Toward Accurate HPC Productivity Measurement[1]

Stuart Faulk
U. of Oregon

John Gustafson
Sun Microsystems

Philip Johnson
U. of Hawaii

Adam Porter
U. of Maryland

Walter Tichy
U. of Karlsruhe

Lawrence Votta
Sun Microsystems

Contact*: faulk@cs.uoregon.edu*

Abstract

One key to improving high-performance computing (HPC) productivity is finding better ways to measure it. We define productivity in terms of mission goals, i.e., greater productivity means that more science is accomplished with less cost and effort. Traditional software productivity metrics and computing benchmarks have proven inadequate for assessing or predicting such end-to-end productivity. In this paper we describe a new approach to measuring productivity in HPC applications that addresses both development time and execution time. Our goal is to develop a public repository of effective productivity benchmarks that anyone in the HPC community can apply to assess or predict productivity

1. Introduction

Our work focuses on developing an effective approach to characterizing and measuring software productivity in high performance computing applications. Removing or ameliorating productivity bottlenecks in next-generation high-performance computing systems is a key objective of DARPA's High-Productivity Computing Systems (HPCS) Program. Before we can hope to address software productivity problems we must agree on what we mean by "productivity" in HPCS applications, and how such productivity can be measured. Goals of our work include:

1) *Develop a common definition of HPCS productivity* that the HPCS developers, suppliers, and buyers (e.g., government agencies) can agree on. For our purposes, this means a definition that is consistent with the mission-level view that greater productivity means that more science is accomplished with less cost and effort.
2) *Develop effective measures of HPCS productivity* that encompass the overall development process—design time as well as execution time. In particular, we seek to develop measures that apply to a wide range of development environments and broadly across high-performance computing application domains (e.g., weather prediction, fluid dynamics, nuclear applications, etc.) to assess, compare, or predict productivity.
3) *Provide productivity measurement capabilities* to guide productivity improvement for both hardware and software developers. Effective, objective measurement provides the basis for systematic productivity improvement. We seek to provide common, public benchmarks and metrics to use in assessing and improving productivity.

Both HPCS developers and buyers have traditionally used standardized benchmarks (e.g., LINPACK) to guide development choices. System developers use benchmark results to guide platform development and subsequently demonstrate the speed of their machines. Buyers traditionally use such benchmarks to predict computation times and choose among competing platforms. However, the benchmarks and corresponding metrics employed to date have proven to be decreasingly effective predictors of end-to-end productivity as software development time has come to dominate execution speed as the primary productivity bottleneck. Traditional benchmarks focus almost entirely on hardware speed. Thus, they typically attempt to predict only execution-time productivity, ignoring development time. Further, they do not measure other properties of an application that matter to users: reliability, repeatability, portability, reusability, maintainability, etc.

To address these issues, we are creating a new type of standardized benchmark that 1) encompasses the breadth of design-time and execution time activities as well as 2) the productivity contributions of both functional and non-functional requirements. In addition to defining a canonical computation problem, these "productivity benchmarks" seek to characterize an end-to-end productivity problem by capturing the representative context of the computation. We will call such a multi-dimensional productivity benchmark a productivity benchmark suite (PBS).

[1] This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCH3039002.

2. Productivity Benchmarking Approach

A PBS comprises a canonical computation problem in the context of behavioral and developmental requirements representative of a particular high-performance computing domain. In addition to the functional and non-functional requirements, the productivity benchmark suite will provide targeted metrics and tools for measuring productivity in terms of overall costs and benefits across the development cycle. The goal is to create a set of benchmarking capabilities that, when applied, will exercise and measure not only the execution efficiency of a platform on a particular class of high-performance computing problems, but all the dimensions of development that contribute to the value of a solution.

Our long-range goal is to develop a public repository of empirically validated PBSs that are representative of the productivity challenges in each distinct high-performance computing domain. Platform developers or buyers can then apply these PBSs to assess and predict productivity of particular high-performance computing platforms on their domain interest.

Our approach to developing PBSs is based on the empirical derivation of canonical workflows [1] and purpose-based benchmarks [2]. Together with associated non-functional requirements, value function, and metrics, these sufficiently constrain a benchmark problem that different developers will be able to apply the benchmark and generate productivity measures that can be meaningfully compared. The key components are:

Canonical Workflows: Briefly, canonical workflows are used to characterize and constrain the process context of a productivity benchmark. A canonical workflow characterizes both the development process and the execution workflow associated with creating and using a high-performance computing application to meet an overall set of mission goals. It characterizes the process steps and work products associated with characteristic development paradigms in the high-performance computing community. An initial set of canonical workflows have been defined [1] and will be validated and refined through analysis of development efforts in different HPC domains.

Purpose-based benchmarks (PBB): PBBs are computational problems that accurately embody the design and execution time challenges of real applications in a domain. Unlike traditional benchmarks, PBBs are designed to exercise both the development process and the development platform in essentially the same manner (with reduced size) that real development problems do in a particular application domain. A detailed discussion of PBBs is given in [2].

Non-functional requirements: The benchmark will include representative execution time and developmental requirements with their associated metrics of completion and effectiveness. These include any requirements on the development process, administration, static-design, and run-time behavior characteristic of the application domain.

Characteristic value function: Associated with the requirements is a representative value function. The value function characterizes a value proposition (i.e., relative values of the different requirements) associated with applications in the domain interest.

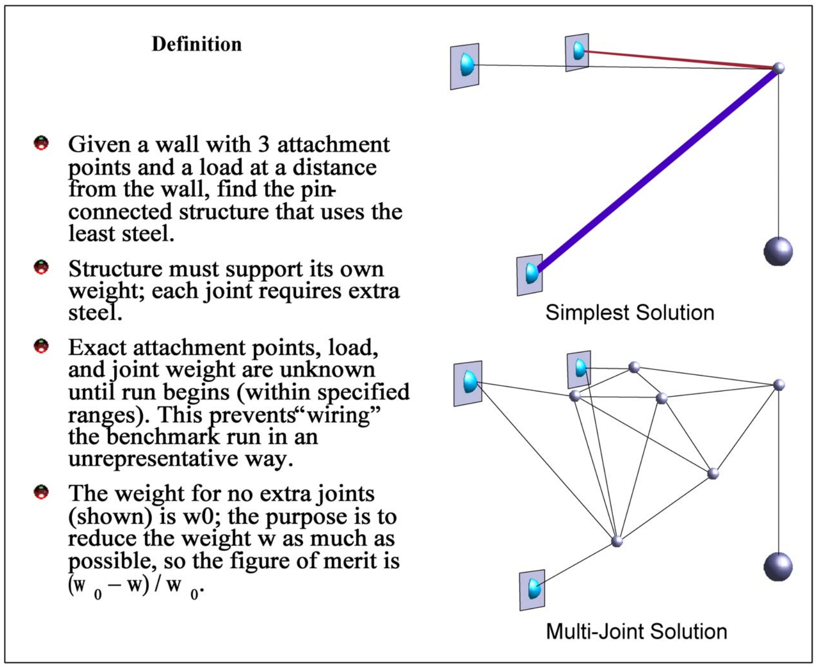
Productivity metrics and tools: A set of standardized metrics, algorithms and tools for measuring productivity associated with both development time and execution time activities and goals.

We will derive the properties and content of PBBs based on observation of real developers and from carefully controlled experiments. For example, we will obtain the application properties of interest and their relative values directly from developers in particular HPC domains by direct inquiry or by observation. A more complete description of our development approach will be published in [3]. The following gives a brief overview of key parts of our approach.

3. Purpose-Based Benchmarks

Currently, HPCS solutions are typically compared in terms of operations per second for certain well-known programs, such as those found in LINPACK or the STREAM benchmarks. We refer to these programs collectively as "activity-based benchmarks." As vendors and users of HPCS systems, we find that activity-based benchmarks are inadequate. In our experience user needs are never expressed solely in terms of processor speed. Rather, they are expressed in terms of simulations to be done, scientific questions to be answered, bridges to built - work to done. Processor rate affects these issues, but in the final analysis, it is not a measure of productivity.

In contrast to activity-based benchmarks, we are proposing a new class of benchmark called purpose-based benchmarks. A purpose-based benchmark (PBB) is a detailed description of a realistic, application-level goal, constraints on how that goal is met, and infrastructure for measuring how well the goal is met. Executing the benchmark means doing the end-to-end work needed to satisfy the specified need. By measuring this work across different HPCS solutions, we gain insight into the productivity offered by different solutions in different contexts. Because the PBBs must be executed on different HPCS solutions, they must be architecture- and programming language-independent so as not to unfairly preclude or disadvantage novel or nontraditional HPCS solutions. Specifically, the PBB includes:


Definition

- Given a wall with 3 attachment points and a load at a distance from the wall, find the pin-connected structure that uses the least steel.
- Structure must support its own weight; each joint requires extra steel.
- Exact attachment points, load, and joint weight are unknown until run begins (within specified ranges). This prevents"wiring" the benchmark run in an unrepresentative way.
- The weight for no extra joints (shown) is w0; the purpose is to reduce the weight w as much as possible, so the figure of merit is $(w_0 - w) / w_0$.

Figure 1: Truss Benchmark

- **Application statement:** This is a problem of direct interest to an HPCS user. For example, minimize the drag coefficient of a particular shape-constrained vehicle.
- **Scientific background:** To facilitate comparisons among different benchmark implementers, the PBB will contain a general description of the preferred scientific or mathematical solution approach. For instance, solve this problem using the following system of linear equations.
- **Solution constraints and requirements:** This is a statement of development and operational constraints, such as: the 24-hour weather forecast can take no longer than 3 hours to compute, or the software must meet certification standards for use in safety-critical environment A particular level of security might be required, as defined by codified government standards.

. Figure 1 illustrates a representative truss-design problem from mechanical engineering. In this problem the truss is to be attached at three points to a wall and must support a load at a given distance from the wall. The truss must support its own weight and each joint requires extra steel. The exact attachment points, load, and joint weights are unknown until run begins. The user's functional goal is to find the pin-connected structure that has the lowest weight.

To support their development and use, we are developing an open PBB repository. Our goal is to provide an open forum for developing, evaluating, and improving credible and repeatable PBBs. We are beginning to populate the repository by identifying specific HPCS application areas from which to draw candidate problems. These areas include: Nuclear Applications, Life Sciences, Mechanical Design, Crash Simulation, Fluid Dynamics, Weather and Climate Modeling, Signal/Image Processing, and Financial Modeling.

4. Productivity measurement

The economic definition of productivity is the **output** per **unit-of-work**. However, the mutable, intangible nature of both the processes and the products of software development make the outputs and units-of-work difficult to define or measure. The default has been to choose metrics that are relatively easy to measure, but that bear only a loose relationship to the value of what is produced (e.g., source lines of code [4] or function points [5])

Our goal is to address these issues by providing a framework for characterizing and measuring the perceived value of the output to system stakeholders. We define the output to include any properties of the system that consume work and have stakeholder value, including those that have no direct physical analog in the code (e.g., usability).

Initial work has shown that we can characterize the overall value of a solution in terms of the set of properties of interest to stakeholders typically defined as functional or non-functional requirements (e.g., security, availability, locality, portability, maintainability, and so on). We can capture this by representing the total relative value as a vector over the values of the properties of interest using the following framework. We associate with each property of interest i:

1) A metric of completion C_i
2) A relative value weight v_i

Briefly, the metric of completion C_i denotes the degree to which the realization of property i meets stakeholder requirements for that property. The relative value weight v_i represents the importance of the property i relative to the other properties of interest. The value of some set of properties i = 1 to n is given by the vector:

$$V_A = (v_1C_1, v_2C_2, \ldots, v_nC_n) \quad (1)$$

Assuming independence and that we can normalize each of the v_iC_i to a common metric (e.g., labor or cost), we can express the total value as the sum.

$$V_A = v_1C_1 + v_2C_2 + \ldots + v_nC_n \quad (2)$$

For example, we could calibrate each C_i such that $C_i = 1$ whenever property i meets its design goals, and v_i gives the relative importance of property i expressed as a percentage such that $\sum^{n}_{i=1}(v_i) = 100$ and $V_A = 100$ exactly when all the C_i are satisfied. Relative productivity is then given by the value produced divided by the work consumed to produce it:

$$P = V_A / W \tag{3}$$

Equivalently, we can say that the greater the value of V_A for a given amount of work, the higher the productivity. This corresponds to our intuitive view that greater productivity implies greater value per unit of work.

By design, our value function must be used in the context of a computing application that establishes the value space of interest. For productivity benchmarking, this context will be given by the PBS. The definition of the PBS will include the definitions of the properties of interest, corresponding metrics of completion, and representative value weights. Appropriate properties and values will be obtained from empirical studies of representative development efforts in the application area of the benchmark. An example illustrating the applicability of the model on a real HPC development is given in [3]. In [3] we also illustrate an approach to measuring non-functional attributes like maintainability.

5. Measuring development efforts

There are two major goals of empirical measurement in the context of our benchmarks:

1) Characterization: The first goal of process measurement is to better understand what actually happens during such development including identification of potential problems and bottlenecks in HPCS development, clarification of the similarities and differences between the various workflows for development, and the potential creation of predictive models for required resources and product quality.
2) Control: Once a baseline set of measures has been obtained measurement can begin to support project management activities. Model outputs can be used to help guide the new development.

We will measure HPCS development in both qualitative and quantitative ways. Our measurement techniques will include structured interviews, time and motion studies, and automated measurement. Each of these techniques has different strengths and weaknesses. By employing all of these techniques, we can ameliorate the weaknesses present in each form and improve the overall validity of the results.

Structured Interviews: In structured interviews, a researcher talks directly with members of the development team to learn more about the developer's view of the development process and its strengths and weaknesses. Structured interviews are useful for general characterization of a workflow, gaining insight into the kinds of quantitative measures that would be useful to collect, and collecting examples of process problems and solutions. Structured interviews will be used to conduct case studies of past HPC development efforts and to understand what kinds of computational problems and non-functional requirements characterize each application domain.

Time and Motion Studies: In time and motion studies the observer spends time "shadowing" one or more members of the development team, recording the times and tasks performed. Time and motion studies have the advantages of supporting fine-grained models of how developers spend their time, and surfacing issues in the development process that may not be perceived by the developers themselves. The data that is collected is thus of generally higher quality and fidelity than that collected by structured interviews (though far more labor intensive to collect) [6]. Thus time and motion studies will be used to corroborate the results of structured interviews, identify sources of systematic bias, identify additional issues, and provide detailed data on workflows and units of work.

Automated Measurement: A third form of measurement involves collection of data using the artifacts of development itself. For example, if development uses a source code control system, then the system logs can be analyzed to understand the patterns of developer interaction with the source files over time. Automated measurement has the advantages of collecting more objective measures of the process and products of development that are not filtered through the perceptions of developers. It is also low cost, but tends to be less complete than the other forms of measurement. For this effort, we are developing new approaches to automated measurement, based on [7], targeted to the specialized nature of HPCS development. Automated tools tailored to processes and products of specific domains will then be provided as part of each PBS to support productivity measurement of the benchmark's development.

6. Benchmark Development

We are in the process of developing an initial set of benchmarks, metrics and tools to validate our conceptual approach to productivity measurement for HPCS. Our development approach is iterative:

1) Identify a community of developers who will use the benchmark for productivity measurement or prediction,

2) Develop productivity measurement infrastructure appropriate for that community, e.g., define benchmarks, define workflows and corresponding functional- and non-functional requirements, create and install measurement instruments and analysis techniques,
3) Observe and measure developers as they execute the benchmark using the previously-defined infrastructure components, and
4) Analyze benchmark performance and evaluate and improve infrastructure.

We have initiated the process with two developer communities: one non-professional group consisting of graduate students at the University of Maryland, the other consisting of professional software developers working remotely from Russia. Both communities are beginning work on an initial version of the Truss Benchmark each following a different canonical workflow.

7. Summary

Ensuring that next-generation HPC platforms significantly improve real productivity in terms of the science accomplished will require new approaches to characterizing, measuring, and predicting productivity. Current productivity metrics and benchmarks fall short. Our goal is to establish, apply, and validate an effective approach to assessing and predicting productivity that spans both development and execution time. We seek to provide these capabilities in a form that supports platform buyers in choosing the best system and platform developers in providing technology that addresses real productivity problems.

Carefully controlled experiments will help us better understand precisely where developers spend their time and how different platform features might increase the efficiency those activities. From this we expect to develop detailed canonical workflows representative of different development environments. Detailed knowledge of problem characteristics, requirements, values, and workflows will be combined to develop tailored, productivity benchmarks for key HPCS domains. These benchmarks will provide not only a representative computation problem, but representative non-functional requirements that will exercise the entire development process across a value space appropriate to the domain and provide metrics and tools for measuring productivity throughout the development cycle.

Our long-range goal is to develop a public repository of well-validated PBSs that are representative of the productivity challenges in each distinct high-performance computing domain. Platform developers or buyers can apply these PBSs to assess and predict productivity of particular high-performance computing platforms on their domains of interest

8. References

[1] J. Kepner, “HPC Productivity: an Overarching View,” *International Journal of High Performance Computing and Applications: Special Issue on HPC Productivity* (ed. Kepner), vol. 18, no. 4, Winter 2004.
[2] J. Gustafson, “Purpose-Based Benchmarks,” *International Journal of High Performance Computing and Applications: Special Issue on HPC Productivity* (ed. Kepner), vol. 18, no. 4, Winter 2004
[3] S. Faulk, J. Gustafson, P. Johnson, A. Porter, W. Tischy, and L. Votta,. "Measuring HPCS Productivity”, *International Journal of High Performance Computing and Applications: Special Issue on HPC Productivity* (ed. Kepner), vol. 18, no. 4, Winter 2004.
[4] B. Boehm, *Software Engineering Economics*, Prentice Hall, 1981.
[5] A. Albrecht, and J. Gaffney, “Software function, source lines of code, and development effort prediction: a software science validation,” *IEEE Transactions on Software Engineering*, vol. 9, no. 6, pp. 639-648, 1983.
[6] D. Perry, N. Staudenmayer, and L. Votta, "Understanding and Improving Time Usage in Software Development," in *Trends in Software: Software Process*, Wolf and Fuggetta, eds., John Wiley & Sons, 1996.
[7] P. Johnson, H Kou, J. Agustin, C. Chan, C. Moore, J. Miglani. S. Zhen, and W. Doane, “Beyond the Personal Software Process: Metrics collection and analysis for the differently disciplined,” *Proceedings of the 2003 International Conference on Software Engineering*, Portland, Oregon, May, 2003.