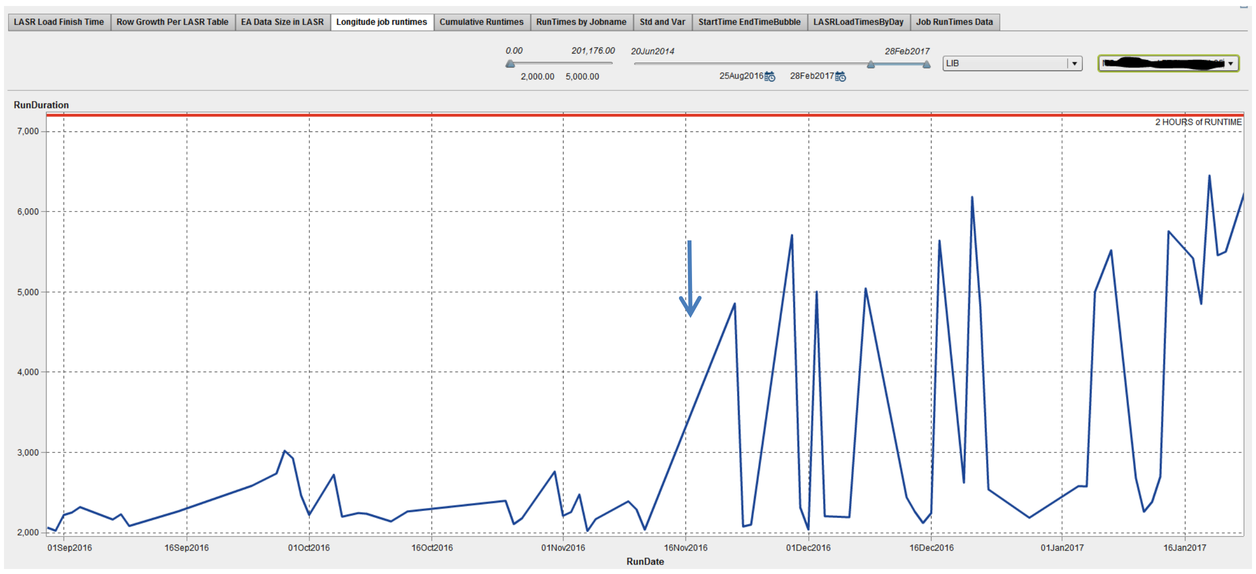


Figure 7. Longitudinal Report on runtimes of individual jobs within our nightly batch process

As you will see from Figure 7, on approximately 16 Nov 2016 one particular job began behaving erratically and had greatly increased runtimes. Given the increase in resource consumption indicated in Figure 6, having a longitudinal view of each job's runtime makes it easy to identify job behavioral changes that correlate with system events.

Nightly Job interactions Report (Operational):

In ERDS, we wanted a view of how the jobs interacted as they executed on a nightly basis. In order to gain this insight we created a bubblechart in SAS Visual Analytics (See Figure 8). Our bubble chart plots each job's start time in seconds past midnight, end time in seconds past midnight, and job run duration. This allows us to see not only the order in which the jobs executed, but when they started, when they ended, and identify any jobs that were running at the same time. The advantage of this particular report becomes evident when comparing these relationships across multiple timeframes.

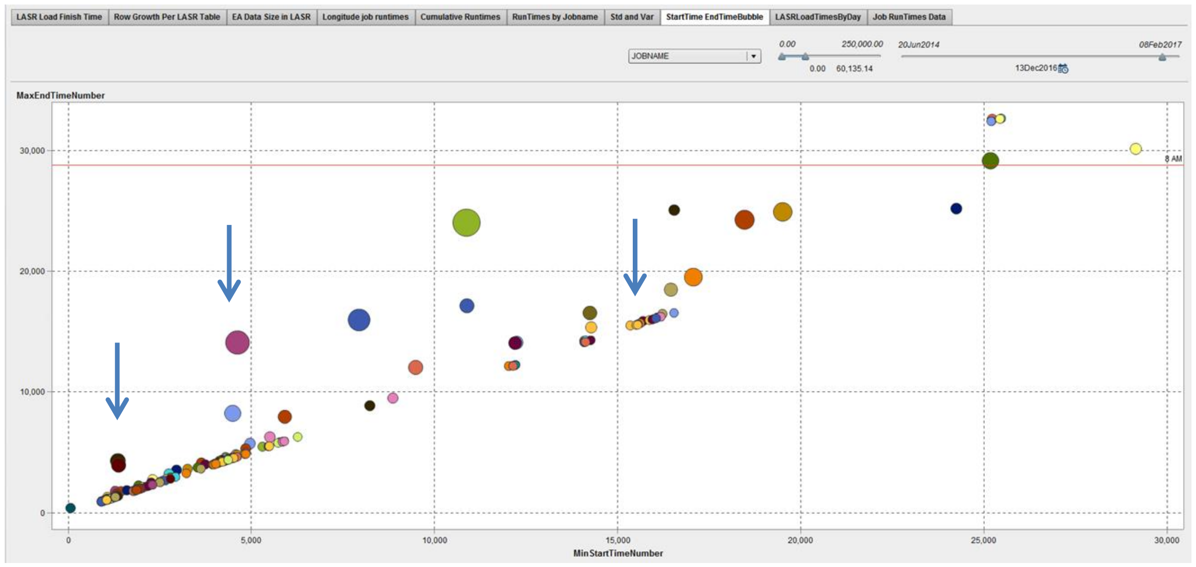


Figure 8. Nightly job execution order display with start and stop times – 13 Dec 2016.

As you will note in Figure 8, we have arrows pointing at jobs in 3 locations on the chart which is dated 13 Dec 2016. When comparing those jobs to the positions of the same jobs on 28 Jan 2017 (See Figure 9), you will notice that each of the arrowed jobs has changed location when compared to the start and stop time of other jobs.

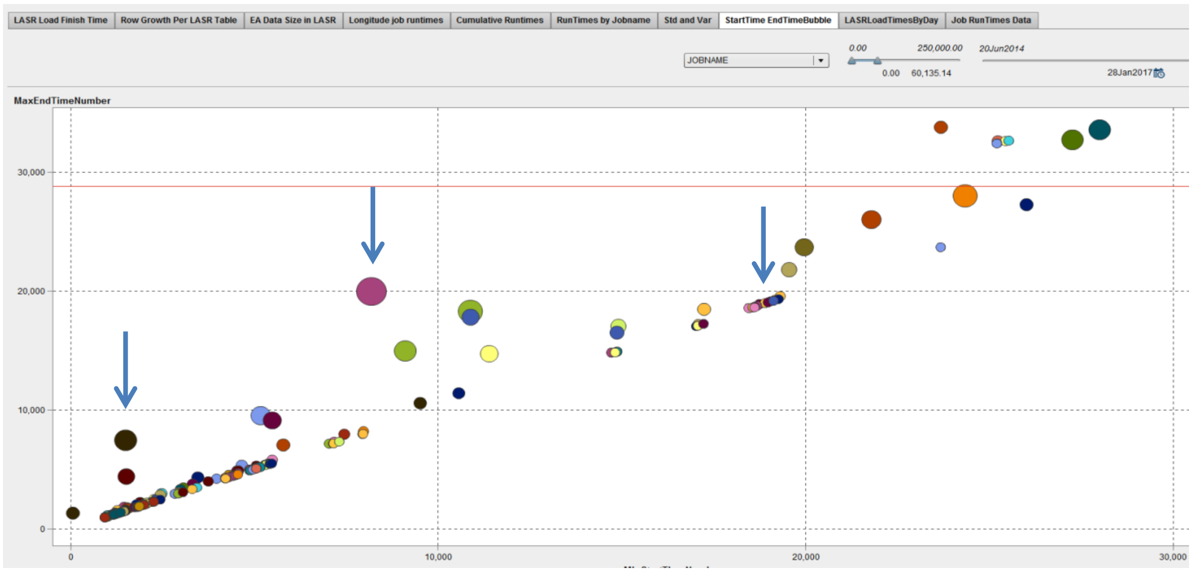


Figure 9. Nightly job execution order display with start and stop times – 28 Jan 2017.

Longitudinally comparing this report across dates provides us with an easy visual means of identifying which jobs are executing simultaneously and which batch jobs may be affecting the performance of other jobs. This is very helpful information when attempting to locate the root cause of a change in performance, as opposed to where the behavior manifests.

Standard Deviations of Job Runtime as a Moving Average Report (Predictive):

ERDS also wanted the ability to visualize the reliability of our system's performance. We accomplished this by creating a report based on the average standard deviation of each job's runtime, and then plotting each average standard deviation on a bar chart.

An increased average standard deviation indicates when a job's runtimes vary greatly and increase unpredictability in our nightly process (See Figure 10). With this data we can then visually see changes in the variability of our nightly job execution times by comparing the area under the curve across different timeframes.

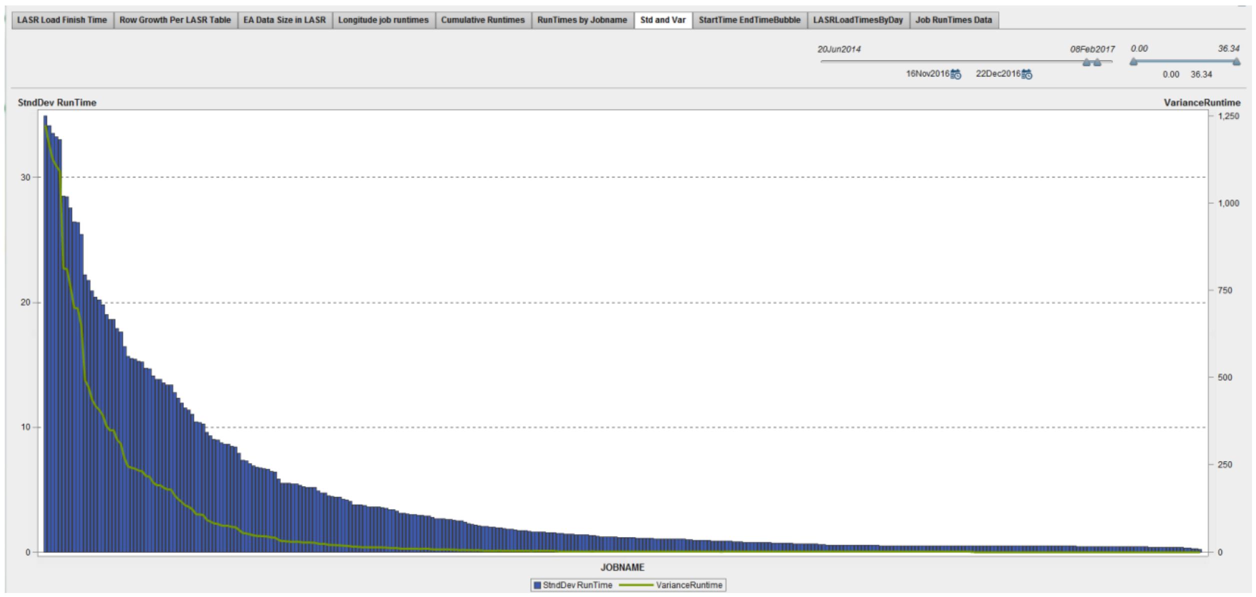


Figure 20. Average Standard Deviation for each nightly batch job from 16 Nov 2016 to 22 Dec 2016

If we compare the report from Figure 10 to the same report for dates 25 Dec 2016 through 30 Jan 2017 (See Figure 11), we can see a definite increase in the area under the curve.

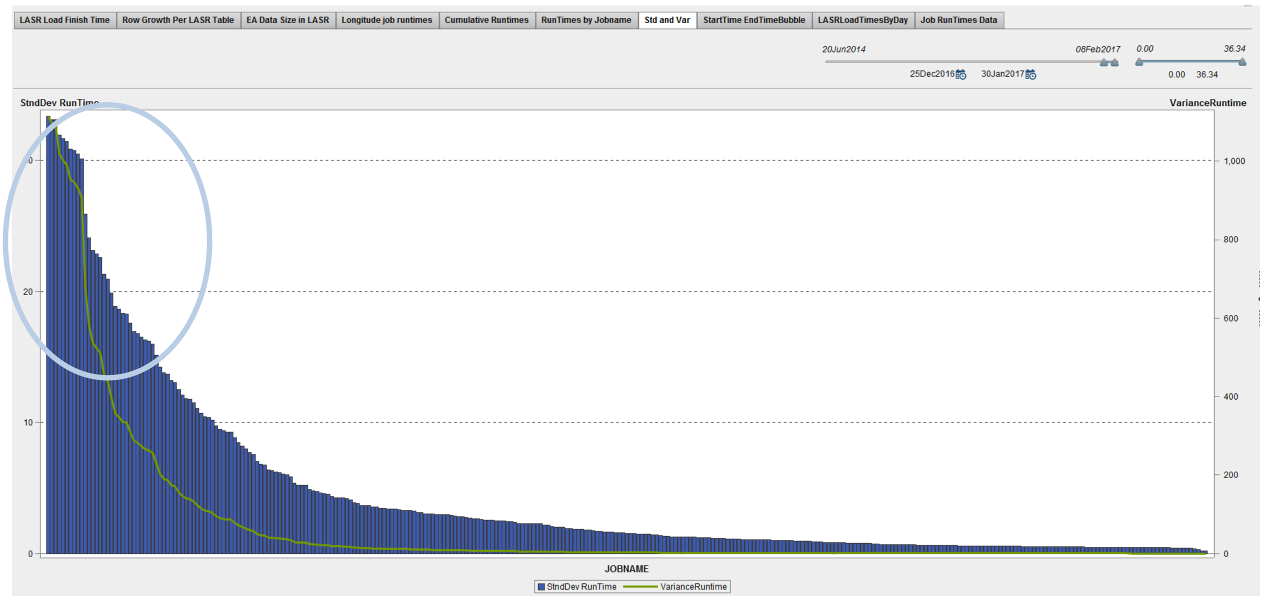


Figure 31. Average Standard Deviation for each nightly batch job from 25 Dec 2016 to 30 Jan 2017

The number of jobs with high average standard deviations increases when comparing Figure 10 and Figure 11. Even the jobs which do not have excessive average standard deviation scores are showing an increase in average run time standard deviations. This report provides us with a measure of stability within our nightly batch job runs and highlights increased volatility before it becomes a problem.

CONCLUSION

SAS Visual Analytics is a powerful visualization tool. We often forget that our systems produce data as well as our customers. By adapting SAS Visual Analytics for the purpose of administering our data and system resource consumption, ERDS has been able to create an easy to understand, visually oriented window view into our systems. This window not only displays how our systems are behaving currently, but also compares current performance to past performance and gives us a view into how the systems will behave in the future.

REFERENCES

Vuppalanchi, Swetha. 2016. “Automating Load for Multiple Tables in SAS® Visual Analytics” Proceedings of the SAS Global Forum 2016 Conference. Cary, NC. SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings16/3660-2016.pdf

Aanderud, Tricia. “Oh Snap! Upload Data to the LASR Server just like That!” 1 Mar 2017. Available at http://bi-notes.com/2015/10/oh-snap-upload-data-to-the-lasr-server-just-like-that/

CONTACT INFORMATION

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

Jessica Fraley
University of North Carolina at Chapel Hill
ITS Manning, Suite 2800 CB #2808
211 Manning Drive
Chapel Hill, NC 27599
Jessica_Fraley@unc.edu