File System Monitoring as a Window Into User I/O Requirements

Andrew Uselton, Katie Antypas, Daniela Ushizima
CRD/NERSC, Lawrence Berkeley National Laboratory
{acuselton,kantypas,dushizima@lbl.gov}
Jeffrey Sukharev {jsukharev@ucdavis.edu}
University of California, Davis

May 26, 2010

Abstract

As part of the 2008-2009 upgrade of the Franklin Cray XT at the National Energy Research Scientific Computing center, its I/O resources were more than doubled. The extra bandwidth reduced contention for I/O resources and those resources were split into two file systems, /scratch *and* /scratch2*, each with more bandwidth than the original* /scratch*. In an effort to improve responsiveness, NERSC consulting encouraged "big I/O" jobs to use* /scratch2*. File system monitoring data indicate that this division of the NERSC workload has been largely successful, and we present several useful characterizations of the two I/O workloads that resulted. This feedback for both the users and the center enhances NERSC's ability to manage and provision Franklin's I/O subsytem as well as to plan for future I/O requirements. The workload characterization presented here also lays the groundwork for the development of a rigorous methodology for measuring the impact of I/O on system utilization, which we outline in Future Work.*

1 Introduction

The Franklin Cray XT at the National Energy Research Scientific Computing center (NERSC) regularly has over 300 users logged in and submitting jobs, with 100-200 applications running at a given time. The scratch file systems on Franklin are a shared resource with no explicit "quality of service" or "fair share" mechanism, so there is inevitably contention for I/O bandwidth. Time lost to I/O contention represents CPU hours lost to the user, as well as a lower overall scientific productivity from the system. Contention also leads to variability in job run time. Some I/O variation is expected and tolerated by users, however, if an application's wall clock time varies significantly, it can undermine performance analysis and workflow management. Analysis of the I/O workload on Franklin using the techniques presented here enables NERSC to manage the I/O resources to benefit the users individually and to improve the overall productivity of the system.

NERSC has roughly 3000 users and 400 different projects spanning the program offices of the Department of Energy's Office of Science, including: astrophysics, material science, chemistry, fusion energy, nuclear physics, climate research, and more. It is a diverse workload that represents NERSC's mission in serving the Department of Energy's research community. Much effort is spent studying the application workload in order to procure systems that meet the requirements of the scientific applications [1] . In this paper we extend this effort to include user I/O requirements. As the premier supercomputing platform at NERSC, it is critical that we understand the Franklin Cray XT system's I/O capabilities and how they meet the needs of the NERSC workload.

Each year NERSC Principal Investigators (PIs) submit an allocation request for time on NERSC machines. From this group of over 400 projects, 50 PIs with large I/O intensive projects answered a detailed survey about the project's I/O requirements, current practices and future needs [6]. Some of the highlights from the survey include: I/O is dominated by append-only writes; I/O read and write sizes vary widely (from a few KB to hundreds of MB); most applications use a file-per-processor approach - where each process writes to its own separate file - rather than using parallel I/O APIs (such as MPI-IO).

A few application's I/O requirements and I/O patterns have been studied closely (FLASH [4], MADbench [2], Vorpal [10] [6]), however, user I/O requirements for the broader application workload are still not well understood.

Even an application whose I/O patterns are known can have I/O usage vary depending on the size of the simulations and the frequency of output. Furthermore, the I/O pattern resulting from hundreds of applications running concurrently can be very different from the I/O pattern of any single application.

It has been our experience that some large applications perform relatively little I/O while some small, low concurrency jobs put a heavy I/O load on the system. Additionally, NERSC has a number of users who run 3rd party applications not authored by NERSC users and thus whose I/O requirements are less well known.

This paper takes a detailed look at the I/O taking place at NERSC, and in particular, on the Franklin Cray XT4 supercomputer.

1.1 The Franklin Cray XT4

The Franklin Cray XT4 machine was delivered to NERSC in the spring of 2007 and was accepted in October of that year. Franklin arrived as a dual-core (Opteron) machine with 9660 compute nodes connected via the Cray SeaStar–2 interconnect, which is organized as a 3D torus. The scratch space on Franklin is presented to the compute nodes via the Lustre [3] parallel file system. At the time of acceptance the single scratch file system, mounted as */scratch*, was measured to have a peak write bandwidth of about $11GB/s$ and a peak read bandwidth of about $8GB/s$. In the Autumn of 2008 Franklin was upgraded to have quad-core nodes with twice the memory, doubling the compute capacity of the system. Prior to the upgrade NERSC was concerned that this change, doubling the compute capacity without changing the I/O capacity, would lead to an imbalance in the system. This concern turned out to be well founded. Even before the upgrade, some users had voiced concerns about I/O performance, and after the upgrade reports of slow I/O increased. A comparison of the balance between I/O and compute capacity on other supercomputing platforms lead Cray and NERSC to augment Franklin's I/O capacity. The bandwidth capability was tripled and and storage was increased by 20%. At that time, NERSC also divided the I/O resources into two equal file systems */scratch* and */scratch2*. Each file system now has a peak performance of roughly $17GB/sec$ for both reads and writes.

Computing time on NERSC systems is not divided equally amongst the 400 projects. Rather, there are roughly a dozen projects that together use about a third of the computational time and a couple dozen projects that use the next third. The remaining 350 projects share the last third of the computational time at NERSC. In order to balance I/O across the two scratch file systems on Franklin, NERSC contacted "big I/O" users - those who either used more than the default disk space or who had previously reported slow I/O on the system - and asked them to run applications out of */scratch2* rather than */scratch*. NERSC hypothesized that the few users whose applications were sensitive to I/O performance would take the time and effort to move to a new filesystem. The I/O intensive jobs and users who moved would benefit from lower contention on */scratch2*, on the one hand, and */scratch* would be relieved of the "big I/O" to the benefit of the rest of the users. Almost all the users who were contacted were members of the top 50 projects by allocation at NERSC. The users were not required to move to */scratch2*, and other users were not prevented from moving to */scratch2*. Nevertheless, this informal policy had effectively separated the user workload into two parts, with "big I/O" users running out of one file system and smaller I/O users running in the other.

In order to understand the I/O requirements of the Franklin job mix we monitor the jobs closely. In order to do so we rely on the job log maintained by the batch scheduling system and on the Lustre Monitoring Tool (LMT) [9], a server-side I/O data acquisition system providing bytes read and bytes written by the servers every five seconds.

In the remainder of the paper we will describe the characteristics of the separate workloads of the two scratch file systems. Our ultimate goal is to develop a rigorous methodology for measuring the how well the I/O subsystem performs under a given workload. With such a measure we could evaluate if, as we believe, this segregation has lead to an improvement in the utilization of the Franklin system as a whole. The workload characterizations presented here are a first step in that direction. Finally, we will introduce (as future work) a new methodology for analyzing the connection between the LMT data and the job log that holds the promise of leading to the desired measure of system performance.

2 Basic Workload Statistics

The first, simplest way to characterize workload is to answer the question, "How much data was moved?" The data in Table 1 comes from the two sources of monitoring on Franklin: the server-side I/O statistics gathered by LMT [9] and the job log maintained by NERSC's batch system. Table 1 gives the month-by-month totals for bytes read and bytes written on the two file systems along with the total number of jobs run each month. Note that 1.6 times as much data is read as is written to the two Franklin scratch file systems. The total number of jobs being processed by Franklin is relatively steady from month to month, and that is true both for the totality of jobs and for those run at high concurrency. However, the aggregate I/O totals vary widely both from month to month and from */scratch* to */scratch2*. The fact that reads dominate writes is contrary to our expectation based on the results of the

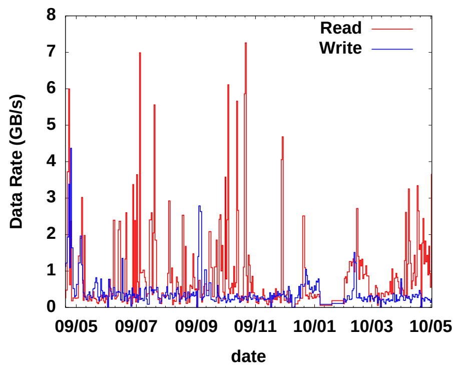
	/scratch		*/scratch2*		number of jobs		
	read	write	read	write	all jobs	> 15 nodes	> 255 nodes
2009-04	2.518	2.280	0.043	0.171	49,161	24,499	4168
2009-05	0.988	1.318	0.257	0.310	47,453	21,940	2919
2009-06	2.278	1.180	0.355	0.476	37,307	18,609	3807
2009-07	2.989	0.714	0.397	0.390	36,969	16,254	3093
2009-08	1.545	0.774	0.255	0.274	47,236	18,913	3757
2009-09	1.690	0.980	0.267	0.298	45,215	19,347	2940
2009-10	4.086	0.673	0.252	0.977	57,329	23,312	4321
2009-11	4.775	1.223	0.509	1.349	52,896	22,422	3630
2009-12	0.728	0.888	0.181	0.466	40,699	19,673	3717
2010-01	0.200	0.370	0.063	0.326	47,484	23,944	3791
2010-02	2.481	0.907	0.272	0.680	57,136	25,610	3898
2010-03	1.090	0.532	0.386	0.743	51,275	25,218	4481
2010-04	3.262	2.374	0.234	2.201	54,007	25,830	3308
total	28.63	14.21	3.24	6.46	624,167	285,571	47,830
average	2.20	1.09	0.25	0.50	48,013	21,967	3679
std. dev.	1.291	0.587	0.143	0.348	6,450	2998	479

Table 1. Amount of data read and written in PB (10^{15} bytes) on the two scratch file systems. *\/scratch* moves four times as much I/O as */scratch2*. On */scratch* there is twice as much read activity as write. On */scratch2* there is twice as much write activity a read. Overall, there is 1.6 times as much read as write activity. I/O varies widely, though the number of jobs being run is fairly steady.

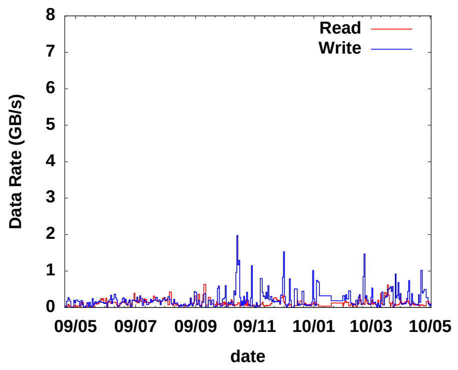
NERSC survey of I/O intensive users. Clearly, the pattern for the workload as a whole is not governed by those who reported that their I/O was dominated by append-only writes. Our survey of I/O intensive users did not tell the whole story.

The job log does not record which scratch file system, if any, a job used, nor does the server side data from LMT identify from which job a particular I/O operation originated.[*] We can see from Table 1 that the workloads do appear to be consistently different between the two file systems in their volume and the balance between reads and writes. The workload on */scratch2* appears to more closely resemble the write intensive I/O profile of the projects identified originally as heavy users of I/O, but most of the I/O and even most of the write I/O still takes place on */scratch*. Looking at how much data is stored on the two file systems we find */scratch* typically is maintained at 50% utilization while */scratch2*'s steady state utiliza-

[*]Lustre does have an optional capability to record client-side statistics on the server, but it is a memory intensive activity that becomes prohibitively expensive at large scale. LMT does not employ that capability.

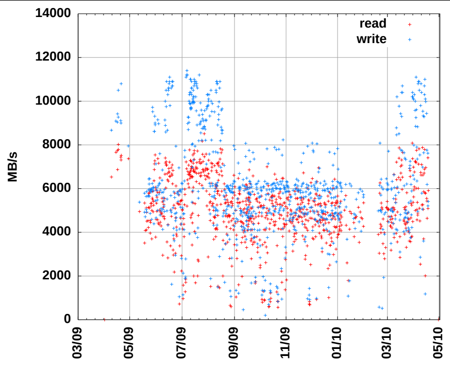


(a) */scratch*

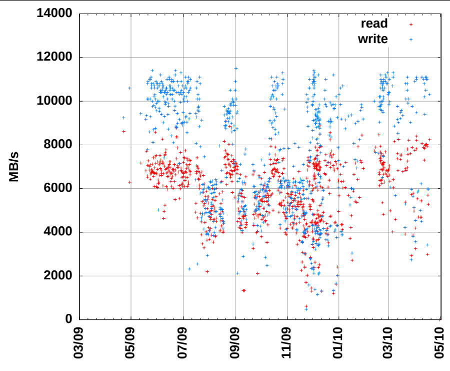


(b) */scratch2*

Figure 1. 24 hour average rates calculated daily



(a) */scratch*



(b) */scratch2*

Figure 2. A regularly scheduled production-time I/O write and read experiment indicates variability.

	/scratch		*/scratch2*	
	read	write	read	write
optimum	80	55	80	55
average	141	118	111	91
ratio	1.76	2.15	1.39	1.65

Table 2. Contention is worse on */scratch* than on */scratch2* and is worse for writes than for reads.

tion is around 30%. The number of users running in each file system however, has a much wider spread. Only about 85 users run out of */scratch2* whereas over 450 users run from */scratch*. On average each */scratch* user reads 4.8TB and writes 2.4 TB of data per month. On */scratch2* the pattern is reversed. The average user running out of */scratch2* reads 2.9TB and writes 5.9TB of data per month. It is interesting to note that only about 30 users have disk usage of more than the default 750GB quota on either scratch file system and only a handful of users have disk usage of more than 3TB. This means users are writing and reading many temporary files.

Figure 1 shows the daily average value of read and write rates on the two file systems as reported by LMT. Here again, it is apparent that reads dominate on */scratch* and writes dominate on */scratch2*, that most I/O is taking place on */scratch*, and that the I/O pattern varies markedly from day to day,.

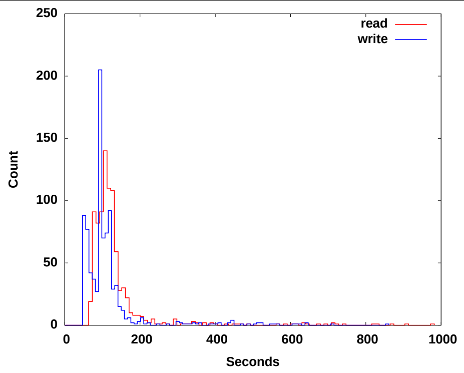
The next section reports on a standard I/O performance test that is run at regular intervals on each file system. That ”test probe” gives some insight into the consequences of I/O contention given the workloads on the two file systems.

3 Probing I/O Contention and Variability

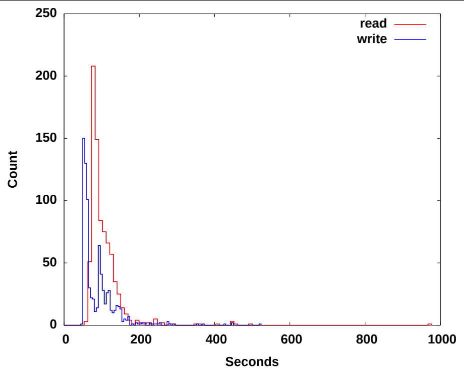
To better understand performance variation on Franklin we initiated a regularly scheduled set of I/O tests, which are launched by a cron job and run three times a day. The performance of a parallel file system is commonly measured with a benchmark that probes the file system for its performance given a variety of constraints. One such benchmark is IOR [5, 7]. The output from IOR gives, among other details, the time spent writing and the amount written, the time spent reading and the amount read, and the rates calculated from these values. Our chosen test ran IOR with the following settings: 64 tasks on 16 nodes, transferring $4MB$ with each I/O transaction, using POSIX I/O calls to a separate file per task, and writing $100GB$ from each task and then reading that $100GB$ back in. These automated tests were performed between April 2009 and May 2010.

Figure 2 shows a scatter plot of the performance variation for the 800 tests each for read I/O and write I/O on both the */scratch* and */scratch2* file systems. Table 2 summarizes those observations.

Figure 3 recapitulates the scatter plots of Figure 2 as plots of the frequency histograms for the durations of the tests. The *x-axis* gives the duration of the test with the scale divided into 100 bins. The *y-axis* gives the number of tests with a duration in the range for each bin. The histograms have the general shape of a Poisson distribution, or perhaps a long tailed power law distribution of some sort, and have a couple of modes as well. The precise nature of the distributions is beyond the scope of the present discussion, as we only consider how the average



(a) */scratch*



(b) */scratch2*

Figure 3. This figure shows the histograms of the duration of the IOR variability test. Optimum values for read tests are a little slower than for writes. On *\/scratch2* the dominant mode for reads and writes is at or near the optimum value, though there is a significant mode at slightly slower values. On *\/scratch* the dominant mode is significantly slower than the optimum for read and writes. Contention is present on both file systems but tends to dominate only on *\/scratch*.

test probe compares to the "ideal case". The "long tail" corresponds to the poorly performing tests in the scatter plot. The minimum value for each distribution represents a good estimate of the ideal case, where the probe had no interference at all. The ideal case values conform well to tests run on a dedicated system. The ratio of the average test duration to the best-case duration gives some indication of the relative interference observed on the file system. It should be noted that the relative interference is not necessarily a universal property of I/O on the file system. On the contrary, very short jobs like the test probes will tend to sample the transient I/O rates on the system, and thus would be expected to vary as the aggregate I/O on the file system itself varies. Jobs with much longer sustained I/O will see something more like the average behavior of the file system, so will have a very different distribution.

We note in passing that there are occasional periods of slower performance for reads and writes on both */scratch* and */scratch2*. Those periods of degraded performance correlate with times of especially severe imbalance of space allocation across the servers and were present at various times on both file systems. For our purposes this is just another opportunity for contention and lost compute time, though it does represent a separate mode of operation for the file systems and is reflected in the histograms in Figure 3. That, in turn, makes the interpretation of Table 2 a little tricky, since the histograms are capturing two separate effects: the two modes of operation, and the penalty for experiencing contention. The next section explores contention and variability in closer detail.

4 The LMT View of I/O Contention and Variability

LMT provides server-side data rates for the file systems every five seconds. Plotting the instantaneous level of file system activity over time gives a useful view of file system activity. In Figure 4 the *x-axis* is wall-clock time for the period of interest, and the *y-axis* is the aggregate data rate: red for reads, and blue for writes. If one further superimposes the start time, stop time, and reported rates for an IOR test then one can compare what the servers experienced with what the IOR test reports. The comparison in Figure 4(a) is a useful sanity check on the two monitoring methods, where an IOR is run during a "dedicated testing time" with no other activity on the system.

In Figure 4(b) two IORs are launched simultaneously during the dedicated testing time. They must contend with each other for file system bandwidth, and in this case seem to have split that bandwidth relatively evenly. Thus each test takes twice as long to complete, and if the tests had been run one after the other the pair would have completed in about the same total time. However, in Figure 4(b) the CPUs for both jobs are occupied for the entire time. If the CPUs are forced to be otherwise idle during the I/O then the contention directly leads to lost productivity for the system.

Figure 5(a) shows one of the test probe IORs being interfered with during regular production time. Most, though not all, of the incidents of poor test performance that we investigated from those in Figure 2 had a similar character. Another example of interference can be seen in

[†]When the sample density is large enough the graph is easier to read using points rather than lines, but the underlying data is of the same sort in either case.

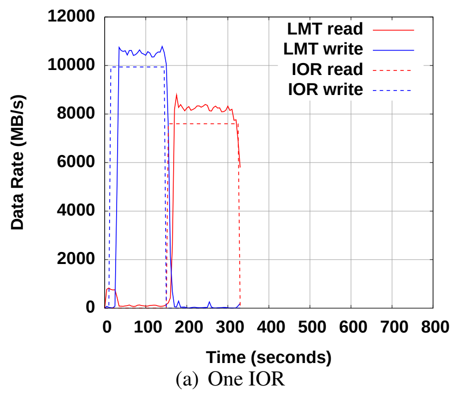


Figure 4. a) One IOR test running on a quiescent system proceeds in an orderly fashion. b) Two IORs compete for the available bandwidth, and in this case each gets about half, they both take twice as long. If two IORs had been run one after the other they would have together completed in about the same time as those in b). However, in b) twice as many CPUs were kept occupied for the entire time.

Figure 5(b). In this case each sample is plotted as a point.† This was not during one of the regularly scheduled tests, but was observed in the normal course of monitoring I/O behavior. An interval of high read activity (at about 11:00) appears to suppress write activity, presumably from other jobs.

One way to get a sense of the I/O workload is to look at a 24 hour period of LMT data as in Figure 6. In Figure 6(a) the */scratch* file system is heavily occupied with read I/O for the entire day. In Figure 6(b) the */scratch2* file system is relatively lightly loaded but for a few brief, transitory I/O operations. Reviewing daily LMT data reports will show days that span the range from light to heavy activity, from read dominated to write dominated, and with well-defined I/O operations as in Figure 6(b) or with a continual, variable, and undifferentiated sequence of observations. Looking at LMT data in this way does not scale in the sense that increasing amounts of data does not lead to a clearer picture of the workload. In the next sections we look at statistical methods of analyzing the LMT data, in order to establish a more rigorous characterization of I/O workload.

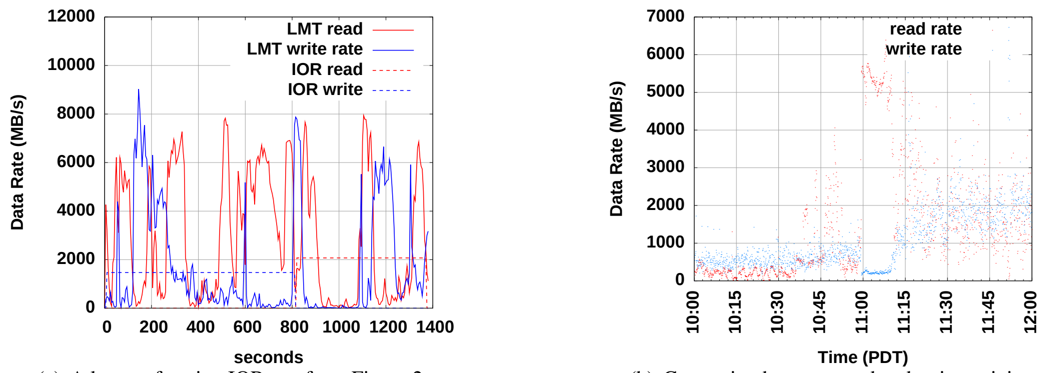
5 The I/O Power Spectrum

The I/O rate observed by each server at each sampling interval gives an instantaneous and transient view of the file system as a whole. By gathering all such observations into a histogram of observed I/O rate versus count we can get an aggregate view of I/O that emphasizes the modes of behavior rather than the time order of events.

The power spectrum for LMT data is built by first constructing the histogram of the LMT I/O observations. The LMT observations are binned based on the number of bytes moved in each observation. Figure 7 uses 100 bins ranging from 0 bytes up $2.5GB$. The power spectrum modifies the underlying histogram by multiplying the count by the bytes moved for that bin. Thus the power spectrum gives the total contribution to the I/O of the I/O in each bin. Figure 7(a) gives the power spectrum for */scratch* for the 13 months of operation since the I/O upgrade in March of 2009, Figure 7(b) gives the power spectrum for */scratch2*.

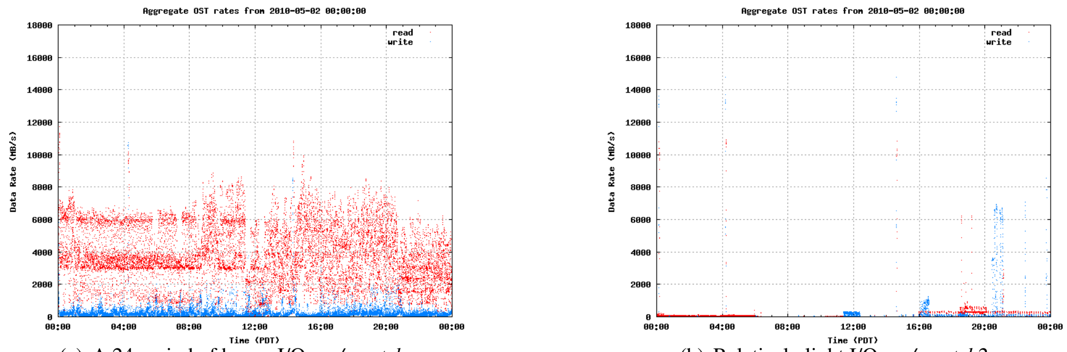
The power spectrum for the I/O observations gives a useful characterization of the aggregate workload on the file system. A month of such data reveals, in some cases, a broadly diffuse spectrum where the workload consists of a wide variety of different behaviors, none dominant. In other cases the distribution is a low flat line punctuated by a few very dominant modes. In that case the workload itself is made of a few dominant jobs with a very specific and repeatable I/O profile.

The centroid of the power spectrum is the bin at which half the I/O is in larger observations and half in smaller observations. For */scratch* read I/O and for */scratch2* read and write I/O the centroid is very close to the center of the range of bins at about $900MB$. The centroid for */scratch* write I/O is much smaller at about $400MB$. This tells us that write I/O on */scratch* is much more dominated by smaller individual I/O observations, and we may surmise that this is because it is dominated by much smaller write requests from the job mix targeting */scratch*. It is I/O from very large and sustained transfer requests that is most likely to result in individual LMT observations at the peak available transfer rate. There is a great deal of structure in



(a) A low performing IOR test from Figure 2 (b) Contention between read and write activity

Figure 5. Jobs competing for I/O bandwidth can interfere with each other. a) One of the regularly scheduled IOR tests suffered due other activity on the system. b) A period of high read activity appears to suppress the write activity.



(a) A 24 period of heavy I/O on */scratch* (b) Relatively light I/O on */scratch2*

Figure 6. Looking at LMT data 24 hours at a time can give an intuitive sense of whether the file system is busy and what the workload looks like. It is difficult to get a complete or rigorous view for the year as a whole, though, by simply looking at many such daily graphs.

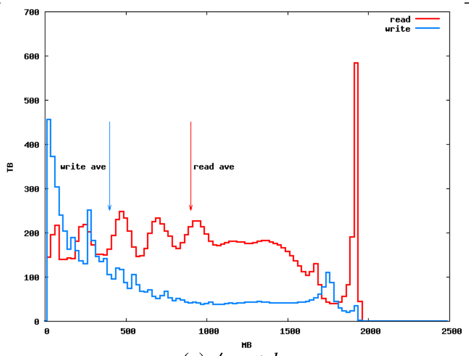
the power spectrum of the LMT data for the file system as a whole and when examined month by month or day by day.

The two values so far used to characterize workload - the aggregate amount of data transfered and the centroid of the power spectrum - both distinguish the workload on the two file systems. These aggregate values cannot represent the more nuanced differences that would come from workloads that differ, for example, in presenting short intervals of high I/O versus long intervals of low I/O. The centroid of the power spectrum cannot distinguish between a workload of identically performing jobs versus a distribution of jobs with the same average behavior, but widely varying individual I/O rates. The next section presents a very different statistical view of the I/O behavior on Franklin and on the way answers a very different sort of question about the workload.

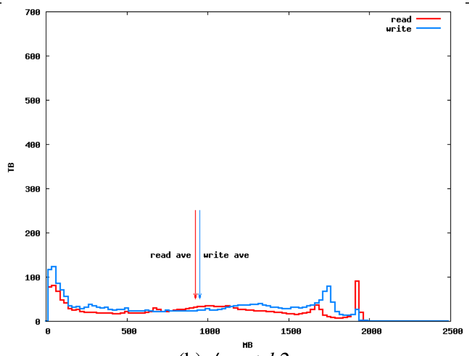
6 The Auto-correlation Function

We make a slight digression into a separate question that also has some interest, and from the results of answering that question we gain insight into our original workload characterization goal.

One may wonder if a job that has a regularly scheduled I/O component of its workflow might be able to benefit from examining how busy the file system is and, if it is busy, deferring the I/O by an amount τ, waiting until things calm down a bit. If the system is busy, how long is that likely to last? If the system is quiet, is it likely to remain quiet long enough to acomplish the needed I/O? This amounts to the auto-correlation of the I/O rate on the



(a) */scratch*



(b) */scratch2*

Figure 7. The power spectrum gives the total I/O contributed in each of 100 bins based on the LMT observations. For example, an LMT observation of $1GB$ transfered to/from a given server in the 5 second interval contributes $1GB$ of I/O to the $1GB$ bin. In a) there would appear to be about 200,000 such observations for read I/O and about 50,000 for write I/O. The centroid of the spectra in */scratch2* are both near the middle of the distribution at about $900MB$, as is the centroid for reads in */scratch*, but the centroid for writes in */scratch* is much smaller at around $400MB$.

file system.

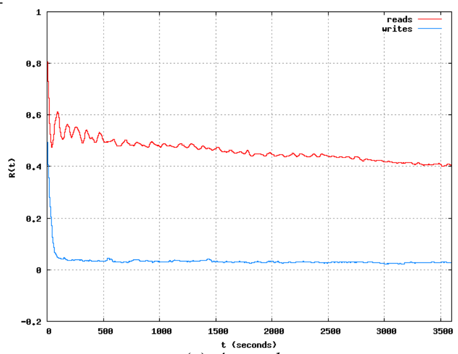
Figure 8 graphs the auto-correlation function (ACF) for April 2010 for both */scratch* and */scratch2*. For τ less than about two minutes it is very likely that the current observed I/O rates will persist for reads and writes for both file systems. Beyond that */scratch2* is likely to be quiet if it was busy or vice versa. On */scratch* the write autocorrelation also drops significantly in the first two minutes, but the read-autocorrelation remains high well past an hour. A computation on */scratch2* that can wait two minutes to save a checkpoint might well do so if the file system is currently busy, and a job that can complete its I/O in two minutes has a reasonable chance of doing so unmolested when it sees the file system is quiecent. On */scratch* the expected wait for a quiescent system is much longer, simply because it is always rather likely that a lot of reads are going on as in Figure 6(a).

As with the power spectrum, the auto-correlation functions change from month to month reflecting changes in the underlying workload, though the */scratch* read autocorrelation is like that in Figure 9 most months, and the */scratch2* auto-correlations seldom show much correlation past a few minutes. In the previous section we saw that the centroid of the */scratch* write power spectrum was half the value of the other spectra reflecting more, smaller writes there. With the auto-correlation calculation we see that it is the reads on */scratch* that stand out as being very different, and in this case it reflects that there is an ongoing high read activity on scratch that persists for many hours at a time.

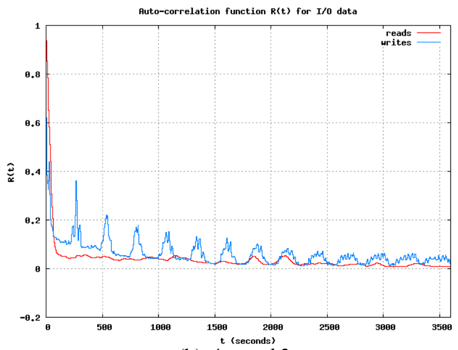
These statistical methods of characterizing the workload on Franklin are helpful in exhibiting the long-term and persistent trends that might not be as apparent by looking at the instantaneous rates as in Figure 6. It is clear that the strategy of segregating the "big I/O" workload on */scratch2* has had a significant effect on how the two file systems are used, but we cannot yet say if doing so saved CPU hours for the users compared to other strategies. In order to evaluate alternative strategies we need one more thing. Knowing which job is responsible for which I/O activity will allow us to count the number of CPU hours lost due to delays induced by I/O contention. In the next section we outline one strategy for extracting this information from the available data.

7 Future Work

In order to answer the original question of whether the segregation strategy actually improved the utilization of the compute resource one additional piece of the puzzle remains. We need to be able to determine how many CPUs were impacted by a given instance of contention. The most direct way to do this is to connect observed I/O with particular jobs. There are several possible ways to approach that, including implementing a compute node monitoring scheme like the Integrated Performance Monitoring library (IPM [8]). The scheme we propose is to infer job-level I/O events from the LMT stream of observations and employ clustering analysis to associate those events with specific jobs. In cases where an otherwise idle file system suddenly transitions to having a high I/O rate, stays there for an extended interval and then returns to idle, it is easy to infer that this was the result of the tasks of a single job acting in consert to carry out the job's I/O. In



(a) */scratch*



(b) */scratch2*

Figure 8. The auto-correlation functions for */scratch* and */scratch2* differ mainly in the reads. The data presented is for the month of April 2010, and is typical for both */scartch* and */scratch2*. For reads and writes on */scratch2* and for writes on */scratch* the auto-correlation tends to fall within ten minutes or so to low correlation, but reads on */scratch* maintain a strong correlation well beyond an hour.

the other extreme, where the file system is very busy and remains so for an extended period, with a high degree of variability in the level of activity from one interval to the next, one may guess that many jobs are busy with I/O simultaneously, and disentangling them may be impossible. A sufficiently clever automated ”event recognizer” may be able to characterize a large fraction of the I/O as being from separate jobs. If enough such job I/O events can be recognized one may be able to analyze their frequency with an eye towards calculating how often and to what extent the events do cause jobs to interfere with each other and therefore how much wasted compute time there was due to I/O contention.

In the proposed scheme we manipulate the LMT data stream using signal processing techniques. Transients in the signal mark significant events. An automatic recognition procedure marks the event start and end points. The procedure starts with the definition of a minimum amplitude level, here estimated from a year of observations to be about $1GB/s$. The signal is convolved with a smoothing kernel to guarantee differentiability and minimize background oscillations. Next, the procedure calculates the first derivative to determine the maximal and minimal turning points of the signal to identify event start and end points. Figure 10 illustrates the processing steps, showing the original signal, the signal filtered with a window of 600 seconds, and the identified events. The criterion for start (S) and end (E) points for a particular peak P_i (Figure 10(b)) considers S_i to be the closest minimum to P_{i-1} in $(P_{i-1}, P_i]$; E_i is obtained similarly. The threshold amplitude and smoothing kernel determine the sensitivity of the algorithm and can be tuned to define the significance of the events. Figure 11 presents a parameter study for the smoothing kernel window size and suggests that there is wide latitude in the choice of that parameter.

The LMT signal $f(t)$ is defined in the time domain. However, it is possible to represent the same function in the frequency domain with sinusoids as the set of basis functions, producing a Fourier transform $F(f)$. Currently, we are building a visualization tool (see Figure 12) that will assist in the exploration of read/write traces in the frequency domain.

A Short Term Fourier Transform (STFT) is a sequence of Fourier transforms with a fixed window size, and a spectrogram is the intensity plot of the magnitudes in a STFT. Color spectrograms have been used for many years in audio spectral analysis and speech recognition. Figure 12 shows the spectrogram visualization tool as it is being used to identify spectral signatures of individual jobs or job phases in the LMT signal.

The upper panel of the visualization tool shows the original LMT stream of read rate values for a 24 hour period. As in Figure 6 the x-axis is time and the y-axis is the aggregate read rate observed by the servers.The bottom panel shows the spectrogram of the signal in the top panel. Again, the x-axis is time - corresponding directly to the signal in the panel above - but now the y-axis is frequency. The colormap represents the amplitude of the frequency component, from blue for the lowest amplitude, through green, to yellow, and finally to red for the highest amplitude. The spectrogram shows well defined regions of event activity with low frequency transients as well as higher harmonics. When condensed to a few descriptive parameters the spectrum during an event can be combined with the other general facts about it, such as how much data was moved, and used in a clustering scheme to pick

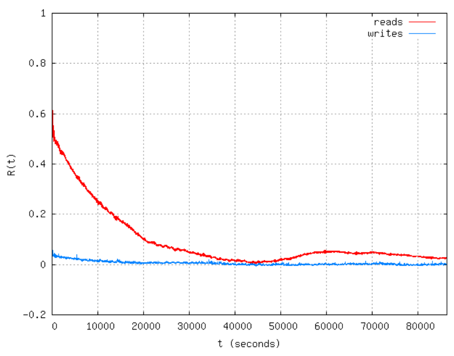


Figure 9. The auto-correlation for read I/O on *scratch* drops to about 0.5 within a few minutes, but when followed to much longer lags shows an ongoing high correlation out beyond two hours. In this graph the auto-correlation is calculated out to a 24 hour (86400 second) lag. The high correlation is because */scratch* is frequently engaged in a constant high read I/O similar to that depicted in Figure 6(a).

out when two jobs are "the same." In conjunction with the job log data this can then allow us to associate specific I/O events with the actual binary that the job was running, which in turn allows us to associate I/O events with size of the job. Armed with this association we can then compute in a rigorous way the cost, in CPU cycles lost, of a delay induced by I/O contention.

8 Conclusions

NERSC more than doubled the I/O resources for the Franklin Cray XT and split those resources in two. The increased resources lead to a better balance between compute and I/O capabilities, and the user community noticed both the improved performance and the decreased variability in I/O performance. NERSC encouraged "big I/O" users to move to using */scratch2* so that they would experience less interference and so that jobs with lower I/O requirements would not have to contend for resources with them.

A review of our monitoring data shows that this segregation was largely successful. We were able to characterize I/O contention on the system using a regularly scheduled I/O test. We observed that read rates are as important as write rates to NERSC users. This appears to contradict the results from the NERSC user survey where PIs reported "predominately write" activity for their I/O intensive applications. This leads us to conclude that the selected projects were not entirely representative of the NERSC workload. Compared to the size of the scratch file systems and the average growth of disk use, the volume of write data shows that users must be managing most of their disk use without system staff intervention.

LMT provides both a low-level, detailed view of the file system and a high-level overview of long term and aggregate use. Average daily rates and historical trends show the workload changing from month to month. The monthly summary statistics and the statistical analysis of the LMT data stream using the power spectrum and the auto-correlation function all show that the workloads on the two file systems differ. The workload on */scratch2* reflects those characteristics consistent with "big I/O" as outlined in the NERSC user survey. Nevertheless, most of the I/O on Franklin still targets */scratch*, raising the possibility that a better balance might be achieved.

From this work one could ask, "Did creating two scratch file systems increase scientific productivity on the NERSC systems, and is the balance of users between *scratch* and *scratch2* ideal?" From the center staff's perspective, after the I/O upgrade and the creation of two file systems, the number of users reporting problems with I/O performance dropped significantly. It is not known however, if maintaining one scratch file system after the I/O upgrade would have produced the same level of user satisfaction. It is clear however, that the */scratch2* file system has lower I/O activity and so users whose applications are sensitive to I/O now have a file system from which to run with little interference from other users. This segregation can be a benefit to both to the "big I/O" users, who have a quieter file system, and to the large number of NERSC users whose I/O is not impacted by those "big I/O" jobs.

9 Acknowledgments

We'd like to thank Jonathan Carter, Shane Canon, Tom Davis, Richard Gerber, John Shalf, David Skinner, Mike Welcome and Zhengji Zhao of NERSC for many valuable discussions of these issues and these results. Steve Luzmoor and Nick Henke of Cray and Eric Barton, Andreas Dilger, and Isaac Huang of Sun, were

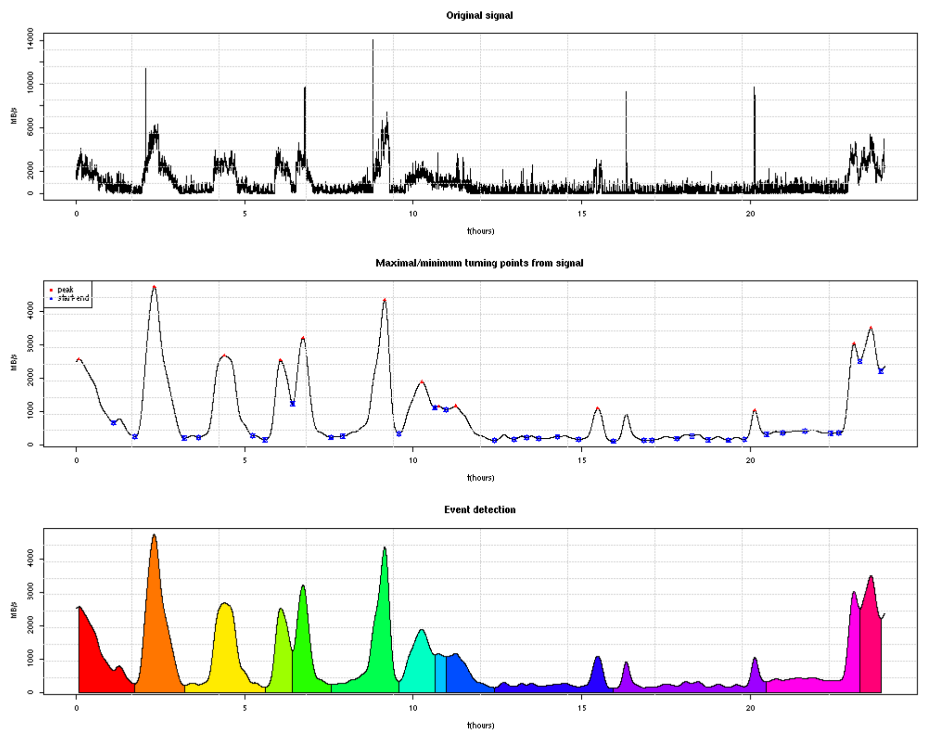


Figure 10. Steps of the algorithm to detect events: (a) read activity on 2010-04-29; (b) smoothed signal, peaks and start-end point candidates; (c) event detection result - different colors indicate different events.

all very generous with their time in explaining Cray XT and Lustre details and encouraging our investigations. We would like to thank Nick Cardo and all the system staff at NERSC for their support in deploying and using the Cerebro/LMT software. Finally, we thank the authors of *Cerebro*: Al Chu, and LMT: Herb Wartens, without whom we could not have collected the LMT data in the first place.

All LBNL authors were supported by the Office of Advanced Scientific Computing Research in the Department of Energy's Office of Science under contract number DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under that contract. The authors were also sponsored in part by the U.S. National Science Foundation through grants CCF 0938114 and the U.S. Department of Energy through the SciDAC program with Agreement No. DE-FC02-06ER25777.

References

[1] K. Antypas, J. Shalf, and H. Wasserman. Nersc-6 workload analysis and benchmark selection process. In *LBNL Tech report 1014E*, 2008.

[2] J. Borrill, L. Oliker, J. Shalf, and H. Shan. Investigation Of Leading HPC I/O Performance Using A Scientific-Application Derived Benchmark. In *Proc. SC07: High performance computing, networking, and storage conference*, Reno, NV, 2007.

[3] P. Braam. File systems for clusters from a protocol perspective. In *Proceedings of the Second Extreme Linux Topics Workshop*, Monterey, CA, June 1999.

[4] Flash code. `http://www.flash.uchicago.edu/`.

[5] The ASCI I/O stress benchmark. `http://sourceforge.net/projects/ior-sio/`.

[6] H. Shan, K. Antypas, and J. Shalf. Characterizing and predicting the I/O performance of HPC applications using a parameterized synthetic benchmark. In *Proc. SC2008: High performance computing, networking, and storage conference*, Austin, TX, Nov 15-21, 2008.

[7] H. Shan and J. Shalf. Using IOR to analyze the I/O performance of HPC platforms. In *Cray Users Group Meeting (CUG) 2007*, Seattle, Washington, May 7-10, 2007.

[8] D. Skinner. Integrated Performance Monitoring: A portable profiling infrastructure for parallel applications. In *Proc. ISC2005: International Supercomputing Conference*, volume to appear, Heidelberg, Germany, 2005.

[9] A. Uselton. Deploying server-side file system monitoring at nersc. In *Cray User Group Conference*, Atlanta, GA, 2009.

[10] Vorpal. `http://www.txcorp.com/products/VORPAL/`.

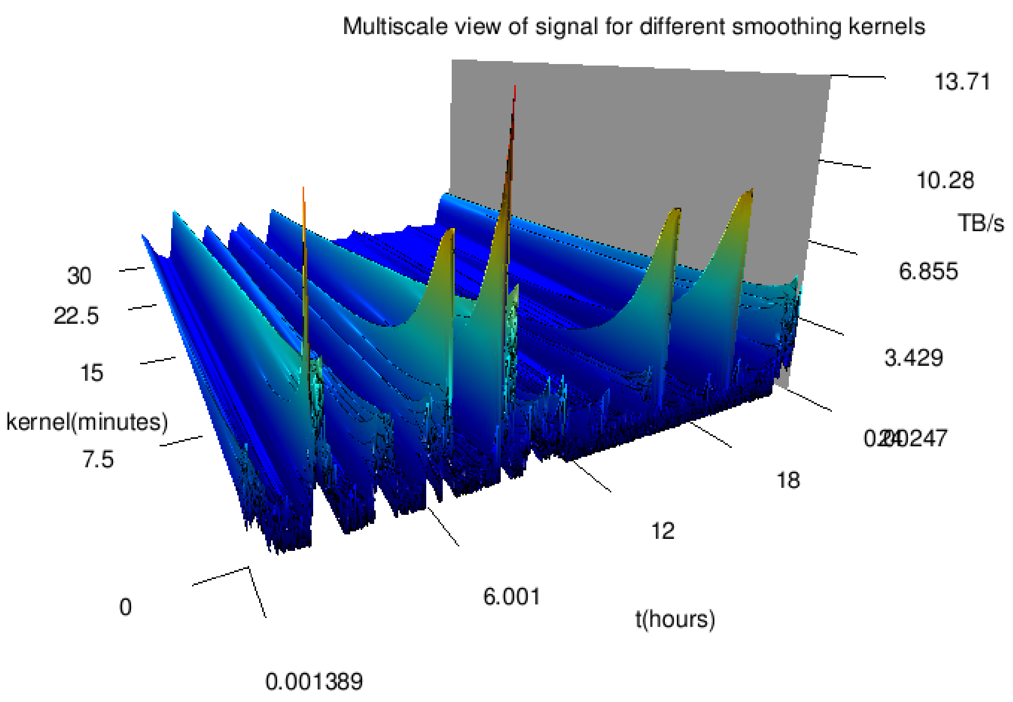


Figure 11. Multiscale representation of signal in Fig.10 that shows that the identity of peaks is stable over a wide variety of window lengths using different smoothing kernel sizes.

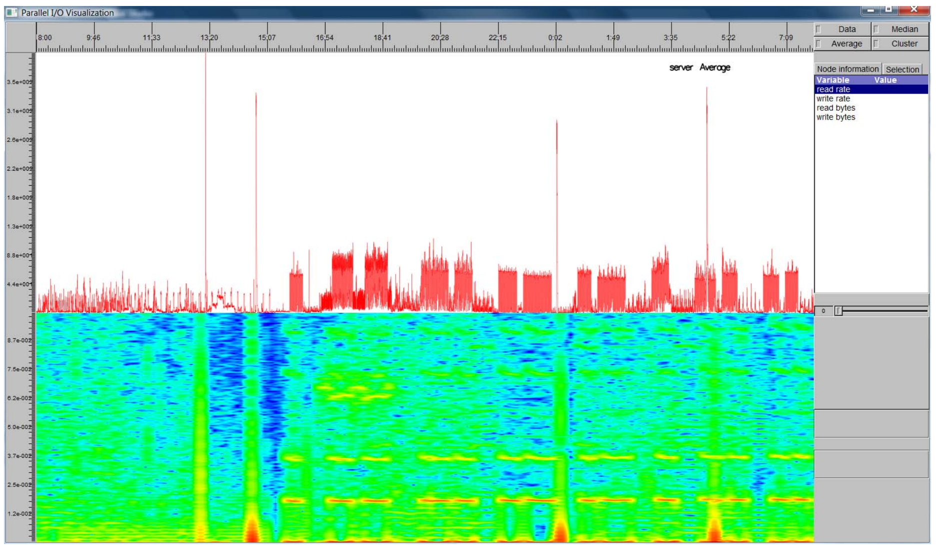


Figure 12. A spectrogram of the LMT read data is constructed via the Short Term Fourier Transform of the signal. The Fourier transform of a moving window about each time step provides the spectrum represented in the heat map in the bottom half of the display.