NERSC Workload Analysis on Hopper

K. Antypas, B.A. Austin, T.L. Butler, R.A. Gerber, C.L. Whitney, N.J. Wright, W. Yang, Z. Zhao
Lawrence Berkeley National Laboratory, Berkeley, CA
Author contact: kantypas@lbl.gov

The National Energy Research Scientific Computing (NERSC) Center is the primary computing facility for the United States Department of Energy, Office of Science. With over 5,000 users and over 600 different applications utilizing NERSC systems, it is critically important to examine the workload running on NERSC's large supercomputers in order to procure systems that perform well for a broad workload. In this paper we show the breakdown of the NERSC workload by science area, algorithm, memory, thread usage and more. We also describe the methods used to collect data from NERSC's Hopper (Cray XE6) system.

Keywords—workload analysis; workload characterization; supercomputing; HPC;

I. Introduction

The National Energy Research Scientific Computing (NERSC) Center [1] serves as the primary High Performance Computing (HPC) facility for the Department of Energy, Office of Science (SC). NERSC supports the entire spectrum of SC research, and its mission is to accelerate the pace of scientific discovery through high performance computing and data analysis. An Office of Science user facility, NERSC serves over 5,000 scientists annually throughout the United States and the world, supporting over 700 distinct projects utilizing more than 600 discrete applications. These researchers, working remotely from Department of Energy laboratories, other Federal agencies, industry, and universities, use NERSC resources and services to further the mission of SC. Computational science conducted at NERSC covers the entire range of scientific disciplines, but is focused on research that supports DOE's missions and scientific goals. The results of the scientific use of NERSC are documented in over 1,500 peer reviewed scientific papers per year as well as in NERSC annual reports and other materials.

Because of NERSC's large number of users and the diversity of applications running on the NERSC systems, it is extremely important to understand the workload and how it is changing so that NERSC can procure systems that best meet the needs of its users. NERSC procures a new supercomputer ever three to four years and at the beginning of every new project, NERSC conducts a thorough workload analysis to understand the characteristics of applications running on the NERSC systems. [2] In this paper we describe the methods used for collecting workload data, and the results of our workload analysis. The NERSC workload analysis was conducted on the Hopper [3], Cray XE6 system. The Hopper system has over 6300 compute nodes, each with two AMD Opteron 12 core (Magny Cours) processors. The nodes on Hopper are connected via the Cray Gemini interconnect. The Hopper Lustre [4] parallel file system has over two petabytes of disk and provides 70 GB/sec of I/O bandwidth. The workload analysis was conducted during 2012 and the first half of 2013. During this period, Hopper supplied approximately 95% of the computational cycles to NERSC users and thus provided a representative sample of the NERSC workload. As part of the analysis NERSC examined the breakdown of science areas, applications and algorithms on Hopper, job sizes, memory and threading usage as well as the top libraries used on the Hopper system.

II. Methods

A. Collecting Job Data from Hopper

All jobs that run on Hopper, whether they use one or more nodes, are submitted by users through a Torque/Moab batch system that integrates with Cray's Application Level Placement Scheduler (ALPS). Users write and submit batch scripts to Torque, requesting some number of nodes for some length of time. From within that script, users can run one or more programs by passing an executable name to the ALPS *aprun* utility, which launches and manages the program executables on the Hopper compute nodes. Programs launched via *aprun* can use some or all of the nodes allocated to the job by Torque/Moab. Each job and each *aprun* gets exclusive use of each node it is allocated; there is no node sharing on Hopper.

Information about completed jobs is available from Torque accounting logs and from informational messages emitted by ALPS. The Torque logs contain job characteristics like time submitted, start time, end time, number of nodes requested, wallclock time requested, a list of allocated node IDs, user name and account. ALPS logs the start time, end time, node IDs, and full command line for each *aprun* command. From the *aprun* command line, we can infer the number of threads used and a number of thread and memory affinity settings.

NERSC collects data from the various job log files, parses the data and stores them in a MySQL job database. This allows NERSC to easily query the data and to create reports for select users, science areas, or time frames.

Cray supports a resource collection method called Cray Application Resource Usage (ARU) tool. This tool was created because ALPS does not pass resources, (such as application memory data), up to batch systems such as Torque/Moab used on Hopper. Instead these resources are collected by ARU from a separate ALPS logfile. It was a simple extension to NERSC's MySQL job database to include the new ARU data.

B. NERSC Information Management (NIM) System

NERSC maintains a database, called the NERSC Information Management system (NIM), of user and project information. All jobs at NERSC are associated with both a project (known as a *repository* or *repo*) and a user. Each project is allocated computing resources by a DOE program manager from one of six DOE Office of Science program offices.[1] Each project is also assigned a NERSC *science category*.

Every job run at NERSC is associated with a project (repo), and resource usage is attributed to DOE offices and science categories by joining the account field from the Torque accounting logs (described above) and repository information from the NIM database.

C. Automatic Library Tracking Database

The Automatic Library Tracking Database (ALTD) infrastructure developed at the National Institute for Computational Sciences (NICS) [4], is a tool that can track all libraries that are linked into applications at compilation time. ALTD was installed and put into production on the Hopper system in June 2012 and since then NERSC has been collecting information about the libraries run on Hopper. ALTD can also track the applications that are executed through a batch system by wrapping the job launcher, aprun, on Cray systems. ALTD is implemented in Python, and uses a MySQL database to store the library usage information that was captured by intercepting the GNU linker, ld. These data can then be mined to generate library usage reports. ALTD is light-weight in that it does minimal logging to wrap the linker and the job launcher keeping overhead at link time and job startup time negligible. The ALTD ld wrapper captures the following information for each successful linking:

- username (who linked the code)
- link date (when an executable is linked)
- executable name
- all libraries that are linked into the executable.

It should be noted that the ALTD ld wrapper only records the libraries actually used by the executable. This is implemented by calling the linker with the tracemap (-t) option, such that libraries included on the link line, but not called in the executable will not be logged. These are stored in a MySQL database and by querying for a pattern that is specific to a library, it is possible to determine the number of times a particularly library was linked and the number of unique users of that library. In addition, the ALTD ld wrapper adds a unique tag into the executable which can be captured and recorded in the database when the executable is launched by the aprun wrapper. This tag is the pointer to the libraries linked into the executable that actually runs on the system. Together with other tools that can report the machine hours used by the executables that run on the system, it is possible to track the actual usage of libraries on the system (not only the compilation/linking time).

[1] The DOE Office of Science Offices: Advanced Scientific Computing Research (ASCR), Basic Energy Sciences (BES), Biological and Environmental Research (BER), Fusion Energy Sciences (FES), High Energy Physics (HEP), and Nuclear Physics (NP).

III. Workload Analysis Characterization

The following section describes the results of the NERSC workload analysis.

A. Science Area Breakdown

Figure 1 shows a breakdown of Hopper CPU hours by science area. NERSC supports all six Office of Science Offices: Advanced Scientific Computing Research (ASCR), Basic Energy Sciences (BES), Biological and Environmental Research (BER), Fusion Energy Sciences (FES), High Energy Physics (HEP), and Nuclear Physics (NP). At NERSC, users are given an allocation of time on the systems from the DOE Program Managers and so the breakdown of time on Hopper is representative of the allocation of time by each of the Science Offices to various projects. As priorities within the Office of Science change from year to year the allocation given to a particular science area will change.

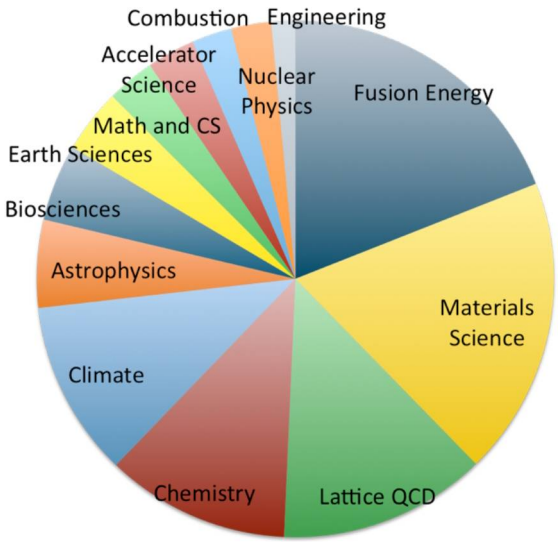

Fig. 1. Hopper CPU hours broken down by science area.

B. Application Code Breakdown

Figure 2 shows a breakdown of the top codes at NERSC by the number of hours used on Hopper in 2012. The colors are not meaningful and only intend to break up the pie chart to be more readable. The chart shows a highly concentrated and unevenly distributed workload with three applications making up 25% of the workload and 10 applications making up 50% of the workload. 35 applications make up 75% of the workload with the remaining more than 600 codes comprising the remaining 25% of the workload. The top three applications run at NERSC in 2012 were the Community Earth System Model (CESM), VASP, a Materials Science, plane wave density functional theory electronic structure calculation application,

and MILC a Lattice Quantum Chromodynamics (QCD) application. The top 25 applications are listed in Table 1.

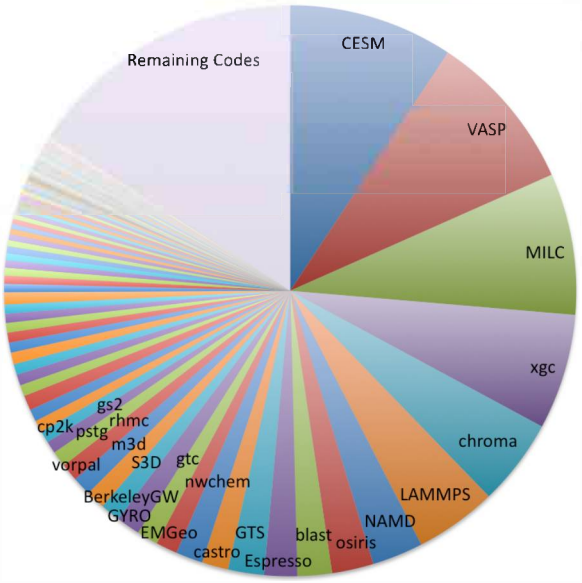

Fig. 2. Hopper CPU hours broken down by code usage.

TABLE I. TOP 25 APPLICATIONS ON HOPPER IN 2012

Application Name	Science Area
CESM	Climate
VASP	Materials Science; Plane Wave DFT;
MILC	Quantum Chromodynamics
XGC	Fusion, Particle-in-cell
CHROMA	Quantum Chromodynamics
LAMMPS	Molecular Dynamics
NAMD	Molecular Dynamics
OSIRIS	Fusion, Particle-in-cell
BLAST	Bioinformatics
ESPRESSO	Materials Science, Plane Wave DFT
GTS	Fusion, Particle-in-cell
CASTRO	Adaptive Mesh Refinement, astrophysics
NWCHEM	Chemistry
EMGEO	Geophysics
GTC	Particle in cell Fusion application
BERKLELYGW	Materials Science, Plane Wave DFT
GYRO	Eulerian gyrokinetics; Fusion
S3D	Combustion
M3D	Fusion, continuum
VORPAL	Accelerator, Particle-in-cell
RHMC	Quantum Chromodynamics
PSTG	Fusion
CP2K	Materials Science; Plane Wave DFT
GS2	Fusion, continuum
GROMACS	Molecular Dynamics

C. Algorithm Breakdown

The apparent complexity of the NERSC workload as shown in Figure 2 can be simplified by recognizing that groups of applications often share similar underlying algorithms. This observation is analogous to earlier insights that a small number of computational kernels account for most HPC use [6]. However, a categorization of algorithms acknowledges that a complete application will often be composed of multiple kernels, and optimal performance of the full application may require different implementations of each kernel. While no codes account for more than about 10% of the available compute resources, Figure 3 shows that the top five algorithm classes (Fusion particle-in-cell, Lattice QCD, plane-wave density functional theory, climate and molecular dynamics) each represent 8-12% of the workload. With only thirteen categories, we can describe nearly three quarters of the applications.

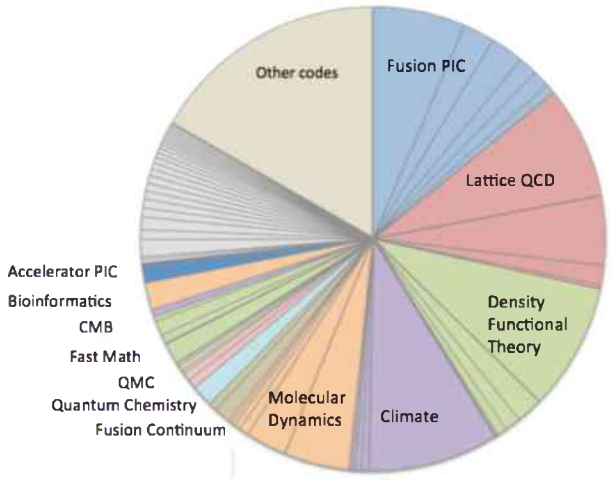

Fig. 3. Hopper CPU hours broken down by algorithm area.

D. Third Party Application Support

The NERSC User Services Group installs and supports an array of applications on the NERSC systems, primarily Materials Science and Chemistry applications. Figure 4 shows that 23% of the hours used on the Hopper systems were applications supported and installed by NERSC staff. VASP, NAMD and LAMMPS are the top three applications that NERSC staff support.

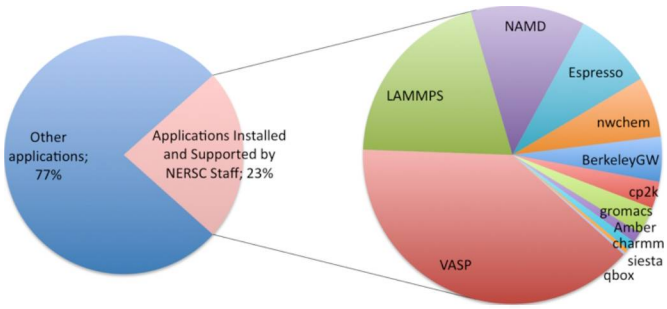

Fig. 4. Third Party Application Support

E. Job Size Breakdown

NERSC users run applications at a wide range of job sizes. Some user applications run across all the nodes in the system while other users run applications on smaller number of nodes and submit many jobs to do parameter studies or high throughput screening. Figure 4 shows the percentage of hours

run on Hopper at various sizes. Approximately 15% of computational cycles use more than 40% of the compute nodes and 40% of cycles use more than 10% of the system. Because of the large range of job sizes it is important for NERSC systems to have a fast and scalable interconnect to support large jobs in addition to having a robust batch system which can support many smaller concurrent jobs.

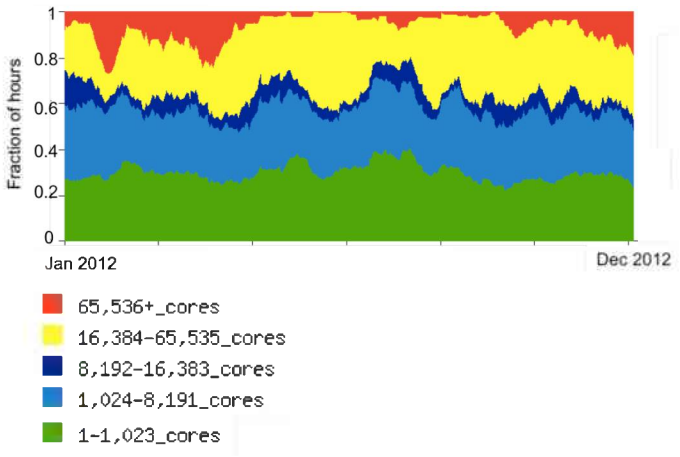

Fig. 5. Hopper Job Size Breakdown

F. Memory Usage

Understanding task memory usage of a given workload is important as the computer industry moves towards architectures with more cores per node with less memory available per MPI task. Hopper has approximately 6300 compute nodes, each of which has two, twelve-core AMD Magny-Cours processors, for a total of 24 cores per node. 6,000 nodes have 32 GB of memory, and the rest have 64 GB. To illustrate, for a 32 GB node, if all 24 cores were running an MPI task, each MPI task would have 1.3GB of memory available to it. (In practice this number is slightly lower due a small amount of memory overhead from the OS and file system services on each node.) Figure 6 provides information on application memory usage on the Hopper system. The plot is a histogram of memory usage per MPI task and shows both the percentage of node hours in each memory range bucket as well as the integrated percentage of node hours. The horizontal axis is memory high watermark in GB per MPI task. The vertical axis shows the percentage of total node hours run within each histogram bucket range.

Figure 6 shows that close to 50% of user node hours used less than 0.3 GB per MPI task. With 24 cores on a node, this means an application using 0.3GB of memory per MPI task, could easily utilize all 24 cores on the 32 GB node. More than 20% of node hours used between 0.3 and 0.7 GB per MPI task; about 12% used between 0.7-1.0 GB per MPI task; and about 7% used between 1.0-1.3 GB per MPI task. Altogether, about 87% of user node hours went to jobs using 1.3 GB per MPI task or less, which indicates that these applications could run utilizing all 24 cores per node. The 13% of node hours with jobs using over 1.3GB of memory per MPI task would either have to run 'un-packed', with fewer than 24 cores per node ,or use the small number of large memory nodes.

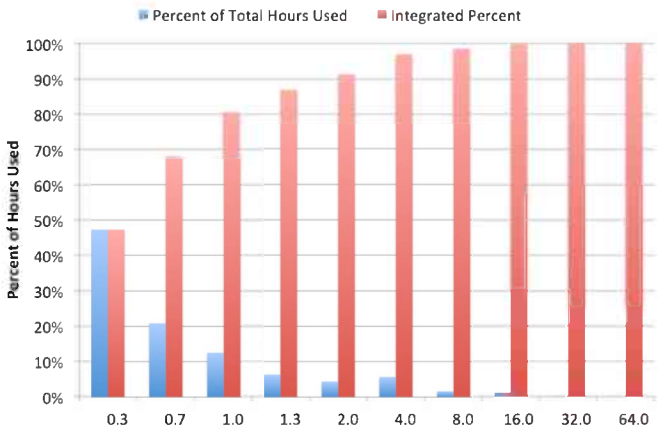

Fig. 6. Per-task memory high watermark in GB

G. Thread Usage

Another interesting statistic that can provide insights about the workload is the number of threads that applications are using per MPI task. Figure 7 and Figure 8 show the breakdown of Hopper node hours by threads per MPI task.

The figures indicate that almost 80% of node hours are used by jobs using only a single thread per task. From requirements workshops [7][8] with NERSC users, it is known that MPI and OpenMP are the dominant programming models, making it likely that the 80% of node hours using a single thread are MPI-only jobs, though, SHMEM applications, Co-array Fortran applications or UPC applications could also fall into this category. A smaller but distinct peak is found for six threads per MPI task, which is the number of cores per NUMA node on Hopper, and the one that is found to give most efficient use of computational performance since six threads would be mapped to the nearest memory region [9]. Almost all the user applications (98%) used six or fewer threads per MPI task. There are very few jobs with 12, 16 and 24 threads per MPI task.

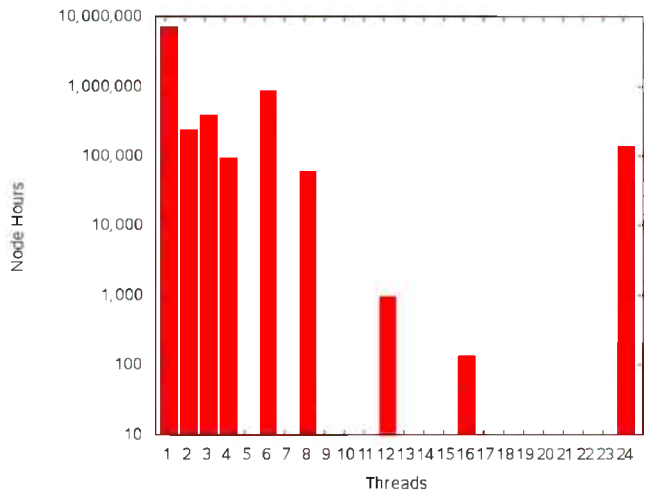

Fig. 7. Histogram of Hopper node hours by threads used per MPI task

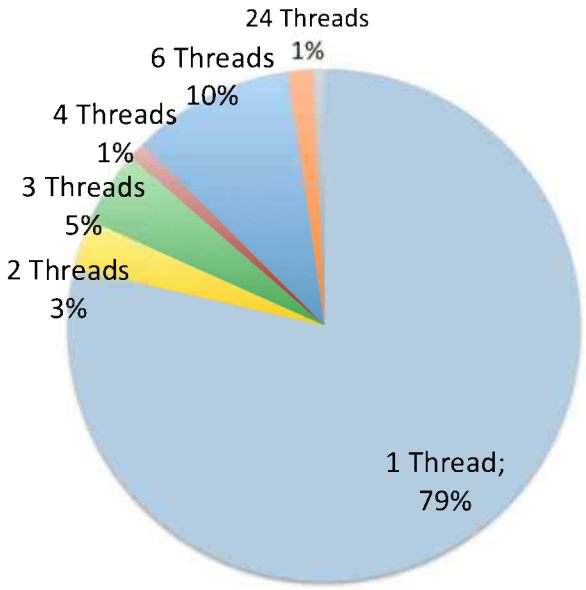

Fig. 8. Breakdown of Hopper CPU hours by number of threads used per MPI task

H. Library Usage

Figure 9 shows the top libraries used on Hopper by number of unique users collected using the ALTD tool. The five most popular libraries are: mpich, the library implementing the MPI standard on Hopper, zlib, a compression library used within many other libraries, libsci the highly optimized package of math libraries provided by Cray, followed by hdf5 and netcdf, both self-describing, portable, I/O libraries. Information on library usage provides information to NERSC on the value of each library and how many resources and staff to devote to supporting each one. Understanding user library preference also provides guidance to vendors about which libraries to spend the most time optimizing and supporting.

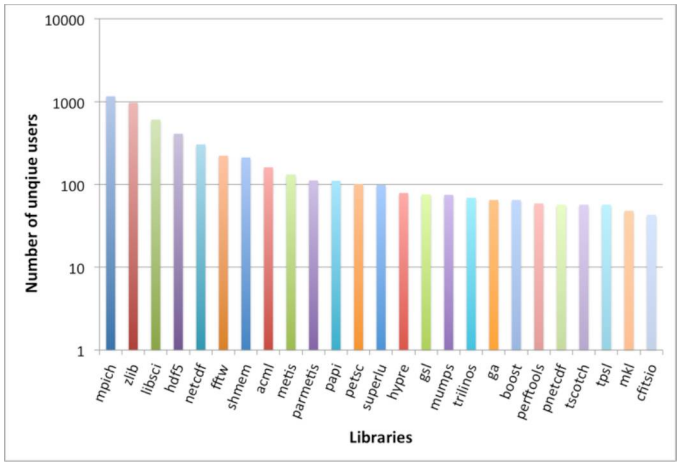

Fig. 9. Top 25 libraries used on the Hopper system by number of unique users

IV. Analysis and Conclusions

The NERSC workload analysis shows the breadth and the diversity of science areas, applications, algorithms and jobs sizes. By some metrics however, the NERSC workload is more homogeneous. Regardless of application or science area, the workload analysis shows the majority of hours on Hopper, almost 80%, are from applications using a single thread per MPI task. Furthermore, close to 90% of applications, use fewer than 1.3GB of memory per MPI task.

NERSC regularly conducts workload analyses and in the future will look more closely at memory and threading requirements. With computing architectures moving towards more cores per node with less memory per core, applications may need to add threading, such as with OpenMP, and reduce memory usage. The positive news for the NERSC workload is that many applications use less than 1GB of memory per task alread. More worrying is that only 20% of applications are running with multiple threads per MPI task. One possibility is that applications running on Hopper have OpenMP threading implementations, however the OpenMP threading isn't being executed because of the large memory per core available on Hopper allowing users to run with MPI-only.

It is critically important that NERSC continually study its workload in order to adapt to user needs and to provide the systems, services and software to support the broad workload.

Acknowledgment

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

DISCLAIMERS

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

COPYWRIGHT NOTICE

This manuscript has been authored by an author at Lawrence Berkeley National Laboratory under Contract No. DE-AC02-05CH11231 with the U.S. Department of Energy. The U.S. Government retains, and the publisher, by accepting the article for publication, acknowledges, that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. Government purposes.

REFERENCES

[1] S. Dosanjh, S. Canon, J. Deslippe, K. Fagnan, R. Gerber, L. Gerhardt, J. Hick, D. Jacobsen, D. Skinner, N.J. Wright, "Extreme Data Science at the National Energy Research Scientific Computing (NERSC) Center", *Proceedings of International Conference on Parallel Programming – ParCo 2013,* (March 26, 2014)

[2] K. Antypas, J. Shalf, H.J. Wasserman, "NERSC-6 Workload Analysis and Benchmark Selection Process", *LBNL Technical Report,* August 13, 2008, LBNL 1014E

[3] K. Antypas, T. Butler, J. Carter, "The Hopper System: How the Largest XE6 in the World went from Requirements to Reality," Cray User Group 2011 Proceedings, May 2011.

[4] T. Butler, "DVS, GPFS and External Lustre at NERSC - How It's Working on Hopper", Cray User Group Meeting, Fairbanks, AK, May 2011.

[5] M. Fahey, N. Jones, B. Hadri, The Automatic Library Tracking Database, Conference: Cray User Group 2010, Edinburgh, United Kingdom

[6] P. Colella, Defining software requirements for scientific computing. Slide of the 2004 presentation included in David Patterson's 2005 talk, (2004); http://www.lanl.gov/orgs/hpc/salishan/salishan2005/davidpatterson.pdf

[7] R. Gerber and H.J. Wasserman, eds., "Large Scale Computing and Storage Requirements for High Energy Physics - Target 2017", November 8, 2013

[8] R. Gerber, H.J. Wasserman, "High Performance Computing and Storage Requirements for Biological and Environmental Research Target 2017", June 6, 2013, LBNL LBNL-6256E

[9] N. Wright, H. Shan, A. Canning, L.A. Drummond, F. Blagojevic, J. Shalf, K. Yelick, S. Ethier, K. Fuerlinger, M. Wagner, N. Wichmann, S. Anderson, and M. Aamodt, "The NERSC-Cray Center of Excellence: Performance Optimization for the Multicore Era", Cray User Group Meeting, Fairbanks, AK, May 2011.