

A System-wide Productivity Figure of Merit

Declan Murphy, Tom Nash, and Lawrence Votta, Jr.

A System-wide Productivity Figure of Merit

Declan Murphy, Tom Nash, and Lawrence Votta, Jr.

SMLI TR-2006-154

March 2006

Abstract:

The goal of this note is to combine productivity and performance benchmark measurement and subjective evaluations into a single system-wide figure of merit that could, for example, be used for budget justifications and procurements. With simplifying assumptions, which we believe do not compromise its usefulness, we have been able to decouple the variables that form the figure of merit so that each comes from an identifiable benchmarking, costing, or subjective evaluation activity. In this way, we expect that these different evaluations will be brought together into a single measure of productivity defined in a traditional way as dollar value (utility) output divided by dollar cost input. The main body of the text presents an operational-framework in a consciously heuristic style and describes how each component might be determined. This figure of merit is not really heuristic. We develop it with somewhat more rigor in Appendix I.



Sun Labs
16 Network Circle
Menlo Park, CA 94025

email addresses:

declan.murphy@sun.com
nash@fnal.gov
thomas.nash@sun.com
lawrence.votta@sun.com

© 2006 Sun Microsystems, Inc. All rights reserved. The SML Technical Report Series is published by Sun Microsystems Laboratories, of Sun Microsystems, Inc. Printed in U.S.A.

Unlimited copying without fee is permitted provided that the copies are not made nor distributed for direct commercial advantage, and credit to the source is given. Otherwise, no part of this work covered by copyright hereon may be reproduced in any form or by any means graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an information retrieval system, without the prior written permission of the copyright owner.

TRADEMARKS

Sun, Sun Microsystems, the Sun logo, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

For information regarding the SML Technical Report Series, contact Jeanie Treichel, Editor-in-Chief <jeanie.treichel@sun.com>. All technical reports are available online on our website, <http://research.sun.com/techrep/>.

A System-wide Productivity Figure of Merit

Declan Murphy, Thomas Nash, Lawrence Votta, Jr.

1. Introduction

Establishing a single reasonably objective and quantitative framework to compare competing high productivity computing systems has been difficult to accomplish. There are many reasons for this, not the least of which is the inevitable subjective component of the concept of productivity. Compounding the difficulty, there are many elements that make up productivity and these are weighted and interrelated differently in the wide range of contexts into which a computer developed in the High Productivity Computing Systems (HPCS) program may be placed.

This is not entirely a new phenomenon. Anyone who has driven a large scale computing budget request and procurement has had to address the problem of turning a set of preferences and new criteria into a budget justification and a procurement figure of merit that will pass muster with agency (and OMB) auditors. The process of creating such a procurement figure of merit helps to focus the mind and cut through the complexity of competing user demands and computing options.

Imagining that we are initiating a procurement where $\text{Productivity} = \text{Utility}/\text{Cost}$ will be the criteria, we have developed a total productivity figure of merit. This framework includes such system measurables as machine performance and reliability, developer productivity, and administration overhead and effectiveness of resource allocation. This is all in the context of information from the particular computing environment that may be proposing and procuring an HPCS computer. We note that this framework is applicable across the broad range of environments represented by HPCS mission partners and others with science and enterprise missions that are candidates for such systems.

The HPCS program includes research and development of all aspects of the problem. In principle, these efforts will be able to provide the input that can be brought together with information about the local computing environment into a single figure of merit to allow comparison between different designs and configuration options.

Each variable in our figure of merit is intended to come from a single source, either from one of the HPCS R&D areas or from a mission organization that may procure a system. While we identify the potential source of each variable, we do recognize that some of these numbers will not be easy to obtain, particularly those involving the impact of system design on human productivity. Nonetheless, we believe that, at the least, this framework will identify the individual metrics that these efforts should strive to measure. In the end, we will all have to admit that some combination of measurements, informed guesses, and subjective evaluations will be needed to arrive at a figure of merit number.

We provide a spreadsheet that can be used in most environments to enter the input variables and compute the results. Our intention is to start entering inputs as guesses at a very early stage and continuously improve them as more understanding and measurements are obtained. The spreadsheet includes a place to indicate the quality of each entry (from “wild guess” to “post-mortem measurement”) so that a quick look at the spreadsheet may indicate how accurate the figure of merit has become.

We also recognize that there is coupling between some of the variables we treat as independent. For example, a user’s productivity is impacted by the way jobs are allocated. To deal with this, we suggest assuming an environment in which a particular variable is determined. This means that values for all the other variables, their “operating point”, must be specified for each measurement of a variable.¹ For measurable variables, these operating points come from measurements and studies. One could iterate to a final answer, but we argue that this is unnecessary because the effect of any such coupling, with reasonable operating point guesses, is far smaller than the precision (such as it might be) of any of the measured variables involved.

Just like the blind men and the elephant, each one of these variables corresponds to a different perspective on the effort to optimize productivity. Taken together, they give a good overall measure of the productivity of different versions of our planned computing system.

Not surprisingly, our figure of merit has much in common with ideas expressed in earlier HPCS work.² However, we extend beyond this in the following ways:

In a well-balanced HPCS, significant costs will be incurred for other resources than just the CPU cycles that dominate thinking in the commodity cluster architectures. In particular, memory and bandwidth resources will have cost as much or more than CPU, and efficient programs and job allocation will have to optimize use of memory and bandwidth resources as much as CPU. Our framework allows for the inclusion of any set of significantly costly resources.

A single job may be highly optimized, and those in the project it comes from will inevitably believe its utility (or value) approaches infinity. However, a computer center must optimize the total workload, given its organization’s evaluation of relative priority (utility or value) for each project and job. The overall utility of the total output of the computer depends on the degree to which the allocation of system resources reflects the institution’s priorities and determination of value (or

¹ This concept is described a bit more rigorously in Appendix I.

² In particular, the important aspects that address user work productivity at the job-level reduce identically to Sterling’s w-model of productivity. Thomas Sterling, Productivity Metrics and Models for High Performance Computing, *J. High Performance Computing and Applications* 18(4):433-440 (2004).

utility). Further, the productivity of the administration staff depends on a system's administration environment and tools, and on its stability.³

We have emphasized an approach that, like the blind men, focuses on what can be measured at defined parts of the beast and builds to a picture of the whole productivity equation for a system in a particular environment. This assumes that our elephant is reasonably smooth and predictable between those points we can feel with our measurements. One could also come to the same formula by looking at the elephant as if we were the committee that designed it, with a completely parameterized model. From this complete model, by calling out assumptions and simplifications, we can reduce to our productivity figure of merit where decoupled measurements at the job level and system level, in the context of environment specific work flow and valuation, give us a result that is a reasonable approximation of the lifetime productivity. This approach is described in Appendix I.

In Appendix II we apply a spreadsheet we have prepared to compute the figure of merit for one historical example where the environment and its workflows are pretty well understood. We hope to provide examples for other examples of HPCS installations in the future. We expect this to demonstrate the broad applicability of our figure of merit. More ambitiously, one may be able to extract a set of parameters that broadly characterize the interests of different environments as manifested in how they would shape their own procurement figure of merit. Examples of these parameters are the degree to which capability or capacity is considered important, or job mix average ratios like CPU to IO or CPU to inter-processor bandwidth.

³ These extensions may be seen as a response to Kepner's remark "that all necessary ensemble averaging over users, applications and systems can be performed without loss of generality." In fact, we show in Appendix I that our figure of merit follows from ensemble averaging in this way, starting essentially at the synthesis expression in Section 2.2 of Kepner's paper. Jeremy Kepner, HPC Productivity Model Synthesis, *J. High Performance Computing and Applications* 18(4):505-523 (2004).

2. The Productivity Figure of Merit

We start with

$$Productivity = Utility/Cost$$

We expand the utility into system level and job level components

$$P = \frac{U_{sys} E_{util} E_{adm} A_{sys} E_{job} R}{C} \quad (1)$$

As a convention we use the letters U , E , A , R , C to denote variables of utility, efficiency, availability, resources, and cost, respectively. The subscripts indicate the variables that address system level (including administrative and utility) and job level factors.

We recognize that some aspects of the system level efficiency will never be amenable to measurement and will always require subjective evaluation. Only subjective evaluation processes can address the first two variables in the utility numerator. In principle one can measure the last three variables and the HPCS research program is addressing such measurements.

We have been emphasizing that this is to be a figure of merit, including estimates and evaluations of what we expect to be the productivity output of an installation. For clarity, in explaining this formulation, we will start by talking as if we know what happened over the lifetime, as if we are doing a post-mortem. We will mention in passing, in this section and Section 3, how the components relate to ongoing work on productivity estimators. We will return to discuss these estimators in more detail in Section 4.

The goal of those optimizing utility at the job level is to maximize resources they can effectively apply to their problem. This will enable them to bring their project to a successful conclusion with higher utility (larger scale or finer granularity solutions or higher throughput of data intensive problems) or more rapidly (allowing more problems of similar utility to be accomplished within the lifetime of the resources). It is “not in their job description” to address the relative utility of their problem compared to others (though they may be inclined to do so). So, we consider utility at the job level, U_{job} , to be just the cost (\$) of the resources that they have *effectively* used, and the job level efficiency $E_{job} = U_{job}/C_{sys}$, with C_{sys} the total lifetime system cost. As defined below, U_{job} and E_{job} are averaged over all jobs.

The government, or the owners or stockholders, establish how activities are to be valued by the institution management. Management valuation appears in the largely subjective U_{sys} variable, which includes, for example, what constitutes “success” for a computing activity. They may well value a successfully completed activity higher than the cost of the resources used. (It would be unusual for them not to do so at budget proposal time.) U_{sys} assumes, as managers might often wish, that all resources are available and fully

allocated at all times during the life of their expensive system.⁴ We define it as a multiplier to the peak system resources in CPU units. System utilization efficiency is included in E_{adm} and E_{util} as described below.

Once a system is delivered, the system administrators (and vendor) strive to meet management expectations for availability, A_{sys} and system level resource utilization efficiency (including allocation), E_{adm} .

More detailed descriptions of the individual variables follow. These are further expanded in the next section. We indicate dimensional units in brackets.

P , the productivity, is dimensionless, as required by the common economic interpretation of the productivity concept we use, output/input, [\$/[\$].

C is total cost of ownership, including all costs for developing software and running it on the computer, over a defined lifetime of the system (T). This will be expanded into components below. [\$/]

T is the lifetime of the system as defined in the typical budget approval process. It does not show up explicitly in the top-level equation, but it is important to the definitions of the variables as well as for the measurements. It will depend on considerations specific to each environment, including, for example, whether the procurement and justification (and cost) involve continuing upgrades. Those responsible for budget submissions at the proposing institution are the source of this variable. [yrs]

E_{job} is the ratio of U_{job} to the total system cost C_{sys} . U_{job} is the total productively used resources over the lifetime by all individual jobs in all project activities, normalized to the assumption of 100% availability and 100% resource allocation. To a large extent, this efficiency measures how well programs use parallel resources. We include other costly resources (e.g., memory, bandwidth, I/O) besides CPU. In order not to favor one class of resources over another, we weight the resources by their relative costs.⁵ We say “productively used” so that we only include resources that would be consumed in direct support of maximizing utility for the specified problem. Estimators for E_{job} and for \bar{t}_{proj} , the corresponding average total project personnel time (including development and production) per project, are what Job-level Productivity Benchmarks should aim to measure for different environments and systems. All of these somewhat difficult to define issues are detailed in the next section. [Dimensionless]

⁴ However, a system approaching 100% allocation will have long queues that would have a deleterious effect on the total utility coming out of the computer. Effects like long queues should be included as part of the environment conditions (the “operating point”) in which job level productivity measurements of E_{job} and \bar{t}_{proj} , defined below, are made.

⁵ For example c_{CPU} is that fraction of the total lifetime cost, C_{sys} , that is attributable to CPU, in \$/ops. For CPU, U_{job} can be understood (in Sterling’s notation) as $S \times T \times c_{CPU}/C_{sys}$ summed over all jobs during the lifetime T , with S the attained performance [ops/sec].

A_{sys} is the availability, the fraction of total resource time available to the jobs making up R_{job} . This accounts for all planned and unplanned downtimes, but does not include job restarts due to failures, which is included in E_{sys} below. Note this is not fraction of system time, but fraction of resource time, so that it accounts for portions of the large system being available when other portions are down. This parameter is based on vendor estimates of MTTF, maintenance requirements, and other RAS considerations. [Dimensionless]

The product

$$E_{sys} \equiv E_{util} E_{adm}$$

is the effectiveness of the workload resource allocation at meeting the institution's priorities. System architecture, software, and administrative tools can make a big difference to this important metric. This is where utility (or value) enters the picture and we go beyond just looking at job-level optimization. We assume that the institution will have a process for establishing the priority or the utility or the value of the individual projects and jobs. E_{sys} is the ratio of the actual total value of the jobs run over the lifetime to the optimal total that would come from maximally efficient management of a reference HPCS platform,⁶ for a given job mix, a given installed system configuration, and a given time pattern of resource downtime. As shown in Appendix I, we allow for time dependent values, since allocation efficiency that completes a job or project long after it is needed has little or no value. We recognize explicitly (see further discussion in Appendix I) that not all the factors that go into E_{sys} are amenable, even in principle, to before or after the fact quantitative measurement. Some will need to be evaluated subjectively. This is why we have written E_{sys} as a product with E_{adm} as the measurable part and E_{util} the part requiring subjective evaluation. E_{adm} is the estimator that, along with the cost of administration, is what System-level and Administration Productivity Benchmarks should aim to measure, for different systems, configurations, and user environments. E_{util} represents other considerations that result in reduced efficiency at delivering utility. Both are discussed further below. [Dimensionless]

U_{sys} is the summed utility (or value) of all projects or activities, per unit of total available resources, R , that would complete successfully during the lifetime T if the system administrators given the systems tools and capabilities of the HPCS Reference Platform were able to attain the optimum level of resource utilization and the system was perfectly available. The values may be assigned by whatever process the institution uses to prioritize its projects and justify its budget proposals. This variable only includes the local institution's evaluation of the value of the projects it intends for the installation. In some environments, it may

⁶ Since E_{sys} is intended to compare the system level resource utilization efficiency for different system designs, vendors, and configurations, it needs to be a ratio to some absolute standard. A reasonable definition that we will use is the "ideal" one, which delivers the sum of the values of all the projects.

be ignored and set to a constant such as C_{sys} . We convert all resource units to CPU operations, weighting by their relative costs. [\$/ops]

3. Expansion of Key Terms

Now, let's expand these high level terms:

A. Cost

$$C = C_{sys}^{initial} + c_{sys}^{annual}T + c_{adm}n_{adm}T + Nc_{proj}\bar{t}_{proj} \quad (2)$$

Here,

$$C_{sys} = C_{sys}^{initial} + c_{sys}^{annual}T \quad (3)$$

is the total life cycle cost of the system. [\\$]

It includes the following costs:

One Time Costs

$$C_{sys}^{initial} \quad [\$]$$

Initial hardware

Initial software (system and purchased application software)

Non-standard or specialized facility equipment (e.g., water cooling)

Facility preparation (electrical and other building modification)

Installation

Recurring Costs

$$c_{sys}^{annual} \quad [$/yr]$$

Hardware maintenance (repair and preventive)

Software maintenance (repair and preventive)

Facility maintenance

Hardware and software upgrades (as required by contracted goals)

Electrical costs (for system and cooling)⁷

Space and standard facility equipment (GSA or other rate per sq-ft per year)

$$C_{adm} = c_{adm}n_{adm}T = c_{adm}(n_{adm}^{base} + n_{adm}^{productivity})T$$

is the total lifetime administration costs.

n_{adm} is average number of personnel at the local institution supporting system maintenance and management, user assistance, and system workload

optimization. c_{adm} is the cost per person-year [\$/person-year] of this support.

These costs are in large measure affected by both the workload environment and

⁷ Note that electrical running costs may be dependent upon the job mix.

the system and its administration tools. n_{adm} is the personnel load that System-level and Administration Productivity Benchmarks can aim to measure along with E_{adm} . n_{adm} can be broken into two parts, one that is a baseline cost in the host environment for managing a system of this magnitude, n_{adm}^{base} , and a part that corresponds to the effort to optimize utilization and other activities where the administrators' productivity can be affected by administrative tools, $n_{adm}^{productivity}$. [Persons]

N is the total number, over the lifetime T , of unique program activities (projects with the goal of creating an appropriately efficient program and a correct and useful result). The definition of program activities and an estimate of N is very dependent on the local environment and should come from the institution for which systems are being evaluated. [Dimensionless]

\bar{t}_{proj} is the average total project personnel time for one of the unique program activities counted by N . This includes, in addition to the software development effort, personnel costs for production related effort (job submission, bookkeeping, etc.) This, along with E_{job} is what Job-level Productivity Benchmarks should aim to measure for different environments and systems. [Person-years]

c_{proj} is the average cost per person-year of all project-related personnel. [\$/Person-year]

B. Productively Used Resources at the Job Level

As noted before we define the job level utility as the cost of the resources productively used

$$U_{job} = c_{CPU} E_{CPU} R_{CPU} + c_{mem} E_{mem} R_{mem} + c_{BW} E_{BW} R_{BW} + c_{IO} E_{IO} R_{IO} \quad (4)$$

and the job level efficiency as

$$E_{job} = U_{job}/C_{sys} \quad (5)$$

Here, the subscripts refer to the resource types, CPU, memory, inter-processor bandwidth, and I/O bandwidth resources, respectively. This can obviously be generalized to include other resources with significant costs.

The c_r are the total costs attributable to each resource per unit of that resource. The total life cost is C_{sys} described above,

$$C_{sys} = c_{CPU} R_{CPU} + c_{mem} R_{mem} + c_{BW} R_{BW} + c_{IO} R_{IO} \quad (6)$$

The R_r are the total lifetime resources of type r used by all of the N project activities.⁸ The resources are assumed for the purposes of this job-level variable to have been 100% allocated and the time to be 100% available for one of the N activities.⁹ Note that costs only come into Eq.(5) to provide a relative weight for the different resources.

The R_r are based on performance measurements and a choice of configuration options. Remember that the R_r include the lifetime, so that the units for CPU, memory, bandwidth, and IO are [ops] (not ops/sec), [byte-years], [bytes], and [bytes], respectively. As we noted in the previous section, we weight the resources by their relative costs, so U_{job} , is just the cost (\$) of the resources that the project teams could have used if they were perfectly efficient. The c_r provide the conversion from resource units to utility as cost in \$ [\$/ops], [\$/byte-years], [\$/bytes], [\$/bytes].

The E_r are the fraction of the total of each resource productively utilized on average by all the jobs in the N project activities. The variables E_r are efficiency estimators that Job-level Productivity Benchmark efforts can aim to measure (for specified workflows and job mixes) along with the average effort per activity \bar{t}_{proj} .

By “productively,” we mean that resources used only to support parallelization are not counted, and that the single processor algorithm being used is not wasteful of resources. This can either be a protocol rule in the benchmarking or the benchmarks can down-rate the utilization fractions E_r for resource usage that is not in direct support of the task or algorithms that are less than optimum.¹⁰

The E_r are [dimensionless].

⁸ Sums over the jobs in a workflow and the projects at a computing center are shown explicitly in Appendix I.

⁹ This assumption includes a simplification that the different resources are all available and allocated in the same proportions. It normalizes U_{job} so that for perfect job level utility optimization $U_{job} = C_{sys}$.

¹⁰ It may be best to define “productive use of resources” with examples. If we replicate memory just to reduce latency by keeping data close to the CPUs in a parallelized problem, this use of the memory resource does not count as productive, although the CPU utilization may increase and be counted as a result. But if the parallelization allows us to expand the problem space (larger matrices, reduced granularity, etc.), or increase the throughput in data intensive situations, and increase the utility of a successful result, this memory use does count. Similarly, bandwidth to access non-local memory only counts if utility is increased as a result, and not just if it is replicated and “indirect” support of parallelization. The excess usage of a resource wasteful algorithm (for example, one that burns resources unnecessarily on a single processor) or of wasteful localization of a scaled up problem (for example, data placed so unnecessary bandwidth is used) does not count as productive use. In the end, to paraphrase Justice Potter Stewart in the context of something else difficult to define, we know “productive resource utilization” when we see it.

C. System Level Efficiency

E_{adm} , the measurable part of system efficiency, may be understood as the effectiveness of the administrative staff in allocating resources efficiently given the tools and the real environment of their system and the time that they have available, as included in the cost c_{adm} . An estimator for this traditional measure of system utilization, E_{adm} , is what System-level and Administration Productivity Benchmarks can aim to measure, as discussed below.

E_{util} is the subjective component of the figure of merit to allow for evaluation of issues, which might be attributed to system hardware or software, such as project failures or delays and the accessibility of the computing system environment to staff with different levels of computing skills. In general, it includes system or configuration factors that impact the effectiveness of programming teams at accomplishing their goals. This is where utility vs. time considerations may be included, as discussed in Appendix I.

E_{adm} and E_{util} are [dimensionless].

4. Productivity Benchmarks and Operating Points

We could get an accurate figure of merit as part of a post-mortem - after the life cycle is over. At that point, we could have access to real experience. But that's not what we want. We need to predict and estimate in advance. So where do our productivity estimators come from? We have been assuming there are two classes of productivity benchmarking activities: 1) at the job-level, measuring development and production activities; and, 2) at the system-level, measuring administration activities and the effect of the differences in system designs and configuration options on administration overhead and system-level resource utilization.

Development, job-level, productivity benchmarks would aim to measure - for the problem mix of a specific environment - the development time required to attain different levels of productive resource utilization. The simplest example is how much programming time it takes to attain various levels of speedup. Curves of productive resource utilization vs. development and other project personnel time will increase, probably like step functions and will usually present an obvious point of diminishing returns. This can be taken as the "operating point." Averaged over the work flow and job mix, it provides an estimator of \bar{t}_{proj} and E_{CPU} , E_{mem} , E_{BW} , and E_{IO} .

Similarly, administration, system-level, benchmarks can aim to measure E_{adm} . For aspects involving effective scheduling, this could be accomplished, specific to a given environment and system configuration, by creating a simulated scheduling environment and allowing working administrators to attempt to optimize the allocation of a list of prioritized (value assigned) jobs for some simulated time period. An ideal allocation would be defined for each environment as attaining the full value of each project

including time considerations. The ratio of the measured to the ideal would give an estimator for this aspect of E_{sys} . Just as for the development benchmarks, this can be treated as a curve where better allocation, more efficient use of resources, and a general increase in output value, uses more administrator time, and a reasonable operating point of diminishing returns is selected. The cost of the administrator time at this operating point would be included as all or part of c_{adm} (depending on how complete a benchmark simulation is used).

5. Conclusion, Examples, and a Spreadsheet to Learn From

The productivity figure of merit we have described here is really much simpler than it may appear. It is no more than a way of getting to a single number that combines what we know or can guess about a system configured for a particular environment into something that approximates a measure of total productivity. It can be used in HPCS design comparisons, subsequent budget justifications, and ultimately, we hope, in procurements.

Many of the numbers needed as input are traditional cost and performance variables. The hard parts, of course, are the benchmarks needed to measure the productivity of humans when confronted with these new systems. We see these productivity benchmarks as simply measuring curves of efficiency of job resource utilization, or system utility optimization, vs. the cost in time of the human effort. These curves should clearly indicate an obvious point of diminishing returns, an operating point.

We recognize that we have made use of words like *simply*, *clearly*, *obviously*, and this may be unfair. We know that getting a number that approaches being a true measure of productivity for a given system is going to be difficult. We need goals for the productivity benchmarking efforts to aim at, and we think this framework provides them.

Appendices II and III are examples of figure of merit evaluations for a fictitious laboratory environment that might bear a remote resemblance to a well-studied mission partner history. These examples are based on a spreadsheet template that we have prepared. The template itself is available from the authors and the HPCS website.

The two examples correspond to the same environment and set of projects supported by two different systems. The first (“Lavender”) resembles a traditional MPI cluster-like high performance computing configuration circa 2003 in an ASCI-like environment. The second (“Orange”) is more like what we hope to attain with the significant performance and productivity advances of a full scale HPCS system.

The actual numbers in these two examples should be considered to be very preliminary guesses. They have been included only to demonstrate that the model and spreadsheet show appropriate behavior for varying inputs and as an example of what the inputs are about. In fact, the key goal of this paper is to identify the variables that need to be determined by historical studies and performance and productivity benchmarking, so that system-wide productivity comparisons become possible.

The first 6 pages of the spreadsheet are entry sheets to be completed by the different entities that may be responsible for the information: Vendor Cost, Host Environment and Values, Performance, Job-level Productivity, Administration Productivity, and Subjective Productivity Evaluation. The last page of each appendix summarizes the calculated results.

Note that in addition to providing the list of project activities and their values, the host environment must also define the list of job-level benchmarks to be used for both the performance and productivity measurements. This list may be specific to each of the projects or common to the whole environment.

As we noted in the Introduction, we see the process of obtaining an informative figure of merit as being incremental. One may start by entering guesses, policies, and goals, and then progress through preliminary measurements and even post-mortem analysis. For each entry, the spreadsheet has a quality descriptor which may be selected from a list, presently including: *canonical, policy, wild guess, informed guess, measured 100%, measured 30%, measured 10%, measured 1%, post mortem.*

We strongly recommend playing with the spreadsheet, changing performance, productivity, and cost variables and seeing their effect on the figure of merit and its components on the last page of the spreadsheet. In this way, one can learn quickly in a particular environment what matters and what doesn't for the life-cycle productivity, our figure of merit. Even with the simplifications,¹¹ and possible biases,¹² built into this approach, we believe it goes furthest towards allowing a real understanding of how best to reach the goal of maximizing overall productivity.

Acknowledgements

We acknowledge with pleasure the important contributions made to the development of a figure of merit, particularly in the critical early stage when the key ideas were being understood, by the following: Susan Squires, Jan Strachovsky, Michael Van De Vanter, and Robertus Van Der Wijngaart.

¹¹ In the likely event that host environments wish to specify time-dependent values for projects, these may be parameterized functions or tables. In either case, the spreadsheet will have to be revised to accommodate time-dependent values and system level efficiencies since the present examples do not take time into account.

¹² Value judgments may be implicit in some of our simplifications. If these are not consistent with those of a particular environment, they can obviously be adjusted using the basic framework of this approach and the spreadsheet. What is important is to attempt to include as many cost, performance, and productivity components as possible into the best possible single productivity figure of merit.

Appendix I

Extending the productivity model synthesis at Section 2.2 of Kepner's paper¹³ to all aspects of productivity over an HPCS installation lifetime, we can write a reasonably complete parameterization as

$$P = \frac{\text{Utility}}{\text{Cost}} = \frac{\sum_{p=1}^N \sum_{j=1}^{J_p} \sum_r \text{resources} U_{pjr} E_{pjr}^{\text{util}}(R_r^{\text{delivered}}(t')) \int_0^T dt E_{pjr}^{\text{adm}}(t) E_{pjr}^{\text{job}}(t) A_r(t) r_r(t)}{C_{\text{sys}}^{\text{initial}} + \int_0^T dt \left[c_{\text{sys}}^{\text{annual}}(t) + c_{\text{adm}}(t) + c_{\text{proj}}(t) \sum_{p=1}^N n_p(t) \right]} \quad (\text{I-1})$$

The parameters are generally as defined in the body of the paper, but here shown with possible dependence on the three subscripts p, j, r and the time. They are described further below.

We sum over jobs j in project activities p . As discussed in the paper, project activities will be defined for specific mission environments. The jobs within a project include all development, testing, or production phase submissions associated with (chargeable to) the project. We also sum over resource types r . The resources are typically $r = \{CPU, \text{memory}, \text{bandwidth}, IO\}$, but should include any resource types that contribute significantly to the system cost.

U_{pjr} is the institution assigned value per unit of productively allocated and used resources of type r to job j in project p . $E_{pjr}^{\text{util}}(R_r^{\text{delivered}}(t))$ is the fraction of that value delivered during the lifetime given that the delivered resources as a function of time is

$$R_r^{\text{delivered}}(t) = \int_0^t dt' E_{pjr}^{\text{adm}}(t') E_{pjr}^{\text{job}}(t') A_r(t') r_r(t') \quad (\text{I-2})$$

E_{pjr}^{util} represents the familiar utility vs. time curves described by Snir and included in Kepner's synthesis.

The $r_r(t)$ are the system's maximum resource capabilities per unit time [ops/sec], [bytes], [byte/sec], [bytes/sec], so that the $R_r = \int_0^T r_r(t) dt$, as used in Eqs. (4) and (6) of the main text, are the total potentially available system resources. A_r is the availability of resource r , and E_{pjr}^{adm} and E_{pjr}^{job} are the system and job level efficiencies of applying the resource to job j in project p .

In the denominator, $n_p(t)$ is the number of people working on project p at time t .

¹³ Jeremy Kepner, HPC Productivity Model Synthesis, *ibid*.

Our goal is to reduce this complex equation to one that depends only on a set of independently measurable parameters and environment variables. We start by integrating averages of all the project and job level efficiencies and the resource availability in the numerator and the costs in the denominator:

$$P = \frac{\sum_{p=1}^N \sum_{j=1}^{J_p} \sum_r^{resources} U_{pjr} \bar{E}_{pjr}^{util} \bar{E}_{pjr}^{adm} \bar{E}_{pjr}^{job} \bar{A}_r R_r}{C_{sys}^{initial} + c_{sys}^{annual} T + c_{adm} T + N c_{proj} \bar{t}_{proj}} \quad (I-3)$$

Summing and averaging over the resources, projects and jobs, we arrive at Equation (1)

$$P = \frac{U_{sys} E_{util} E_{adm} A_{sys} E_{job} R}{C}$$

Here we have defined the variables used in the main text as

$$R = \sum_r \frac{C_r}{C_{CPU}} R_r \quad [\text{ops}]$$

$$U_{sys} \equiv \sum_{p=1}^N \sum_{j=1}^{J_p} \sum_r^{resources} U_{pjr} \frac{C_{CPU}}{C_r} \quad [\$/\text{ops}]$$

$$E_{util} \equiv \overline{\sum_{p=1}^N \sum_{j=1}^{J_p} \sum_r^{resources} \bar{E}_{pjr}^{util}} \quad [\text{dimensionless}]$$

$$E_{adm} \equiv \overline{\sum_{p=1}^N \sum_{j=1}^{J_p} \sum_r^{resources} \bar{E}_{pjr}^{adm}} \quad [\text{dimensionless}]$$

$$A_{sys} \equiv \overline{\sum_r \bar{A}_r} \quad [\text{dimensionless}]$$

$$E_{job} \equiv \overline{\sum_{p=1}^N \sum_{j=1}^{J_p} \sum_r^{resources} \bar{E}_{pjr}^{job}} \quad [\text{dimensionless}]$$

$$C = C_{sys}^{initial} + c_{sys}^{annual} T + c_{adm} T + N c_{proj} \bar{t}_{proj} \quad [\$]$$

Note that, as we emphasized in the main text, in each case the efficiency sums imply averages over the “operating conditions” of the other efficiency and availability factors

(indicated by the bars over the top). Productivity benchmarks of E_{job} and E_{sys} and the subjective evaluations must take these operating conditions into account.

Vendor Cost Entries

		<i>Entry Quality</i>	<i>Units</i>	
$C_{sys}^{initial}$	150.00	<i>informed guess</i>	[M\$]	Initial system cost including facility setup
C_{sys}^{annual}	20.00	<i>wild guess</i>	[M\$/yr]	Annual recurring costs, hardware, software & facility maintenance and planned upgrades, electrical and space
$\frac{C_r}{C_{sys}}$				Fractional resource costs, initial & recurring
<i>CPU</i>	0.33	<i>wild guess</i>	[dimensionless]	r=1
<i>memory</i>	0.33	<i>wild guess</i>	[dimensionless]	r=2
<i>bandwidth</i>	0.33	<i>wild guess</i>	[dimensionless]	r=3
<i>other</i>	0.00	<i>canonical</i>	[dimensionless]	r=4

Host Environment Entries

		<i>Entry Quality</i>	<i>Units</i>	
T	7	<i>policy</i>	[yrs]	System Lifetime
N	9	<i>policy</i>	[dimensionless]	Number of unique program activities (projects)
n_{adm}^{base}	5	<i>wild guess</i>	[persons]	Base administrative personnel count
c_{adm}	225	<i>informed guess</i>	[K\$/person-yr]	Loaded average admin personnel cost
c_{proj}	300	<i>informed guess</i>	[K\$/person-yr]	Loaded average project personnel cost
<i>Projects</i>	U_p			p <i>Benchmarks</i>
<i>Egret</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	1 HPL Stream Ptrans
<i>Jabiru</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	2 HPL Stream Ptrans
<i>Hawk</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	3 HPL Stream Ptrans
<i>Kite</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	4 HPL Stream Ptrans
<i>Finch</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	5 HPL Stream Ptrans
<i>Gull</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	6 HPL Stream Ptrans
<i>Raven</i>	0.10	<i>policy</i>	[M\$/TFlops/s]	7 SSCA2
<i>Condor</i>	0.05	<i>policy</i>	[M\$/TFlops/s]	8 SSCA2
<i>Crow</i>	0.01	<i>policy</i>	[M\$/TFlops/s]	9 SSCA2
				10
				11
				12
				13
				14
				15
				16
				17
				18
				19

Note: Where time dependent values and efficiencies are required, this spreadsheet is not complete as shown. In such environments, the time dependent integrals may be evaluated numerically or the time dependence may be parameterized for inclusion in a revised spreadsheet formulation.

Performance Entries

		<i>Entry Quality</i>	<i>Units</i>	
A_{sys}	0.70	<i>informed guess</i>	[dimensionless]	RAS system availability averaged over resource types.
r_r				Peak resource capacity
<i>CPU</i>	20	<i>informed guess</i>	[TFlops/sec]	r=1
<i>memory</i>	30	<i>canonical</i>	[Tbytes]	r=2
<i>bandwidth</i>	500	<i>canonical</i>	[GUPS]	r=3
<i>other</i>	0	<i>canonical</i>		

Job Productivity Entries

		<i>Entry Quality</i>	<i>Units</i>	
\bar{t}_{proj}	75	<i>measured 30%</i>	[person-yrs]	average total project personnel time per project
E_r				Average fraction of peak performance attained
<i>CPU</i>	0.05	<i>informed guess</i>	[dimensionless]	r=1
<i>memory</i>	0.30	<i>wild guess</i>	[dimensionless]	r=2
<i>bandwidth</i>	0.10	<i>wild guess</i>	[dimensionless]	r=3
<i>other</i>	0.00		[dimensionless]	

System-level Productivity Entries

		<i>Entry Quality</i>	<i>Units</i>	
$n_{adm}^{productivity}$	3	wild guess	[persons]	Admin personnel affected by admin productivity tools
E_{pr}^{adm}				Administrative productivity allocation efficiency
CPU	0.40	wild guess	[dimensionless]	r=1 p=1 Egret
memory	0.40	wild guess	[dimensionless]	r=2
bandwidth	0.40	wild guess	[dimensionless]	r=3
CPU	0.40	wild guess	[dimensionless]	r=1 p=2 Jabiru
memory	0.40	wild guess	[dimensionless]	r=2
bandwidth	0.40	wild guess	[dimensionless]	r=3
CPU	0.40	wild guess	[dimensionless]	r=1 p=3 Hawk
memory	0.40	wild guess	[dimensionless]	r=2
bandwidth	0.40	wild guess	[dimensionless]	r=3
CPU	0.40	wild guess	[dimensionless]	r=1 p=4 Kite
memory	0.40	wild guess	[dimensionless]	r=2
bandwidth	0.40	wild guess	[dimensionless]	r=3
CPU	0.40	wild guess	[dimensionless]	r=1 p=5 Finch
memory	0.40	wild guess	[dimensionless]	r=2
bandwidth	0.40	wild guess	[dimensionless]	r=3
CPU	0.40	wild guess	[dimensionless]	r=1 p=6 Gull
memory	0.40	wild guess	[dimensionless]	r=2
bandwidth	0.40	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=7 Raven
memory	0.60	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=8 Condor
memory	0.60	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=9 Crow
memory	0.60	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3

Note: Where time dependent values and efficiencies are required, this spreadsheet is not complete as shown. In such environments, the time dependent integrals may be evaluated numerically, or the time dependence may be parameterized, for inclusion in a revised spreadsheet formulation.

Subjective Productivity Entries

		<i>Entry Quality</i>	<i>Units</i>	
E_{util}	0.30	<i>post mortem</i>	[dimensionless]	Subjective system level productivity efficiency factors

Summary

$E_{job} = \sum_r^{resources} E_r \frac{C_r}{C_{sys}}$	0.15	[dimensionless]
A_{sys}	0.70	[dimensionless]
$U_{sys} = \sum_{p=1}^N U_p$	3.16	[M\$/Tflops/sec]
$E_{adm} \equiv \sum_{p=1}^N \sum_r^{resources} \bar{E}_{pr}^{adm}$	0.41	[dimensionless]
E_{util}	0.30	[dimensionless]
$R = \sum_r \frac{C_r}{C_{CPU}} R_r$	550	[TFlops]
$Utility = U_{sys} E_{util} E_{adm} A_{sys} E_{job} R$	22	[M\$]
$C_{sys} = C_{sys}^{initial} + c_{sys}^{annual} T$	290	[M\$]
$n_{adm} = (n_{adm}^{base} + n_{adm}^{productivity})$	8.0	[persons]
$C_{adm} = c_{adm} n_{adm} T$	12.6	[M\$]
$Cost = C_{sys}^{initial} + c_{sys}^{annual} T + c_{adm} n_{adm} T + N c_{proj} \bar{i}_{proj}$	505	[M\$]
$Productivity = \frac{Utility}{Cost}$	0.044	[dimensionless]

Vendor Cost Entries

		<i>Entry Quality</i>	<i>Units</i>	
$C_{sys}^{initial}$	200.00	<i>canonical</i>	[M\$]	Initial system cost including facility setup
c_{sys}^{annual}	35.00	<i>wild guess</i>	[M\$/yr]	Annual recurring costs, hardware, software & facility maintenance and planned upgrades, electrical and space
$\frac{C_r}{C_{sys}}$				Fractional resource costs, initial & recurring
<i>CPU</i>	0.33	<i>wild guess</i>	[dimensionless]	r=1
<i>memory</i>	0.33	<i>wild guess</i>	[dimensionless]	r=2
<i>bandwidth</i>	0.33	<i>wild guess</i>	[dimensionless]	r=3
<i>other</i>	0.00	<i>canonical</i>	[dimensionless]	r=4

Host Environment Entries

		<i>Entry Quality</i>	<i>Units</i>	
T	7	<i>policy</i>	[yrs]	System Lifetime
N	9	<i>policy</i>	[dimensionless]	Number of unique program activities (projects)
n_{adm}^{base}	5	<i>wild guess</i>	[persons]	Base administrative personnel count
c_{adm}	225	<i>informed guess</i>	[K\$/person-yr]	Loaded average admin personnel cost
c_{proj}	300	<i>informed guess</i>	[K\$/person-yr]	Loaded average project personnel cost
<i>Projects</i>	U_p			p <i>Benchmarks</i>
<i>Egret</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	1 HPL Stream Ptrans
<i>Jabiru</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	2 HPL Stream Ptrans
<i>Hawk</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	3 HPL Stream Ptrans
<i>Kite</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	4 HPL Stream Ptrans
<i>Finch</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	5 HPL Stream Ptrans
<i>Gull</i>	0.50	<i>policy</i>	[M\$/TFlops/s]	6 HPL Stream Ptrans
<i>Raven</i>	0.10	<i>policy</i>	[M\$/TFlops/s]	7 SSCA2
<i>Condor</i>	0.05	<i>policy</i>	[M\$/TFlops/s]	8 SSCA2
<i>Crow</i>	0.01	<i>policy</i>	[M\$/TFlops/s]	9 SSCA2
				10
				11
				12
				13
				14
				15
				16
				17
				18
				19

Note: Where time dependent values and efficiencies are required, this spreadsheet is not complete as shown. In such environments, the time dependent integrals may be evaluated numerically or the time dependence may be parameterized for inclusion in a revised spreadsheet formulation.

Performance Entries

		<i>Entry Quality</i>	<i>Units</i>	
A_{sys}	0.90	<i>wild guess</i>	[dimensionless]	RAS system availability averaged over resource types.
r_r				Peak resource capacity
<i>CPU</i>	4,000	<i>canonical</i>	[TFlops/sec]	r=1
<i>memory</i>	6,000	<i>canonical</i>	[Tbytes]	r=2
<i>bandwidth</i>	100,000	<i>canonical</i>	[GUPS]	r=3
<i>other</i>	0	<i>canonical</i>		

Job Productivity Entries

		<i>Entry Quality</i>	<i>Units</i>	
\bar{t}_{proj}	25	wild guess	[person-yrs]	Average total project personnel time per project
E_r				Average fraction of peak performance attained
CPU	0.12	wild guess	[dimensionless]	r=1
memory	0.75	wild guess	[dimensionless]	r=2
bandwidth	0.25	wild guess	[dimensionless]	r=3
other	0.00		[dimensionless]	

System-level Productivity Entries

		<i>Entry Quality</i>	<i>Units</i>	
$n_{adm}^{productivity}$	1	wild guess	[persons]	Admin personnel affected by admin productivity tools
E_{pr}^{adm}				Administrative productivity allocation efficiency
CPU	0.60	wild guess	[dimensionless]	r=1 p=1 Egret
memory	0.75	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=2 Jabiru
memory	0.75	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=3 Hawk
memory	0.75	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=4 Kite
memory	0.75	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=5 Finch
memory	0.75	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=6 Gull
memory	0.75	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=7 Raven
memory	0.75	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=8 Condor
memory	0.75	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3
CPU	0.60	wild guess	[dimensionless]	r=1 p=9 Crow
memory	0.75	wild guess	[dimensionless]	r=2
bandwidth	0.60	wild guess	[dimensionless]	r=3

Note: Where time dependent values and efficiencies are required, this spreadsheet is not complete as shown. In such environments, the time dependent integrals may be evaluated numerically, or the time dependence may be parameterized, for inclusion in a revised spreadsheet formulation.

Subjective Productivity Entries

		<i>Entry Quality</i>	<i>Units</i>	
E_{util}	0.60	<i>wild guess</i>	[dimensionless]	Subjective system level productivity efficiency factors

Summary

$E_{job} = \sum_r^{resources} E_r \frac{C_r}{C_{sys}}$	0.37	[dimensionless]
A_{sys}	0.9	[dimensionless]
$U_{sys} = \sum_{p=1}^N U_p$	3.16	[M\$/Tflops/sec]
$E_{adm} \equiv \sum_{p=1}^N \sum_r^{resources} \bar{E}_{pr}^{adm}$	0.65	[dimensionless]
E_{util}	0.60	[dimensionless]
$R = \sum_r \frac{C_r}{C_{CPU}} R_r$	110,000	[TFlops]
$Utility = U_{sys} E_{util} E_{adm} A_{sys} E_{job} R$	45094	[M\$]
$C_{sys} = C_{sys}^{initial} + c_{sys}^{annual} T$	445	[M\$]
$n_{adm} = (n_{adm}^{base} + n_{adm}^{productivity})$	6.0	[persons]
$C_{adm} = c_{adm} n_{adm} T$	9.5	[M\$]
$Cost = C_{sys}^{initial} + c_{sys}^{annual} T + c_{adm} n_{adm} T + N c_{proj} \bar{i}_{proj}$	522	[M\$]
$Productivity = \frac{Utility}{Cost}$	86.4	[dimensionless]

About the authors

Declan Murphy is a Senior Staff Engineer at Sun Microsystems Laboratories. He currently leads the Administrative Environment team for Phase 2 of Sun's DARPA HPCS program, researching techniques to enhance productivity of large high performance computers from the system administration perspective. Prior to that he worked on Sun's N1 vision for future enterprise systems, first leading the definition of the N1 Service Model and then helping jumpstart the N1 product line. Declan spent the bulk of the 1990s working on Sun's highly available clustering products. He implemented high availability support for the distributed lock manager (DLM) of SPARCcluster PDB, which shipped in 1994. He worked on the highly available ORB and other availability infrastructure on the Solaris MC research project at Sun Labs. As this technology was productized in the Sun Cluster 3.0 product, which shipped in 2000, he led the development the high availability infrastructure, was joint technical lead for the overall product, and ultimately architect of Sun Cluster. Before that he worked on system performance analysis at Sun and on service processor and IO support at a minicomputer company. He received BA, BAI degrees in computer engineering from Trinity College, Dublin, Ireland in 1987.

Thomas Nash received his AB in Physics from Princeton in 1965 and PhD also in Physics from Columbia in 1970. His research has been in experimental elementary particle- and astro-physics and in the development of the intensive high performance computing required by these fields. After a post-doc at MIT (as a designer of the experiment that co-discovered the J/ψ and charm quark), he joined the Fermi National Accelerator Laboratory in 1972 and pursued his science there until 2003. At Fermilab, he led building the large experiment that first studied heavy quark mesons at high statistics. He also led the group in the 1980s that demonstrated the first micro-processor based cluster computers. These were instrumental in processing the top quark discovery data. In the 1990s, he headed computing at Fermilab as founding Head of the Computing Division and Associate Director for Information and Technology. During this period he oversaw establishing the Sloan Digital Sky Survey at Fermilab, and chaired the inter-laboratory coordinating committee of Department of Energy (DOE) National Laboratories computing leadership. He sat on several classified DOE advisory committees overseeing WMD non-proliferation technology research. His present research for the California Institute of Technology is with the LIGO detectors aiming at first direct detection of gravity waves. Tom now consults with Sun on the DARPA funded High Productivity Computing Systems (HPCS) project. He has co-authored over 150 papers, with 7 recently identified by Science Watch as among those most cited in 2004-5. He is a Fellow of the American Physical Society.

Lawrence Votta received his B.S. degree in Physics from the University of Maryland, College Park, Maryland in 1973, and his Ph.D. degree in Physics from the Massachusetts Institute of Technology, Cambridge, Massachusetts in 1979. He is a Distinguished Engineer at Sun Microsystems Inc. improving the software and system reliability and availability of Sun's products while pursuing his research interest in high availability

computing and empirical software engineering. He currently is the Productivity Principle Investigator for Sun's DARPA High Productivity Systems Phase II program. Larry has authored or coauthored more than 60 papers and chapters of 2 books in Software Engineering (and 10 papers in Physics) including empirical studies of software development from highly controlled experiments investigating the best methods for design reviews and code inspection to anecdotal studies of a developer's time usage in a large software development. Recently, his work on combined hardware and software telemetry of complex systems and its analysis have led to 9 patent submissions. Larry is a member of the IEEE and ACM.