Franklin Interactive Node Responsiveness

Richard Gerber
User Services Group
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
Berkeley, CA
June 9, 2008

Introduction

Anecdotal reports of slow interactive response on Franklin's login nodes have been documented via comments on the 2007 NERSC User Survey. Users report that sluggish command-line response at times makes it difficult to work. The cause, or causes, of the poor response time is unknown.

In an attempt to quantify the user experience and maintain a baseline for future comparisons we are monitoring the time needed to complete three small tasks commonly performed by users on login nodes: 1) compile and link a small code, 2) perform a "long listing" of a small directory, and 3) create files. A shell script running via cron every 15 minutes is used to measure the duration of each of these tasks and the results are saved for analysis. This data may also provide insight into the underlying cause(s) of the problem, but that is not addressed here.

Data presented and discussed below confirms that the execution time of these almost trivial commands varies significantly, often reaching values that are many times longer than a user would expect to experience on modern computers, whether PCs or supercomputers.

Method

A cron job that executes on Franklin login node nid4100 runs a script that uses the "time" command to measure the wall clock time it takes to execute three small tasks.

1) Compile and link a simple Fortran code using the "ftn" compiler
2) Create 500 small files in a previously empty directory
3) Execute "ls –l" in a directory containing 6 files

Each task is timed separately. The cron job executes every 15 minutes. Measurements were first taken on April 25, 2008 with all file IO in the /home file system. An additional script was started in cron beginning on May 28, 2006 using the /scratch file system. For comparison, identical measurements were taken beginning April 29 on Bassi using /home. The specific choice of commands was not considered to be of great importance — the goal was not to benchmark performance for these commands, but rather to create a baseline for future comparison and quantify the frequency and magnitude of variations, particularly extreme ones.

Standard Output from the commands was directed to /dev/null and Standard Error – which contained the output of the "time" command – was returned to a master perl script for parsing and recording. Timing results were saved in a text file in the /usr/common file system. Writing to /dev/null gives an absolute minimum measurement from a user perspective because no time is needed to print to the user's console. The temporal resolution of the "time" command is 0.01 seconds. The commands used here were chosen as representative of simple tasks commonly performed by NERSC users at the command line on login nodes. If a non-trivial time is needed to return from these simple commands it is significant. More complex commands could reasonably be expected to take longer.

Results

All measurements recorded through June 6, 2008 are shown in Figs. 1-3. Each point represents a single measurement. The scale on the Y axis is logarithmic.

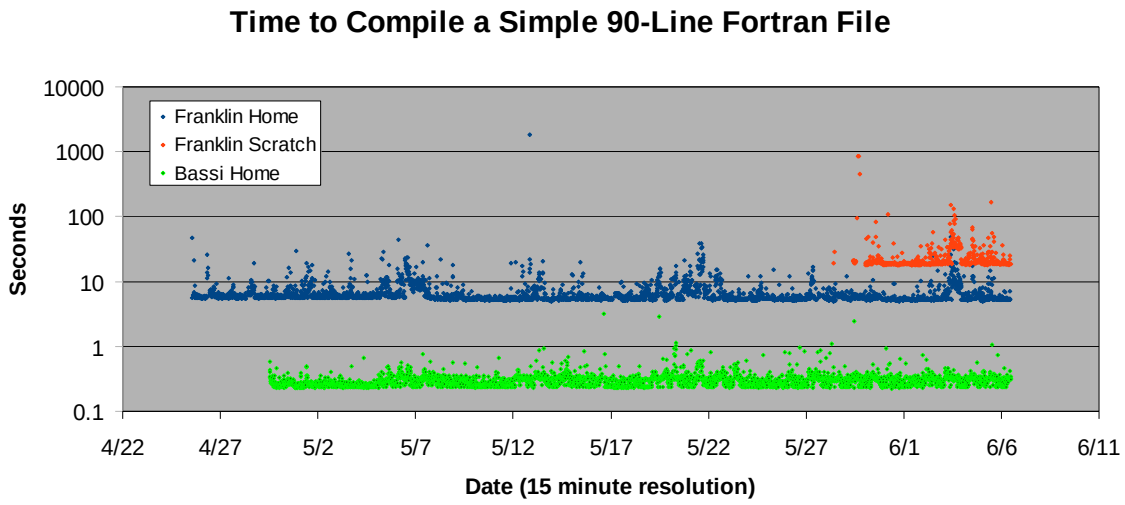

Figure 1. The time needed to create an executable file from a simple 90-line Fortran source code file. Each dot represents a single measurement; data was collected every 15 minutes. The source and resultant executable files resided in the file system indicated. The y axis scale is logarithmic.

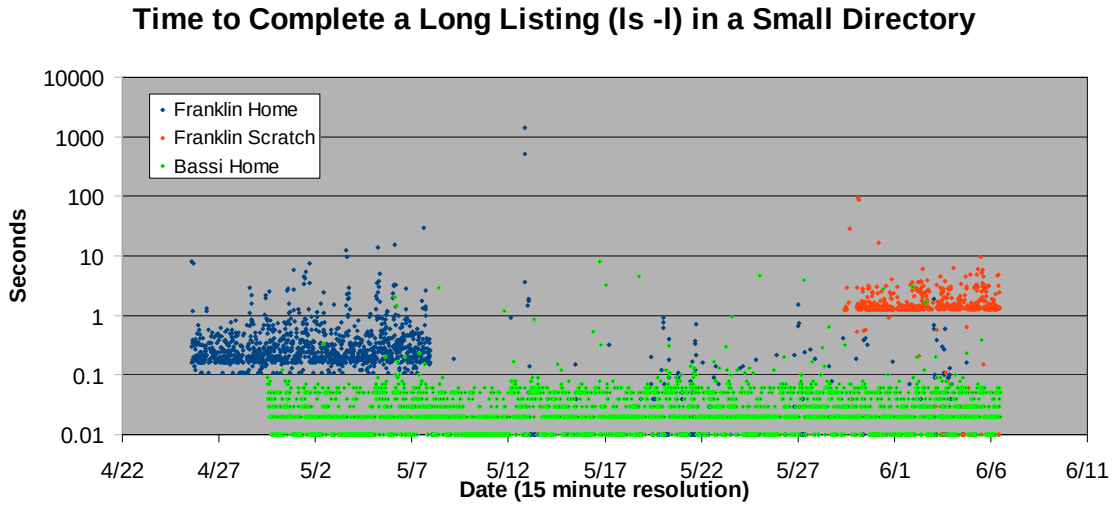

Figure 2. The time needed to execute "ls –l" in a directory containing 6 files. Each dot represents a measurement, taken 15 minutes after the previous one. The files resided in the file system indicated. The y axis scale is logarithmic. Execution times less than 0.01 seconds were recorded as 0.00 seconds and are not shown. The abrupt change for Franklin Home on May 9 is the result of a change in the LDAP authentication configuration.

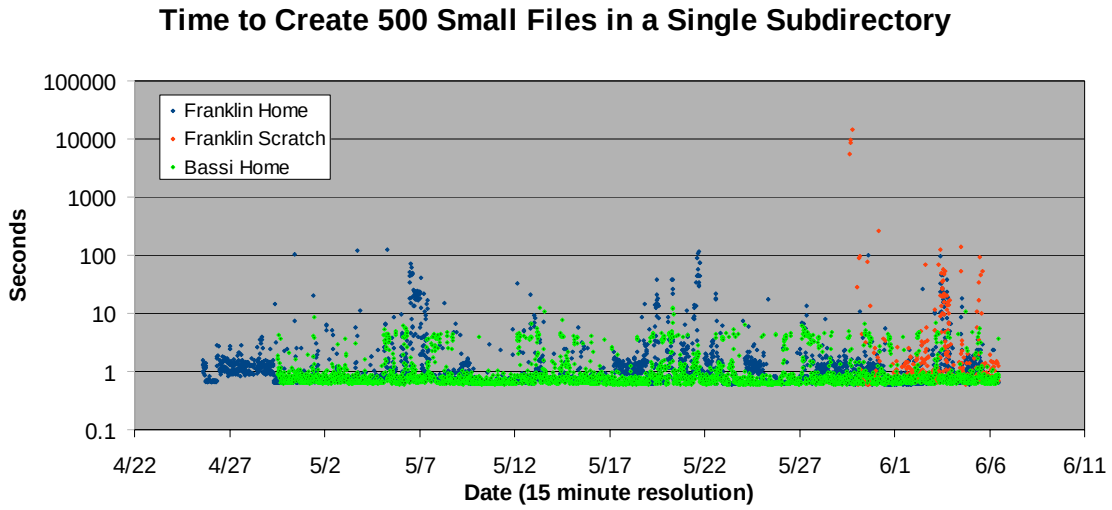

Figure 3. The time needed to create 500 small files in a single subdirectory. Each dot represents a measurement. The files resided in the file system indicated. The y axis scale is logarithmic.

Discussion

The data shows wide variation in execution times for these short commands; a scale logarithmic in time is needed to accommodate many orders of magnitude. This spread is significant because it spans the range from acceptable to unacceptable performance. What is acceptable? This question is somewhat subjective, but the commands tested here can be reasonably expected to finish almost as quickly as a user can type the commands at the keyboard on today's computers. Timings of up to a second might not impact the responsive "feel" of the system, but anything beyond that is noticeable and likely unacceptable to the average NERSC user. Anything that repeatedly takes minutes is likely to generate a significant outcry.

Clearly there are measurements that fall into these unacceptable ranges. The feedback from NERSC users has been expressed to NERSC staff and documented in responses to the 2007 NERSC user survey. Although the survey is still underway as of June 9, 2008, a sampling of comments from the 300 respondents reveals their unhappiness:

- "At times it is difficult to work on Franklin. It can be terribly slow to do anything. I press a key and 30 secs later I get a response from the terminal. There needs to be some way of managing or upgrading … to make Franklin usable."
- "The Franklin login nodes are painfully slow at times. This is a real show stopper as a simple ls can take 3-4 minutes."
- "Franklin's login nodes are not very responsive / slow most of the time."
- "Things I would like to see improved: 1) Franklin frequently has slow command line responsiveness …"
- "Sometimes response is very slow. For example, the ls command takes a few seconds. This is frustrating."
- "Making the response time better on Franklin would be nice."
- "At times it is very difficult to get work done since the response time is so slow."

Compile Times

It took at least 5 seconds to compile and link in /home on Franklin. About 5% of the measurements were more than 10 seconds and greater than 15 seconds 2% of the time. While this may not seem unreasonable, the same build takes 1 second or less 99.9% of the time on Bassi. There is a legitimate argument that, because Franklin executables are statically linked, it should take longer to create an executable than it does on Bassi, which uses run time libraries. But from a user's perspective the build takes at best about 5 seconds longer than on Bassi and in one case was measured at more than 30 minutes – 1,800 seconds. The Franklin /scratch file system is slower still; the minimum time was 18 seconds.

The following graph shows the percentage of compiles that completed in less than the time given on the X axis. For example, 65% of the builds took 20 seconds or less using Franklin /scratch. The difference compared to Bassi is notable because behavior on that system may play into setting users' expectations. The purpose of this report, however, is not to benchmark the performance difference between systems, but rather to identify and measure extreme variations. Franklin's variability is much greater than Bassi's over time scales noticeable to users working at the command line. The data also suggests that users should build codes in the /home file system.

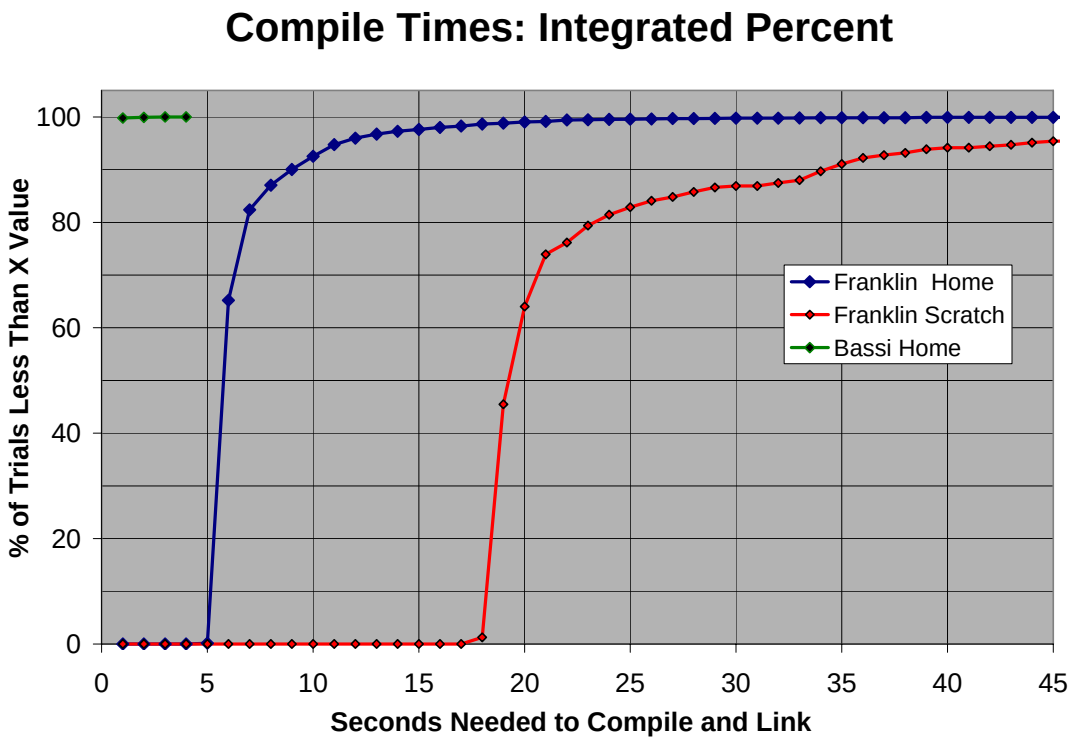

Figure 4. The percentage of measurements that executed in a time equal to or less than shown on the X axis. All trials on Bassi completed in less than 5 seconds.

Listing a Directory's Contents

Users expect to be able to list the contents of a directory in real time, even when the directory contains thousands of files. For example, on the author's 3.4 GHz Intel Pentium 4 desktop machine, a long listing (ls –l) of 3,500 files in /usr/bin takes about 0.06 seconds if redirected to /dev/null and about 3.5 seconds to print to the console. A similar listing of a directory with 6 files finishes in less than 0.01 seconds, even when printing to the console. NERSC users will likely expect similar response from Franklin login nodes.

Figure 2 shows that the trivial "ls –l" in /home on Franklin took always more than 0.1 seconds, often many seconds, and up to 30 seconds. This is a noticeable delay and makes the system "feel sluggish." On May 8 an LDAP caching daemon was enabled, resulting in measurements that are now usually 0.00 seconds (below the resolution of the "time" utility). Compared to Bassi, Franklin's response in the /home file system is now generally similar.

Performance in /scratch remains much worse for listing a directory. The measurements are typically a second to a few seconds, although there are occasional results below 0.01 seconds.

The following graph compares the percentage of measurements that completed in less than the time shown on the X axis. For example, 75% of the listings took 1.5 seconds or less using the Franklin /scratch file system. Three curves are shown for Franklin /home: 1) with the LDAP cache enabled after May 8, 2) with the LDAP cache disabled before May 8, and 3) all measurements over the recording period. With the cache enabled, /home response on Franklin is excellent. Note than in Franklin /scratch, about 20% of the trials completed in approximately 0.01 second or less. Of the remaining 80% virtually all took at least 1.3 seconds. This is suggests the possibility that the results are being influenced by caching somewhere in the process, with a 1.3-second penalty for missing the cache.

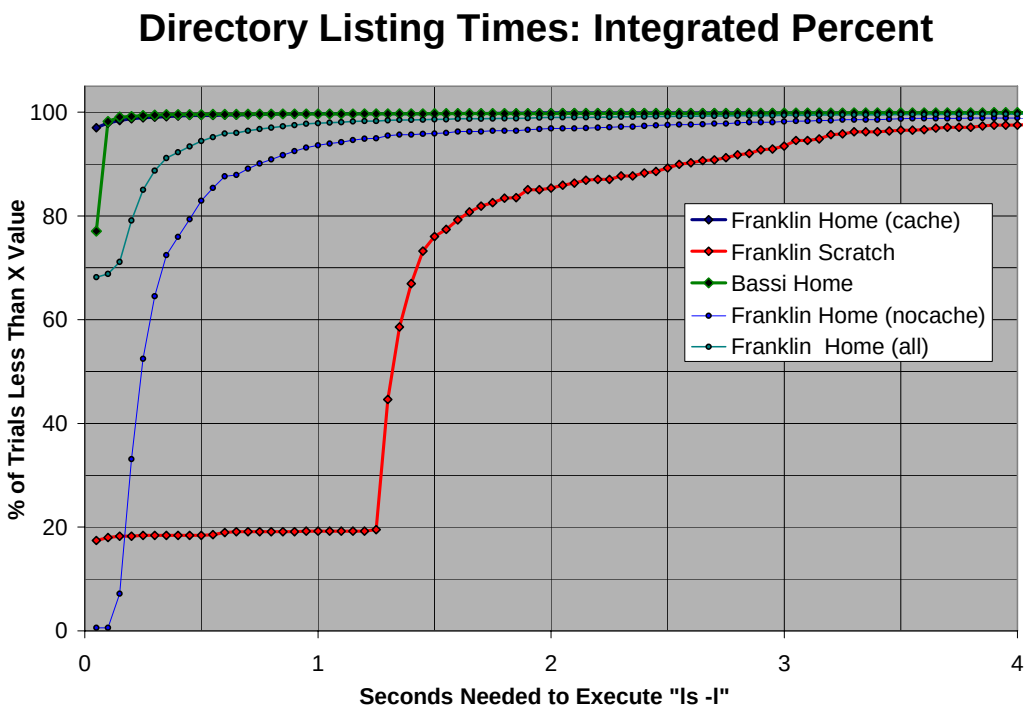

Figure 5. The percentage of measurements that executed in a time equal to or less than shown on the X axis.

File Creation Rate

Users frequently need to create many files in a single directory, perhaps as part of a post-processing run analysis or while building their code from source code files. Slow file creation makes a system feel slow. Figure 3 shows that all three systems tested can create 500 small files in a single directory at about the same rate. However, the variation among trials is huge – many orders of magnitude – with a factor of 10 not uncommon.

The following graph shows the percentage of measurements that completed in less than the time shown on the X axis. For example, 90% of the time it took 3 seconds or less using the Franklin /scratch file system. Most results are comparable to Bassi, but Franklin data has a long tail. The maximum on Bassi /home was about 12.5 seconds. On Franklin /home the greatest value was 123 seconds and 2% of trials measured greater than 18 seconds. On Franklin /scratch the maximum was 4 hours (possibly a pathologic data point?) and 2% of the results were over 52 seconds.

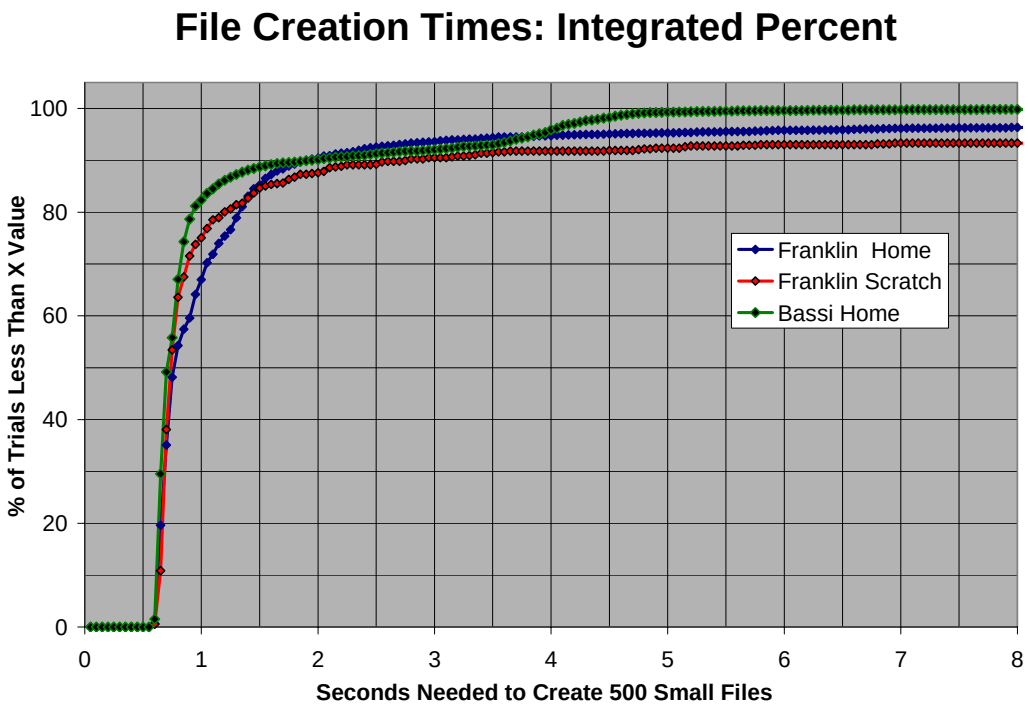

Figure 6. The percentage of measurements that executed in a time equal to or less than shown on the X axis.

Correlations and Periodicities

The variations obvious in Figs. 1-3 are not randomly distributed. There are mountains and spikes interspersed with flat, quiet regions. Although analysis does not reveal any significant periodic signals in the data, it is reasonable to ask if there are correlations with other events. Many known events that might influence the data presented here have been recorded for Franklin (file system errors, node failures, etc.). Searching for correlations between those events and this data is beyond the scope of this report. However, it is possible to test for correlations between the individual data series gathered for this report and the node's 5-minute load average, which was recorded for data taken after April 30, 2008. While the load average does not directly reflect CPU utilization, but it does give loose a measure of the system load, albeit an average over the 5 minutes before and during which data was recorded.

The correlation coefficient was calculated for the data series measured in the Franklin /home file system. Data series with coefficient values greater than 0.5 are considered to be strongly correlated, those greater than 0.3 have a medium correlation, and above 0.1 are said to have a small correlation. The only data pair with a coefficient greater than 0.1 was the 5-minute load average and the time to create files. This lack of correlation suggests that there is not a single root cause that affects compiling, directory listings, and file creation performance, unless it operates on the time scale of seconds or less, which is the time between execution of commands in the scripts.

Data Series Pair for Franklin /home	Correlation Coefficient
Compile & Create	0.102
Compile & ls -l	0.012
Create & ls -l	0.002
Compile & Load Average	0.035
Create & Load Average	0.300
ls -l & Load Average	-0.005

Finally, are the results from Franklin /scratch and /home correlated? The comparable measurements were separated by approximately 5 minutes, which then is the limit to the temporal resolution of any correlations that might be revealed. Given that, yes, there are correlations, as shown in the table below. When compiles are slow in /home they are also slow in /scratch. The file creation rate is less, but still positively correlated. Directory listing times in the two file systems appear to be independent.

Franklin Data Series	
Home & Scratch Compile	0.603
Home & Scratch File Create	0.279
Home & Scratch ls -l	-0.029

Summary

Data systematically collected on a Franklin login node quantify the time needed to perform three simple tasks frequently performed interactively by NERSC users: compiling and linking a small program, generating a small listing of files in a directory, and creating small files. The quantitative measurements show that trivial commands sometimes take much longer than is tolerable in a modern computing environment. While the definition of “tolerable” is subjective, some NERSC users have voiced their unhappiness in the NERSC user survey.

The data provides a baseline for comparison with performance under future configurations. While the results do not identify underlying causes, they can reflect the impact of relevant changes, e.g. the enabling of a local LDAP cache. A correlation analysis between the individual data series do not show strong correlations, suggesting that there is no simple, single underlying cause for the variations observed for all three tests.