Beyond Performance Tools: Measuring and Modeling Productivity in HPC

Michael O. McCracken
UCSD CSE
mike@cs.ucsd.edu

Nicole Wolter
San Diego Supercomputer Center
nickel@sdsc.edu

Allan Snavely
UCSD CSE
allans@sdsc.edu

Abstract

Emerging challenges to productivity are not well covered by traditional methods for evaluating HPC programs, systems, and practices. The common measure of merit widely put forward in High-Performance Computing (HPC), high computational performance as measured in floating-point operations per second (FLOPs), does not account for many bottlenecks in real HPC workflow that increase time to solution which are unaffected by performance changes. In this paper we discuss these bottlenecks, show an approach to analyzing productivity based on measurement and modeling of HPC workflow, and present plans for measurement and experimentation tools to study and improve productivity in HPC projects with large computational and data requirements.

1 Introduction

Using high-performance computing (HPC) resources is a prerequisite for a great deal of current scientific research. Projects using those resources require a great deal of effort, knowledge, and planning to be successful and productive. Simply having access to the most recent technology is not a guarantee of success. Recent research into the productivity of HPC systems and users has explored the relationship between time, effort and productive returns, by quantifying development time effort [4, 5, 10], queue wait times, large project success factors [8, 2], and by more clearly defining productivity itself [6]. In this paper, we discuss work toward measuring and modeling the productivity of HPC users in the context of multi-site allocations with large computational and data requirements. The subsequent goals are for supporting a better understanding of productivity in HPC and building tools to help users become more productive. Because the exact definition of the total time-to-solution can change depending on the project and its current goals, we do not propose a single definition of productivity here. Instead, we present a method and propose tools that allow analysis of productivity however it is defined for a project.

To be productive, computational scientists are faced with an optimization problem with important trade-offs between time spent optimizing their research workflow and time spent generating research results. Because of changing policies, system and network conditions, architectures, and research goals, high productivity can only be achieved through striking a balance between improving code and using it. Unfortunately, the information needed to make decisions about these trade-offs is often uncertain or unavailable. For example, queue wait times are often unpredictable and site scheduling policies are dynamic and are not always publicized. Decisions about whether or not to optimize are usually made without a good estimate of the performance improvement to be expected.

In studies of user behavior and experiences [12], we found that user attitudes toward existing performance tools reflected that current tools do not address some of the most significant roadblocks to productivity that users experience. In fact, one user with large computational and data requirements described computational performance as "sinking toward the bottom" [3] of the list of bottlenecks to getting results. Problems such as queue wait time, data transfer speed, storage capacity, portability, and reliability are often more pressing, but are not currently addressed by performance tools. We think that there is a need for tools which support measurement and analysis of these problems alongside program performance. We further suggest that given a *productivity tool* that measures productivity factors and supports decisions about the trade-offs inherent in HPC usage, users will be able to diagnose productivity problems more easily and generate results faster with less time spent waiting. This work is the beginning of a course of research which will test that hypothesis.

2 Productivity Analysis

A natural approach to studying the causes of productivity loss is to apply methods developed for studying causes of performance loss. Therefore, the first step in analyzing a productivity problem is to determine the set of tasks that define the user's workflow, and measure the time spent in each task. This critical first step allows us to verify intuition about bottlenecks and perform controlled experiments. Keeping records of these measurements will allow retrospective study of the effects of decisions about where and how to allocate resources of time and computation. The next steps in productivity analysis are to model the task, and use that model to guide future user decisions. While HPC productivity certainly is affected by a complicated set of factors, we hypothesize that there are a few measurable tasks which account for enough of the overall time of a project to generate a useful model.

Modeling overall productivity enables evaluating potential usage patterns quantitatively, and despite a good deal of inherent uncertainty, we believe it can provide useful information about the productivity effects of common decisions. For example, a project may be faced with the decision to either stay on a heavily-used system, or to port their code to another system with lower queue wait times. A model can give a guideline for how quickly the project must be ported in order for the switch to save time overall.

In the following sections, we explain our approaches to modeling HPC user workflow, both a complete model for experimentation and a simplified form that is more feasible for measurement in large real-world projects.

3 A Complete Workflow Model

Based on our interviews and survey of academic HPC users [12], we developed a comprehensive list of user tasks and a workflow model that describes the paths that users can take among those tasks. Our goal was to develop a model that could both describe the majority of projects and would be simple enough to verify the model's ability to represent real workflows by discussing the model with users.

We began with a rough classification of HPC tasks into three states: development, production running, and post-processing and analysis. We then followed a systematic approach to refining the model that began with those three states and divided them when necessary based on the following four principles for defining task states:

1. Simplicity is important to make data gathering and analysis possible.
In order to verify the models in interviews and support automatic data gathering, the number of states and overall complexity of the model must be kept to a minimum.

2. Consistency is important for understandability.
We avoided defining tasks at different levels of abstraction, so that one task would not appear to be a component of another.

3. Additional complexity must be justified by making a useful productivity distinction.
For example, having two "run" states, one for production and one for testing, is justified because time spent testing does not directly generate results.

4. Potential outside influences on productivity should be represented explicitly.
For example, because post-processing and execution are sometimes performed on separate machines, those tasks should be distinct, in order to represent the influence of system configuration and architecture.

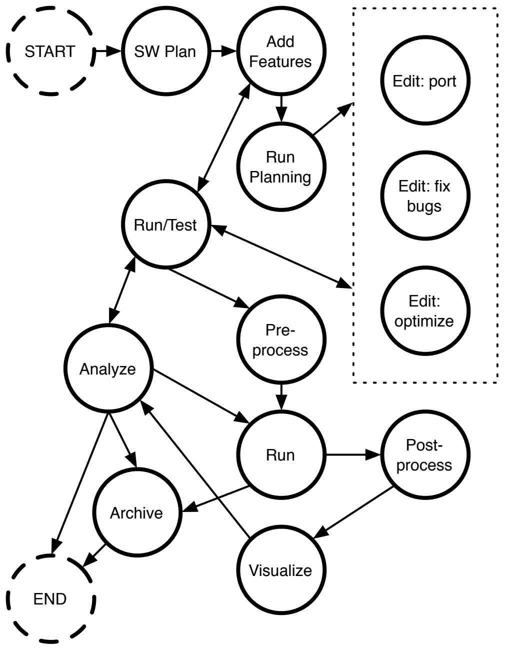

Figure 1. A Real User's Workflow

The final model consists of the 13 task states shown in Figure 1 (and listed in Table 1), and is annotated with all the likely transitions between the states found in on our observation of users. We distributed the model, without the transitions, to a sampling of HPC users and asked them to describe their path through the tasks and time spent in each. We found during the interview process that using this model to guide discussion of productivity is very useful to learn about the patterns of time spent and how users view the tasks they perform.

We found that a project workflow vary distinctly between users. While some workflows include 13 nodes, others only include a few, depending on the scope of the their project. An example of a real user workflow which includes all 13 states is shown in Figure 1. This workflow is derived from an interview about a graduate-student project in fluid dynamics, whose scope included designing a program to solve a fluids problem, compute results, and report on those results.

This complete model will serve as a tool for describing user workflow and simulating the effects of changes in system parameters or user behavior on productivity.

4 A Focused Model for Measurement

Each state in the complete workflow model presented in Section 3 represents time spent in a task that must be measured separately, and some of the differences between the states are not straightforward to determine automatically. Because of the potential for such measurement to interfere with the user's actual work, it is not practical to begin by measuring each state in detail and still study the large projects that motivated this research.

Therefore, we defined a simplified model that captures two of the the most important bottlenecks found in our studies of HPC users, while retaining the ability to describe a majority of common workflows. By focusing on two important bottlenecks and consolidating the 11 other original task states, we create a hierarchy of models, in which a more complete model is an elaboration on our simplified model, and there is a clear mapping between states in each model, as shown in Table 1.

The simplified model has four task states - two "working" states and two "waiting" states. Figure 2 shows the model, with the "working" states shaded. We chose to define "Queue and Run" and "Data Transfer" as the two waiting states because they were the most-often cited productivity problems in our user interviews. These variables are measurable, and their influence upon users behavior can be examined through observing queue trends, and using personal interviews and surveys. The working states include everything else, at a high level of abstraction. While this model does not differentiate between many distinct states, such as planning, debugging, and development, it describes productivity problems that have not been studied in the context of user workflow. The model will serve as a starting point for our work on measurement and decision-support tools that is discussed further in Section 6.

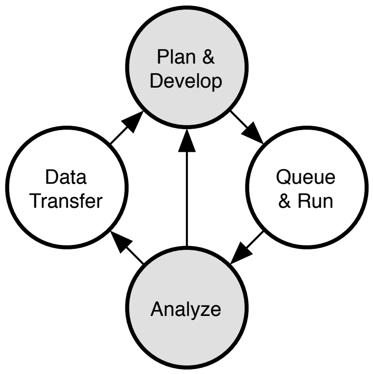

Figure 2. Simplified Workflow Model

Table 1. Job states in complete model, organized by containing node in simplified model.

Plan & Develop	Queue & Run
Software/Project Planning	Preprocess data
Debug	Run Production
Port	Run Tests
Optimize	
Add Features	
Plan Runs	
Analyze	Transfer Data
Analyze data manually	Archive or move data
Postprocess data	
Generate Visualizations	

4.1 Data Transfer

In contemporary HPC systems, efficient data transfer between systems, and to disk and tape, is essential for sustained productivity. Currently, users regularly move significant amounts of data, archiving to tape based storage, to use their distributed compute-time allocations, and to use unique resources at some locations. Because of the high demand for limited resources, large allocations are often distributed between sites to guarantee fair access to each system. This taxes the productivity of large projects, who must transfer large data sets over relatively slow network links to continue their research.

Disk space and data retention policies can also cause problems with projects that produce huge amounts of data. HPC programs generally write to large, shared temporary file systems, separate from the persistent user home directories, which tend to have relatively small quotas. At many sites, files untouched for four to seven days will be indiscriminately purged. Data backup to archival storage is therefore essential, and depending on the data size and the location of archival storage, this can be a workflow bottleneck.

For example, on a IBM Power3 system, Enzo, a cosmology simulation using Adaptive-Mesh Refinement [7], can produce 13 Terabytes per 18-hour run, writing a 700 Gigabyte data set every 45-60 minutes. For perspective, the DataStar system at SDSC [9], more powerful and capable of producing more data, has a total of 180 TB of disk storage for at least 126 allocations. On such systems, transfer to archival storage is on the critical path for working with Enzo.

Our studies revealed that users experience data transfer rates between systems that range from 35 Megabytes/sec and 350 Megabytes/sec to 1 Terabyte/sec, depending on the systems involved and the software used for the transfer. These transfer rates will soon be insufficient to keep up with productivity if trends in computational power prove true. This means that for applications with increasing data handling requirements, productivity will suffer when running on systems not designed for high end-to-end I/O speed - not just fast processors, memory, or network links between supercomputers, but correspondingly fast I/O subsystems and archival data storage systems as well.

4.2 Queue Wait Time

Supercomputer job queues are among the most-visible and often-discussed productivity problem in HPC. There is a tension between users, many of whom certainly dislike time lost while waiting, and centers, whose funding is at least partly justified by maintaining high utilization of their systems. It is important to note that there is a class of users who are relatively unaffected by queue wait times, because their jobs are small and backfill or are part of a large set of jobs with a focus on throughput.

As part of the DARPA High Productivity Computing System (HPCS) program, there has been some discussion recently about the effects of queueing on productivity. For example, a well-attended birds-of-a-feather session at the Supercomputing 2006 conference, was held to discuss current scheduling policies at major centers in terms of their productivity [11]. In our own research, we have come across many instances where queue wait time has drastically inhibited productivity. One user estimated that running one simulation, which takes approximately three months under current policies, would take just a week if he were given all of a current system to run continuously - maintaining 100% utilization.

Because supercomputer centers are not likely to start under-allocating systems to minimize wait time, users who are upset about wait time have few options. They can switch to less utilized systems if they have allocation there, or alter their job size to backfill, if their problem allows it. However, users are often reluctant to switch systems, despite potential productivity improvements. Reasons cited for this reluctance include relationships with user support and the concern that moving to a different system would incur data transfer costs to use visualization or archival resources at the original site. This situation results in inefficient use of computational resources.

5 Related Work

Most single-task aspects of HPC usage have been studied in great depth. Program performance, the motivating purpose of HPC, is well-covered, and many tools exist that support performance measurement and analysis. Modeling parallel performance is also an active field, including a variety of approaches at various levels of detail. Another robust field is batch scheduling, relevant here as it relates to queue wait time and system utilization. In these fields, the literature is too broad to treat well here.

Some recent studies have focused on productivity of parallel program development. Hochstein et al. [4, 5] studied development time among graduate students learning parallel programming. One use of their data was to evaluate productivity differences between shared-memory programming models and MPI. Another project builds Timed Markov Models of programmer workflow to facilitate quantitative comparison of workflow steps between students using different programming models [10]. Our approach to defining workflow is similar, but we study the workflow of entire large projects, addressing different bottlenecks.

There are tools available to help users cope with queues, including tools which give job run time approximations based on the current state of the batch scheduler, as well as tools which help establish if there are backfill opportunities available on a system. In [1], Brevik et al. discuss a technique for predicting bounds on queuing delay for jobs submitted to shared systems, which allows users to optimize for turnaround time by changing where they submit jobs. A tool based on this work is available at SDSC. These tools are useful to aid planning, but do not involve measurement or modeling of those productivity factors. They would complement tools generated by our work.

In [6], several authors discuss the problem of defining productivity and producing metrics to compare systems and practices. These papers present measures for the productivity of a system or of a programming methodology, which may be useful for determining policy or initial language choices. Most recent efforts also define productivity based on program performance and development time, while we choose to use a broader definition, having found evidence that performance is not always the major roadblock for productivity [12]. To our knowledge, our work is the first to suggest measuring and analyzing productivity factors at a project level across multiple systems, as is now often done with performance.

In studies of productivity of large computational science projects in U.S. government agencies, Post and Kendall [8] formulated a list of common tasks and a workflow graph describing the stages in a project. There is significant overlap between their tasks and the states in our full workflow model, and we have benefited from their work. Our model is intended to be measurable and to support simulation, so it describes in greater detail the user actions that are affected by HPC systems and policies, and leaves out more organizational aspects. Our work also shows more of an emphasis on the challenges faced by academic HPC users, such as split allocations and shared access.

6 Future Work

The path taken by research in performance tools provides a template for building productivity tools - first, reliable and low-overhead measurement of the potential bottlenecks must be supported, followed by modeling and analysis later, to support reasoning about complicated situations such as multiple-site, long-term project planning and potentially, procurement planning.

We have begun by characterizing the overall task of using HPC systems, and recording user experiences. Using that information, we have developed a model that highlights the most important bottlenecks to productivity. We are currently building measurement tools to help record time spent blocked in unproductive states, with a goal of minimizing extra work for a user to keep this information. Using the simplified model from section 4, we will record data transfer times and queue wait times from running projects. The data will be used to evaluate effects of project decisions on productivity.

Using experience gained from characterizing productivity of real projects using the simplified model, we will use the complete workflow to perform simulation studies that generate plausible user scenarios generated based on the real user workflows we have recorded in our interviews, such as the one in Figure 1. Given a metric of productivity that makes sense for the individual user whose data was used as a template, we plan to simulate project time lines that explore the effects of plausible differences in resource requirements. Ultimately, we would like to classify a project's sensitivity to changes in the time spent in a task state due to outside influences.

Trends in actual use of HPC systems require a new approach to measuring their effectiveness, and we have proposed an approach that involves measurement and modeling of productivity factors that are identified through studying user behavior. We defined models for workflow in HPC and have proposed tools that use those models to help users analyze their productivity and cope with the challenges of using HPC for research.

References

[1] J. Brevik, D. Nurmi, and R. Wolski. Predicting bounds on queuing delay for batch-scheduled parallel machines. In *PPoPP '06: Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 110–118, New York, NY, USA, 2006. ACM Press.

[2] J. Carver, L. Hochstein, R. Kendall, T. Nakamura, M. Zelkowitz, V. Basili, and D. Post. Observations about software development for high end computing. *CTWatch Quarterly*, 2(4A), Nov. 2006.

[3] R. Harkness. personal communication. Jan. 2007.

[4] L. Hochstein, J. Carver, F. Shull, S. Asgari, V. Basili, J. K. Hollingsworth, and M. Zelkowitz. HPC programmer productivity: A case study of novice HPC programmers. In *Proceedings of ACM/IEEE Supercomputing Conference*, 2005.

[5] L. Hochstein, T. Nakamura, V. Basili, S. Asgari, M. Zelkowitz, J. K. Hollingsworth, F. Shull, J. Carver, M. Voelp, N. Zazworka, and P. Johnson. Experiments to understand HPC time to development. *CTWatch Quarterly*, 2(4A), Nov. 2006.

[6] Special issue on HPC productivity. *International Journal of High Performance Computing Applications*, 18(4), 2004.

[7] B. W. O'Shea, G. Bryan, J. Bordner, M. L. Norman, T. Abel, R. Harkness, and A. Kritsuk. Introducing Enzo, an AMR cosmology application. In T. Plewa, T. Linde, and V. G. Weirs, editors, *Adaptive Mesh Refinement - Theory and Applications*, Springer Lecture Notes in Computational Science and Engineering, 2004.

[8] D. Post and R. Kendall. Large-scale computational scientific and engineering project development and production workflows. *CTWatch Quarterly*, 2(4B), Nov. 2006.

[9] SDSC User Services. SDSC DataStar user guide. website. `www.sdsc.edu/us/resources/datastar/`.

[10] V. Shah, J. R. Gilbert, D. Mizell, and A. Funk. Modelling hpc workflows with timed markov models. *CTWatch Quarterly*, Oct. 2006.

[11] A. Snavely and J. Kepner. Is 99% utilization of a supercomputer a good thing? In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 37, New York, NY, USA, 2006. ACM Press. Birds-of-a-feather session.

[12] N. Wolter, M. O. McCracken, A. Snavely, L. Hochstein, T. Nakamura, and V. Basili. What's working in HPC: Investigating HPC user behavior and productivity. *CTWatch Quarterly*, 2(4A), November 2006. Supercomputing 2006 Print Issue.