WORK-BASED PERFORMANCE MEASUREMENT AND ANALYSIS OF VIRTUAL HETEROGENEOUS MACHINES

Stephen L. Ambrosius
Naval Command, Control
and Ocean Surveillance Center
San Diego, CA 92152-7446 USA
sla@nosc.mil

Richard F. Freund
Naval Command, Control
and Ocean Surveillance Center
San Diego, CA 92152-7446 USA
freund@nosc.mil

Stephen L. Scott *
Oak Ridge National Laboratory
Box 2008, Bldg 6012, MS-6367
Oak Ridge, TN 37831-6367 USA
sscott@msr.epm.ornl.gov

Howard Jay Siegel **
Parallel Processing Laboratory
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285 USA
hj@purdue.edu

KEYWORDS: Distributed Computing, Heterogeneous, Measurement, Performance, Work.

ABSTRACT

Presented here is a set of methods and tools developed to provide transportable measurements of performance in heterogeneous networks of machines operating together as a single *virtual heterogeneous machine* (VHM). The methods are work-based rather than time-based, and yield significant analytical information. A technique for normalizing the measure of useful work performed across a heterogeneous network is proposed and the reasons for using a normalized measure are explored. It is shown that work-based performance measures are better than time-based ones because they may be 1) taken while a task is currently executing on a machine, 2) taken without interrupting production operation of the machine network, 3) used to compare disparate tasks, and 4) used to perform second order analysis of machine network operation. This set of performance tools has been used to monitor the utilization of high-performance computing networks, provide feedback on algorithm design and determine the veracity of computing performance models.

* This work was supported by NASA GSRP-HPCC Fellowship - grant number NGT-51161.
** Supported in part by NRaD under contract number N66001-96-M-2277.

1 INTRODUCTION

Many factors contribute to the overall throughput of a heterogeneous suite of computers while running a distributed set of tasks (Siegel *et al*. 1996, Khokhar *et al*. 1993). Machine architecture, memory, I/O speed, network contention and data proximity of the constituent machines are but a few of the important heterogeneities that affect the rate at which a specific set of tasks can complete. Measurement of this overall throughput is important for determining the actual effects of hardware heterogeneity, task scheduling, algorithm selection and other factors that affect the performance of distributed task sets. Also important is the analysis of heterogeneous suite performance to gather information on the individual contributions of those performance factors and the nature of their interactions. These measurements and analyses are useful both in verifying theoretical models of performance as well as tracking and planning the utilization of heterogeneous computing suites.

Presented here is a set of methods and tools developed to provide transportable measurements of performance in heterogeneous networks of machines operating together as a single *virtual heterogeneous machine (VHM)*. The methods are work based rather than time based and yield significant analytical information. A technique for normalizing the measure of useful work performed across a heterogeneous network is proposed and the reasons for using a normalized measure are explored. It is shown that work-based performance measures are better than time-based ones because they may be 1) taken while a task is currently executing on a machine, 2) taken without interrupting production operation of the VHM, 3) used to compare disparate tasks, and 4) used to perform second order analysis of VHM. This set of performance tools has been used to monitor the utilization of high-performance computing networks, provide feedback on algorithm design and determine the veracity of computing performance models.

Central to this paper are the ideas of useful work and normalized work. *Useful work* is the notion that the computational work done by a machine is only valuable to the extent that it furthers the completion of a task. The amount of useful work done on a task in a VHM is not necessarily proportional any machine-based characteristic, such as the number of machine instructions that have been expended upon the task. In fact, in the presence of significant machine heterogeneity within the VHM this is rarely the case. For instance, the number of machine instructions used in completing a task on RISC and CISC architecture machines are not comparable. A better way of describing work is in terms of its proportional contribution to the completion of the task. This idea is further described in Section 2.2. Once a measurement of useful work is made that is not tied to the machine running the task, it is possible to compare the work of different machines executing the same task.

In order to compare the useful work of one task with that of a different task, a method of normalizing the otherwise unitless measurements must be adopted. *Normalized work* is that work which may be compared between tasks. The choice of normalization technique is tightly tied to the desired error tolerance of the comparisons to be derived and to the range and type of heterogeneity present in the machines and tasks to be measured. This is because in the process of normalization is the implicit assumption that one or more processing characteristics are shared by each task to be compared. In actual practice, the degree to which tasks share processing characteristics varies and is difficult to quantify. Complete task profiling solutions to this problem remain elusive (Siegel *et al*. 1996, Ghafoor and Yang 1993). A thorough discussion of the trade-offs involved in selecting a normalization technique is beyond the scope of this paper. However, one such technique is proposed in Section 2.3 for illustration. It uses measurements made on a reference machine to calibrate the measurements taken from other machines in the VHM.

There is considerable work on performance measurement in the literature, e.g., Addison *et al.* (1994), Ambrosius *et al.* (1996), Driscoll and Daasch (1995), Gustafson (1992), Gustafson (1994), Harrison and Patel (1993), Herrarte and Lusk (1991), Jain (1991), and Siegel (1982). In general, the research presented here differs from that previously done in that the methods used are based on a normalized measure of useful work performed rather than direct measures of completion time or system load.

Gustafson (1992) has examined the measurement of useful work over fixed time intervals, concentrating on parallel speedup, while the emphasis in this paper is on the aggregate performance of a network of heterogeneous computers. Over the past several years, he has vigorously pointed out many of the problems with measuring performance by means-based methods, such as counting floating point operations, and argued strongly for using ends-based methods, such as total execution time and solution quality for a given application (Gustafson 1994). This is consistent with the ideas presented here: the performance of a machine or set of machines grouped into a VHM should be measured by the rate of useful work they do for the user.

For the purposes of this paper, a *task* is defined to be a problem statement describing an application. An *algorithm* is defined to be a machine-independent approach to performing a task. A *program* is a machine-dependent implementation of one algorithm, typically the best algorithm to perform the task on a particular type of machine. Thus, a *task solution* is instantiated in the VHM by one or more executable programs, at least one for each type of machine on which it is required to run, that produce a specific useful output product. The terms *task set* and *task set solution* are also used here to describe a non-empty set of tasks and the instantiation of the set of tasks on the VHM, respectively.

The remainder of the paper is structured in three major sections. The next section considers work-based performance theory including: 1) what is work-based performance measurement, 2) definition of useful work, 3) the normalization of work across heterogeneities, 4) the speed-time curve, 5) the resulting performance equations, and 6) utilization factors. Section 3 examines work-based performance in practice. Here the application tool developed around the work-based theoretical concepts is discussed with respect to its two components: computational and graphical. The last section presents some of the advantages and uses of this tool as well as providing some direction for future related research.

2 WORK-BASED PERFORMANCE THEORY

2.1 Work-Based Performance Measurement

Work-based performance measurement is the process of measuring the computational performance of a computer system by the amount of *useful* work it completes, as defined in Section 1. A work-based performance measurement itself is a function of the amount of useful computational work performed. Useful work is discussed in more detail in Section 2.2. In a VHM, this performance number is not necessarily directly proportional to the amount of processor time spent on the problem.

More familiar to most people is the idea of time-based performance measurement. It is possible to quantify the performance of a VHM by measuring the time a task set takes to complete on that VHM. This time measurement becomes really useful only in comparison to other time measurements. Measuring the time a task set takes to complete on a <u>second</u> VHM can establish a relative performance ranking between the VHMs. Measuring the completion time of a second task set solution (e.g., one that performs the same tasks as the first but with different programs) on the original VHM allows a performance ranking of the solutions.

Time-based methods have two deficiencies not suffered by work-based methods. First, the task set must complete before the measurement may be made. This can be inconvenient for task sets that run for long durations. In certain application domains, such as cryptography and quantum chromodynamics, task sets may run continuously for days or weeks. Second, time-based measurements are limited in their degree of comparability. For instance, a time-based performance measurement for a particular task set may not be generally compared with a time-based performance measurement for a different task set on the same VHM. This is because there is no information with which to compare the aggregate useful work product of each task set. Comparison of the time-based measures of multiple task sets over multiple VHMs are similarly confounded.

The critical difference between time-based and work-based performance measurement is the presence of useful work information in the latter. By measuring the quantity of useful work performed over a specific time interval relative to a single machine, as will be described in Section 2.3, performance made may be compared between task sets, even between VHMs. Additionally, performance may be measured during task set execution. Before discussing specific work-based methods, useful work will be considered in more detail.

2.2 Useful Work

Technically, work with a constant force is defined as force times displacement: $W = F \times s$. Thus, the displacement of an object must occur before one may claim that work has occurred on behalf of the force applied to the object (Kane and Sternheim 1988). Under this restriction, force without displacement $(s = 0)$ results in no work being performed on the object: $0 = F \times 0$. Of course, a person attempting to move a heavy weight will argue that they are working.

In a nontechnical sense, they are indeed working via their muscle contractions in an attempt to move the weight. However, according to the law of physics outlined above, their force is not working on the object they are attempting to move. In order for their applied force to work on the object, the object itself must be displaced.

Consider the case where a program is spinning in an infinite loop and not actually progressing towards a solution. Is the machine working? Following the lead of Physics in defining work, the answer will have to be "no, the computer is not working." Although the computer is consuming energy by expending computing cycles on the problem, it is not progressing (causing a displacement towards the solution as a result of the computing force). Therefore, it may be concluded that no work is being performed on behalf of the computation force being applied to the code.

Now, what happens to this concept of useful work if the move is made from the easily seen example above to one where an algorithm is simply less efficient in comparison to another in solving the same problem? Consider the few lines of code necessary to find the largest of three distinct integers. This code may be written in a number of programming languages using many approaches (algorithms). Two simple algorithms to solve this problem are given in Table 1 and Table 2. Using the traditional computer science technique of counting the number of instructions executed to indicate the work performed, it is easily seen that the algorithm in Table 2, on average, performs more work than does the algorithm in Table 1. It is the assertion of this research that regardless of which algorithm is used, the same amount of useful work is performed. The useful work performed is *finding the largest of three distinct integers*. Note that this statement of work does not discuss instruction count. Thus, instruction count is not relevant with respect to the

Table 1: Three operations to find the largest of three distinct integers.

```
                                  largest:  a   b    c
                                             -   -   - _ -
if (a > b)                                     1   1   1   1
  if (a > c)                                   1       1
    largest = a;    a > b,  a > c,  b ? c          1
  else
    largest = c;    c > a > b                              1
else
  if (b > c)                                       1       1
    largest = b;    b > a,  b > c,  a ? c                  1
  else
    largest = c;    c > b > a                                  1
                                             -   -   -   -
                          Instruction count:   3   3   3   3
```

Table 2: Four operations (on average) to find the largest of three distinct integers.

```
                                  largest:  a   b    c
                                             -   -   - _ -
largest = a;                                    1   1   1   1

if (b > largest)                                 1   1   1   1
  largest = b;      b > a,  a,b ? c                    1   1

if (c > largest)    c > a,b,  a ? b                  1   1   1   1
  largest = c;                                         1   1
                                             -   -   -   -
                          Instruction count:   3   4   5   4
```

resulting amount of useful work performed.

Stated more formally, useful work is computational activity that directly contributes to the completion of the task. The amount of useful work performed in a machine over a period of time is equal to the proportion of the task completion obtained during that interval. There are many ways by which task completion can be measured, some of them domain-dependent. For instance, if a task seeks to solve a system of *N* linear equations, then task completion may be measured in terms of the ratio of the number of equations solved to *N*. For the purposes of this paper, task completion is measured in terms of the proportion of the expected task completion time elapsed[1].

This is shown in Equation 1.0 of Table 3. Of course, this method assumes that the rate of useful work being performed over the life of the task is constant (the work rate average), when in fact it is likely to be variable. The amount of error introduced when making interim work measurements in this manner on a VHM containing currently executing tasks grows smaller as the size of the VHM task set increases. This is because each task's proportional contribution to the total VHM work shrinks as time progresses.

The appropriate useful work metric for a given situation varies according to the makeup of the tasks being executed on the VHM, as well as the degree of accuracy required of the measurements. What is important is that the metric not depend on individual machine characteristics, such as instruction count or memory access count. A normalized unit of work is needed. This is introduced below and elaborated on in Section 2.3.

A common every day example similar to the machine normalized measure problem would be the excavation of a house basement. Here, the work could be done using a number of methods ranging from hand dug (least efficient) to diesel backhoe (most efficient). However, the future homeowner generally does not ask the excavation company about their technique when inquiring about price. Instead, the future provides the contractor with the desired basement dimensions. The contractor then computes the volume of dirt to remove and prices accordingly. Notice that nowhere in this discussion does the contractor or future homeowner count the number of scoops necessary, based on the contractor's digging technique, to excavate the basement. This counting does not have to take place as the contractor has already calculated the costs associated with removing a given unit of dirt. If this unit of dirt is one cubic yard, then it may be said that

1. It is simply a convenience to use the choice of time elapsed divided by the expected execution time to determine what percentage of the previously computed total normalized useful work has been done. For example, the VHM scheduling system upon which the work-based methods were originally installed and evaluated, SmartNet (Freund *et al*. 1996), provides estimated completion times for the tasks it schedules.

excavation work is normalized based on the amount of work required to remove one cubic yard of dirt. This normalization of the work greatly reduces the complexity involved in calculating excavation costs. Thus, the question arises, can a work normalization technique provide the same benefits when applied to calculating computational work and its associated cost?

2.3 Normalized Measurement of Work

The aggregate useful work performed on a VHM is the sum of the useful work performed on the machines in that VHM. Two common measures of work used in computer science are the count of all machine instructions executed and the count of just the floating point operations performed. When a VHM contains more than one machine, then two conditions may be present that make problematic the use of either of these measures to glean useful work. If there are machine heterogeneities present, such as differences in instruction sets or number of registers, then the number of machine instructions or floating point operations to perform the same algorithm may vary among VHM machines. Similarly, if there is algorithm heterogeneity, such as two different algorithms solving the same task, then the number of machine instructions or floating point operations performed may again vary. Here a *normalized work* metric that is machine and algorithm independent is desirable so that individual heterogeneous performance measurements may be adjusted to become meaningful when compared with one another. Thus, the reason for the useful work discussion in Section 2.2. This normalized work metric should relate to useful work only.

The problem lies with how to represent the useful computational work performed without directly connecting it to a particular program. It is proposed that each program be associated with a base technique for solving the problem. This technique will then be the measurement by which all other programs are judged. This technique may be used in both sequential and parallel computing. It may also be used to measure work in both homogeneous and heterogeneous computing environments.

The base technique used for the work in this paper is defined by creating a reference (perhaps sequential) version of the task (j), and benchmarking that task on a machine designated as the reference machine (rm). The resulting execution time of the benchmark process is called $Tnormal_{(j,rm)}$ and all solutions to the same problem will be related to this value. This time metric in itself is one measure of useful work performed given in units of execution time on the appointed reference machine. The time metric, $Tnormal_{(j,rm)}$, is essential in determining the *normalized work* value, which is constant across all machines in the VHM.

For the purposes of this paper, it is convenient to create a new unit of work called the *heterogeneous operation* (*OP*). This OP value is the standardized unit of measurement. Normalized work for performance of a given task in OPs (plural, not seconds) is defined as $Work_{(j,rm)} = S_{rm} \times Tnormal_{(j,rm)}$, where $Tnormal_{(j,rm)}$ is the time metric derived above for task j on machine rm and S_{rm} is some constant value associated with the reference machine rm (e.g., floating point arithmetic speed, clock speed, memory access speed, or some mix). Clearly, this work equation is very closely related to the standard Physics formula for work. Here $Tnormal_{(j,rm)}$ represents the displacement in time units given the assumption of a constant work to time ratio. The S_{rm} represents the computational force.

Based on the normalized work concept, a series of equations may now be derived to represent the current work, total work, percent of a task completed, and current and average speed o*f* individual machines, as well as homogeneous and heterogeneous clusters of machines. These equations, provided in Table 3, form the mathematical basis of the work-based performance

Table 3: Performance equations.

m	A specified machine.
j	Task presently executing on a given machine.[1]
n	Number of machines in the VHM.
t_0	Initial start time.
T	Elapsed time from t_0 to desired sample time.
tc_m	Elapsed time from t_0 to the time the previous task completed processing on machine m.
$t_{(j,m)}$	Elapsed time since current task j started (tc_m) on machine m to time T.
$etc_{(j,m)}$	Estimated time to complete task j on machine m.[2]
rm	Reference machine.
S_{rm}	Predetermined constant value associated with the reference machine rm.
$Tnormal_{(j,rm)}$	Time to execute task j on machine rm.

$Work_{(j,rm)} = S_{rm} \times Tnormal_{(j,rm)}$	0.0	Normalized work for task j on machine rm.
$CurrentWork_{(j,m)} = \dfrac{t_{(j,m)}}{etc_{(j,m)}} \times Work_{(j,rm)}$	1.0	Current work for task j on machine m at time T. (Maximum value is $Work_{(j,rm)}$.)
$CurrentWorkVHM = \sum_{m=0}^{<n} CurrentWork_{(j,m)}$	1.1	Current work on n-machine VHM at time T.
$TotalWork_m = CurrentWork_{(j,m)} + \sum_{i=0}^{<j} Work_{(i,rm)}$	2.0	Total work on machine m at time T.
$TotalWorkVHM = \sum_{m=0}^{<n} TotalWork_m$	2.1	Total work on n-machine VHM at time T.
$PercentCompleted_{(j,m)} = \dfrac{t_{(j,m)}}{etc_{(j,m)}} \times 100.00$	3.0	Percent of task j completed on machine m at time T (Maximum value is 100%.)
$T = t_{(j,m)} + tc_m$	4.0	Time T may be represented on any machine m.
$CurrentSpeed_{(j,m)} = \dfrac{CurrentWork_{(j,m)}}{t_{(j,m)}} = \dfrac{Work_{(j,rm)}}{etc_{(j,m)}}$	5.0	Current speed of single machine m on current task j at time T.
$AverageSpeed_m = \dfrac{TotalWork_m}{T}$	6.0	Average speed of machine m to time T.
$AverageSpeedVHM = \dfrac{TotalWorkVHM}{T}$	6.1	Average speed of entire VHM to time T.
$AverageCurrentSpeedVHM = \dfrac{\sum_{m=0}^{<n} CurrentSpeed_{(j,m)}}{n}$	7.0	Average current speed of n-machine VHM to time T.

1. For each machine, tasks are numbered separately from 0 to j (in order of execution).
2. Determining $etc_{(j,m)}$ values is a research area itself and is beyond the scope of this paper.

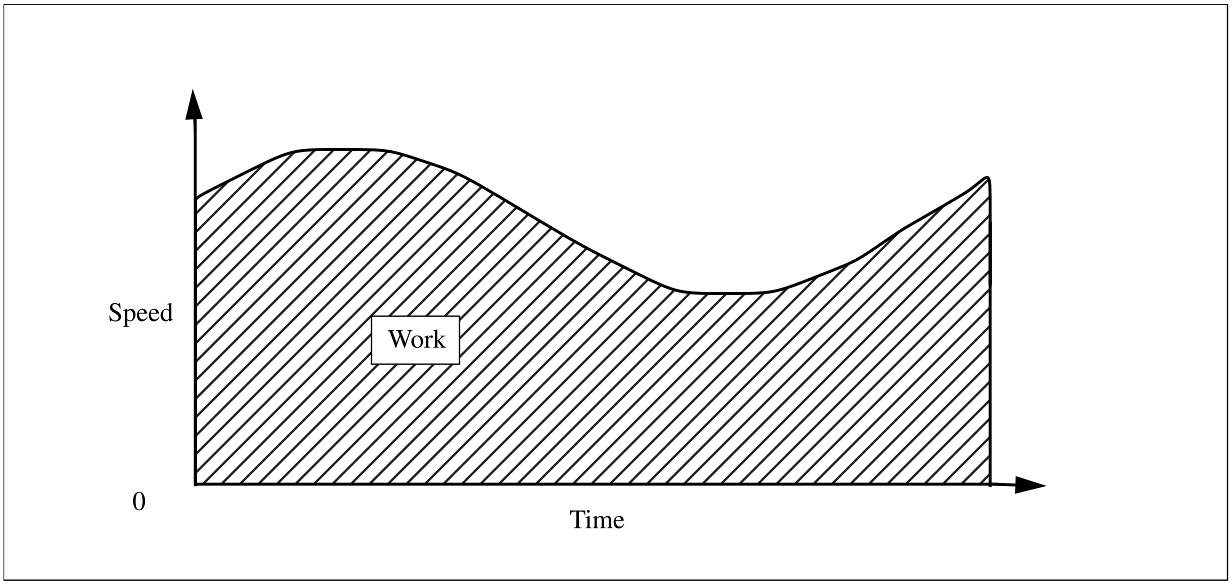

Figure 1: The VHM speed-time curve.

measurement and analysis of VHMs presented in this work. The equations are built on one another by first deriving the appropriate function for a single machine and then extending that to handle a cluster of virtual heterogeneous machines. Table 3 will be further discussed in Section 2.5.

2.4 The Speed-Time Curve

Once the measure of useful work has been normalized over all the applications running in the VHM, a similarly normalized measurement of the current speed of the VHM can be formulated as the sum of the current speeds of all the machines in the VHM. The speed of any one machine in the VHM is given in Table 3, equation 5.0 as $CurrentSpeed_{(j,m)} = \frac{Work_{(j,rm)}}{etc_{(j,m)}}$, where $Work_{(j,rm)}$ represents the normalized work for task j at time T and $etc_{(j,m)}$ represents the estimated time of completion for task j. Discussions about speed in this paper will for simplicity assume that component machines are single-tasking and that expected application completion time is known, although the methods presented here are not constrained by these assumptions.

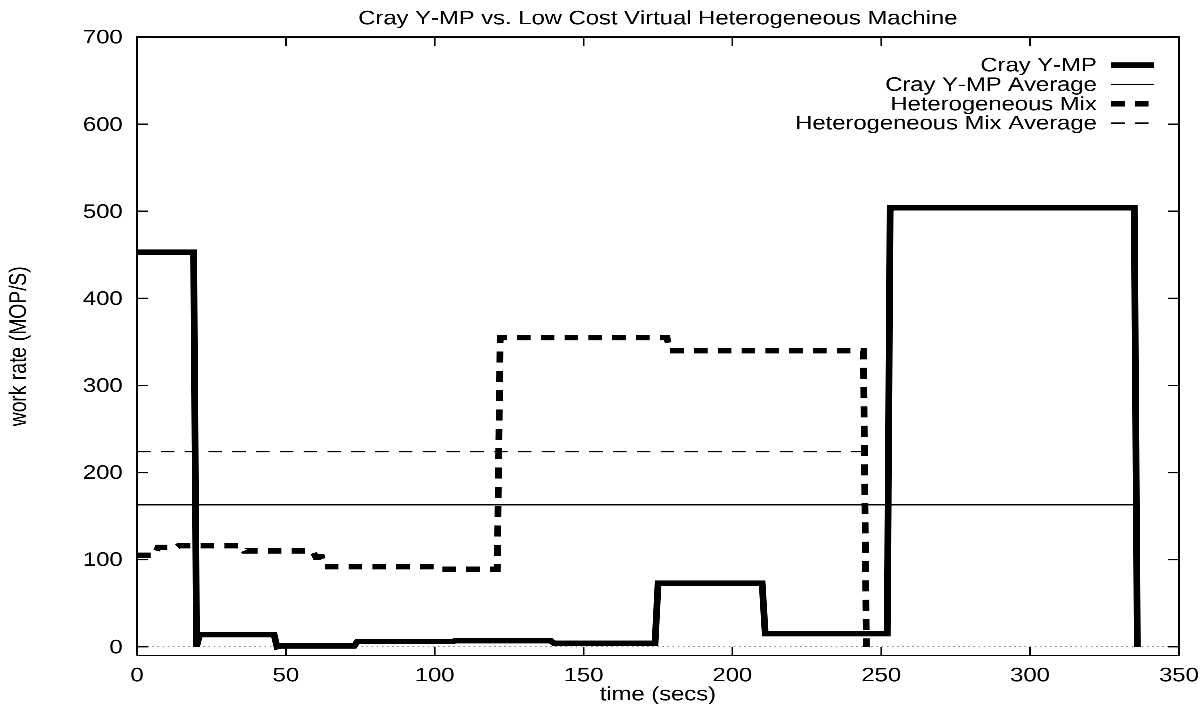

Figure 2: Two examples of work-based speed-time curves.

A hypothetical speed-time curve is shown in Figure 1. When the total speed of the VHM, the rate at which useful work is being accomplished, is plotted over time the total normalized work accomplished by the VHM is represented by the area under the curve. This can be formally defined as the integral equation $TotalWorkVHM = \int_0^T Speed(t)\ (dt)$, where *Speed(t)* is the work rate of the VHM at time *t*. Figure 2 shows an example of two actual speed-time curves and their equivalent average execution rates. Each represents the performance over time of the same set of tasks (nine NAS benchmark tasks operating on a variety of data set sizes) on two different systems. The first is a single Cray Y-MP. The second is a VHM consisting of one each of an HP, IBM, and Sun workstation, as well as one Cray EL92. The "average" lines shown in the graph have an area underneath them equal to that of the area of underneath their parent curve. Furthermore, the area underneath the average line for the Cray Y-MP and the area underneath the average line for the heterogeneous mix are equal because they both represent the same amount of total normalized work. The average lines represent the effective average work output during the run of each set of tasks. Every physical machine in this example executed one task at a time (single-tasking). Because of this, each horizontally flat segment of the Cray Y-MP speed-time curve happens to represents the speed of a different NAS benchmark on that machine, nine in total, though some segment boundaries may be difficult to distinguish from the graph, such as the one at time=107 seconds. The horizontally flat segments of the VHM, because they are derived from summations of the work being performed on all active machines, cannot be directly construed to correspond to any particular task or set of tasks.

Although the VHM consists of machines totaling much less in cost than the Cray Y-MP, the graph shows that the VHM outperforms the Cray Y-MP for this task set (243 vs. 337 seconds). Because the work unit used, Millions of OPs (*MOPs)*, has been normalized over the applications being run, work performed on serial processors can be compared with that performed on parallel ones. Similarly, comparisons of the rate at which different applications execute on the same processor may also be made. This is shown in the speed-time curve for the Cray Y-MP in Figure 2.

Viewing the speed-time curves in Figure 2 gives insight into the disparity in completion times for the two execution runs and also shows an important result of heterogeneous computing. The speed averages given in the graph show the VHM was 39% faster on average than the Cray Y-MP. The graph also shows that the Cray Y-MP is capable of higher processing speeds than the VHM for the given applications and data sets. However, the Cray Y-MP does not perform at this higher speed with all of the tasks it is required to process. Overall the VHM average work rate is greater because of its ability to take advantage of the task set heterogeneity (Freund 1989, Freund 1991).

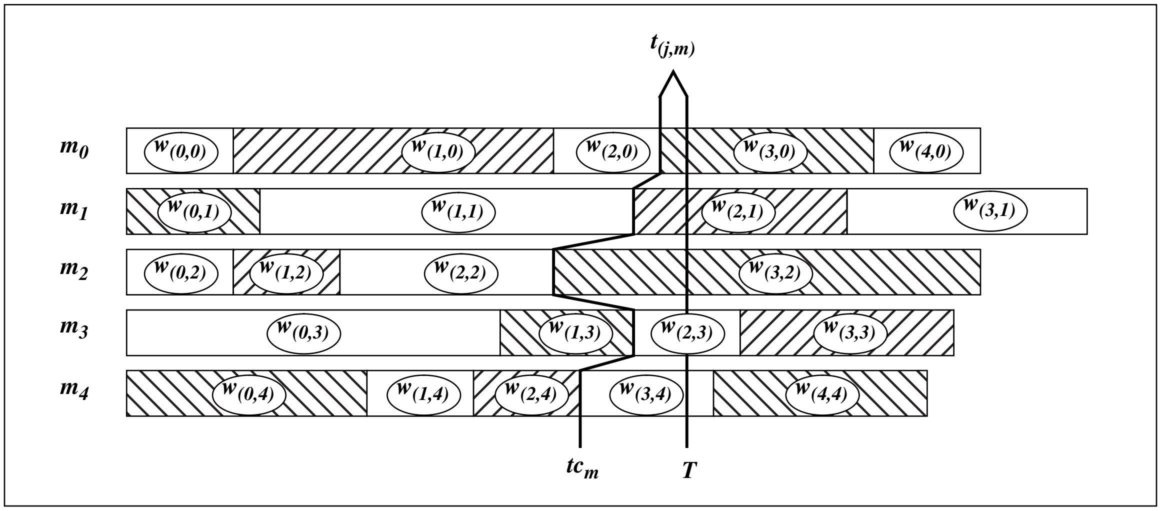

Figure 3: Five machine VHM executing at time T, where $w_{(i,k)}$ is the ith task executed by machine m_k.

This permits applications to be scheduled on machines for which they have a processing affinity.

2.5 Performance Evaluation

Using Figure 3, the performance equations in Table 3 may be derived for a single machine, as well as groups of machines combined into a virtual machine. For simplicity, the equations assume a constant work to time ratio. That is, a machine is assumed to be performing work at a constant rate throughout the execution of an individual task. It is further assumed that each task is allocated to a single machine. However, the notion of task is scalable such that each task may be recursively subdivided further into smaller subtasks with these new subtasks now being referred to as tasks. This subdividing of tasks into subtasks is not restricted by the equations in Table 3. Therefore, the use of the word task is not a restriction, but instead is a language convention for clarity of this discussion. In Figure 3, machines m_0, m_1, m_2, m_3, and m_4 are executing tasks $w_{(3,0)}$, $w_{(2,1)}$, $w_{(3,2)}$, $w_{(2,3)}$, and $w_{(3,4)}$, respectively, at time T. For each machine in Figure 3, the task(s) numerically preceding those listed above have already completed execution.

2.6 VHM Utilization

Quantifying the utilization of a particular machine or network of machines has long been desirable. From a system planning or administration point of view, it is very important to know what spare capacity exists in a computing system.

Utilization can be thought of in two ways: utilization of machine resources and utilization of potential work capacity. Measuring a set of one or more machine resources is an important and commonly used way of measuring absolute machine utilization (Freund 1989, Freund 1991, Harrison and Patel 1993, and Jain 1991). Typical resources to be monitored are CPU time, memory, disk, and I/O. *Utilization* is then defined as the ratio of the amount of resource consumed $R_{consumed}$ to the amount available $R_{available}$ calculated as an average over a time period: $U = R_{consumed}/R_{available}$. In this case, the utilization of a VHM is the simple sum of the component machines utilization $U_{VHM} = \left[\sum_{m=0}^{<n} R_{consumed}\right] / \left[\sum_{m=0}^{<n} R_{available}\right]$.

In contrast, a measurement of utilization by normalized work describes how well the task set is making use of the machine. This is because normalized work is a way of expressing the goodness of fit between a set of tasks and a set of machines. This goodness of fit has as its root the affinity of an application for a machine. In other words, this affinity represents how well an application's computing requirements are satisfied by a particular machine.

Utilization of potential work capacity (work-based utilization) U_{work} can be expressed as the ratio of the rate of normalized work performed $S_{current}$ (An example of this is $CurrentSpeed_{(j,m)}$ in Equation 5.0 of Table 3) to the speed achieved when the machine is running an application best suited for that machine S_{best}, $U_{work} = S_{current}/S_{best}$. For example, take the case of a SIMD massively parallel machine that has been measured as achieving a speed of 8000 MOP/S performing raster-based image processing task that is highly suited to its parallel architecture. That same machine may achieve only 2000 MOP/S executing a discrete event-based simulation task, due to the asymmetrical processing, and may be said to be utilizing $U_{work} = S_{current}/S_{best} = 2000/8000 = 1/4$ or 25% of the machine's known capacity to perform useful work. As indicated in the second paragraph of this section, the utilization of the VHM is the simple sum of the utilization of the component machines. It should be clear from Sections 2.2 and 2.4 that a VHM may be busy, using computing resources at a high rate, yet it may not be operating well in achieving a large portion of its potential capacity for work. Work-based utilization measurement of VHMs gives information on this area by measuring the appropriateness of the task set being processed for the VHM and task scheduling being used.

3 WORK-BASED PERFORMANCE PRACTICE

3.1 Speed-O Overview

A VHM performance analysis tool called *Speed-O* has been constructed at the Naval Command, Control, and Ocean Surveillance Center (*NCCOSC*) in San Diego according to the principles described in Section 2. It is being used to measure the performance of VHMs scheduled by *SmartNet*, a heterogeneous scheduling tool developed at NCCOSC (Freund *et al*. 1996). Performance measurement by Speed-O may be made in real time as well as a replay mode. Speed-O consists of two components: computational and graphical.

3.2 Computational Component

Speed-O tracks the state of tasks being performed in the VHM such that its current and average performance may be measured. Cumulative work is also evaluated and profiled. As discussed in Section 2, the use of normalized work quantities in the VHM performance analyzer makes it suitable for use with any time metric derived for a reference machine.

Configuration for Speed-O operation consists of filling the SmartNet database with work quantities for each task to be executed on the VHM. As discussed in Section 1, the choice of a technique to derive a normalized work quantity is heavily dependent on the measurement results desired and the task set and VHM involved. One way to set up Speedo-O for use is to benchmark each task to be run on the VHM on a single-processor RISC architecture machine, then multiply the benchmark by the rated instruction execution speed of the benchmark machine. This results in a performance number that approximates the number of benchmark machine instructions executed, assuming a highly compute-bound task. The elapsed-time, completion-time, and task execution activity information required by the equation set in Table 3 is provided by native SmartNet tracking facilities.

For every task currently running on the SmartNet-scheduled VHM, the amount of work that task has accomplished is equal to the fraction of its execution time elapsed multiplied by the amount of normalized work it is known to do. This is represented by equation 1.0 $CurrentWork_{(j,m)}$ in Table 3. However, tasks do not always complete as planned. Machines or processing nodes may go off-line, losing work in progress. Process movement may also occur as the result of rescheduling activities. Speed-O incorporates such events into its calculation of work rate by maintaining information on the outcome of task executions on the VHM.

When performance is analyzed in real time (during VHM work in progress) and prior to actual task completion, the task completion time supplied to Speed-O by SmartNet is an estimate subject to error. Upon task completion, the amount of estimation error, if any, is quantified and VHM

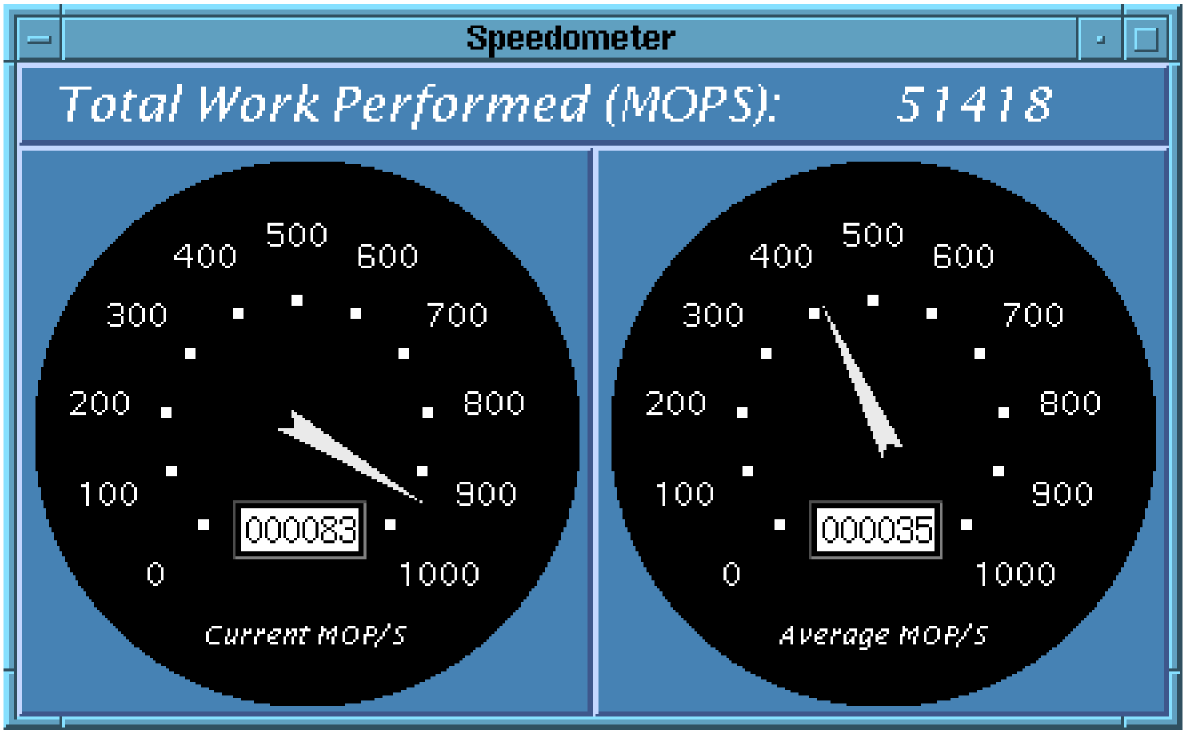

Figure 4: Speed-O dial readout.

performance data is corrected if necessary. Tasks that run longer than their predicted time do not contribute additional work to the VHMs total after they become late in completing. Tasks that run shorter than their predicted time ($etc_{(j,m)}$) cause the VHM to be credited with the higher than predicted average work rate achieved due to that task at the time of task completion.

Speed-O can also analyze activity logs of previously completed task set processing runs (such as those provided by SmartNet). In this case, it can extract the actual task completion time for all tasks and make profiles of VHM performance during a playback of the processing run (called *replay*) that are accurate at all times.

3.3 Graphical Component

To make VHM performance results easily understandable, a graphical dial display and plotting facilities were constructed. The graphical dial display (see Figure 4) indicates current and average VHM work rates as well as the total of all work performed. Optional displays also give the decaying average (average over the last *N* seconds) of VHM event traffic (task starts, completions, aborts, etc.). The dials are auto-scaling (scale the range displayed to include the current value) and have digital utilization readouts embedded in them that describe the current and average readings as a percentage of some selectable reference speeds.

The entire speed-time curve for VHM task sets monitored by Speed-O is available as a two-dimensional plot. An example of this is shown in Figure 2. The average work rate for the curve is plotted as a horizontal line.

4 THE ADVANTAGES OF WORK OVER TIME

The work-based performance measurement methods presented here, when used together, allow the mensuration of VHM performance relative to a designated reference machine. Unlike measures of overall task set completion time, work-based measurement allows the tracking of interim performance levels to gain an insight into VHM operation over time. This is because completion time information for the task set or its component tasks cannot alone express the degree of task set completion while the set is still running, assuming that there is at least some heterogeneity in the tasks or machines.

This unique feature of work-based performance lends itself to monitoring VHMs performing continuous work, such as that found in production metacomputing systems. A *metacomputing* system here is defined as a VHM that can span and consolidate the computing assets of one or more localized HPC centers. In such environments, tasks are continually being submitted to the VHM for execution, while simultaneously other tasks are completing, in general making the timing of a single discrete task set impossible. Even more important to the metacomputing system administrator is that measurements may be taken without temporarily dedicating the VHM to the performance of a particular benchmark task suite to make timings. To measure useful work using only completion time, the metacomputing system administrator would probably need to take the VHM off-line for a period of time to perform a benchmark suite of tasks, and compare the completion time against previous timings.

The ability to continuously monitor work-based performance also has utility in providing real-time feedback for input to task steering by VHM schedulers. These are VHM scheduling decisions (choices about where tasks will run) that are updated or changed during the execution of a task set to increase performance of the VHM as a whole. Unlike homogeneous computing environments, where optimal task scheduling often reduces to opportunistic load balancing, heterogeneous scheduling can benefit greatly from information about the performance affinities between tasks and machines (Freund 1989, Freund 1991, Freund *et al*. 1996) provided by work-based performance measurements. The real-time work-based performance information about machines on the VHM can also be processed for use by administrators of production VHMs to identify subtle performance problems with particular machines during day-to-day operations. Possible problems include: inappropriate machine choice, network overloads, storage bottlenecks, and other more subtle problems that result in performance degradation but not necessarily machine or task failure.

Another important advantage of work-based performance measurement over time-based methods is the comparability of the results obtained. Because work-based performance results are in terms of an absolute standard, useful work, they may be compared with each other in a way that time measures cannot. For example, suppose there are two dissimilar machines, *A* and *B*, and also two different tasks, 1 and 2. This example is shown in Table 4. Task 1 is run on machine *A* and task 2 is run on machine *B*. The completion times of both tasks are measured on the reference machine and the useful work rate is derived. It can be seen from Table 4 that although the completion time (*tc*) for task 1 on machine A was half that of task 2 on machine B, task 2 actually performed useful work at twice the rate (*Speed*) of task 1. Time information alone resulting from heterogeneous sets of machines and tasks cannot provide this comparative information.

Table 4: Time to complete versus speed of work.

$tc_{(j,\,m)}$ - Time to complete task j on machine m.	
$Work_{(1,\,rm)} = 300$ $Work_{(2,\,rm)} = 1200$	$Speed_{(j,\,m)} = \dfrac{Work_{(j,\,rm)}}{tc_{(j,\,m)}}$
$tc_{(1,\,A)} = 10$	$Speed_{(1,\,A)} = 30$
$tc_{(2,\,B)} = 20$	$Speed_{(2,\,B)} = 60$

5 CONCLUDING REMARKS

In this paper, work-based measures of performance have been defined and demonstrated. A set of work-based performance equations was derived that is sufficient to calculate VHM work and speed. The notion of an independent unit of work for VHMs was introduced and an initial effort at defining one was presented. Finally, a work-based VHM performance tool called Speed-O, produced at NCCOSC, was described.

It was shown that work-based measures are better than time-based measures because they may be 1) taken while a task is currently executing on a machine (Sections 2.5, 3.2, and 4.0), 2) taken without interrupting production operation of the VHM (Sections 2.3, 2.5, and 4.0), 3) used to compare disparate tasks (Sections 1.0, 2.2, 2.3, and 4.0), and 4) used to perform second order analysis of VHM operation (Sections 2.4 and 4.0).

Work-based performance measurement can yield information on the causes of VHM efficiency and inefficiency and aid in the comparison of VHM performance models to experimental results. Work-based measures may also be employed in measuring VHMs under conditions of continuous activity (such as operational metacomputing centers).

Work-based performance models and measurement such as have been described have significant potential benefit for metacomputing resource management. Metacomputing centers will need the capacity to perform rapid assessments of their computational readiness and capabilities. Both the metacomputing resources and the kinds of tasks placed on those resources should be considered dynamic in nature. Given the diverse needs of the customers of a metacomputing system, both the resources and tasks are likely to have significant heterogeneities that may be exploited to achieve *superconcurrency* (Freund 1989, Freund 1991), a form of distributed heterogeneous supercomputing. The measurement and management of work efficiency in metacomputing systems will facilitate the configuration work needed to achieve the utmost performance.

An important advantage of work-based performance measurement is that it allows instant comparisons between VHMs and also between VHMs and monoliths (single supercomputers). It allows a second order analysis of some of the causes of VHM efficiency and inefficiency based on actual performance data. This is useful to determining the veracity of heterogeneous computing models.

More work needs to be done on exploiting normal work measures in VHM performance prediction. There is a current study at NCCOSC into the use of work-based performance information to improve task completion time estimates in cases where some completion times are known but not others (as in a sparse completion time matrix). This work has particular applicability to metacomputing centers, where the machine heterogeneity range is large. Also being investigated is the use of work measurements from specific cross-sections (subsets) of machines in the VHM for improving resource management and in providing feedback to parallel algorithm design. Furthermore, additional experience is needed in applying the Speed-O tool to real world applications and standard benchmark suites. This will help to direct the evolution of the Speed-O user-interface as well as proving its usability.

ACKNOWLEDGMENTS

We would like to thank Dr. Taylor W. Kidd and Dr. Debra A. Hensgen of the Naval Postgraduate School for their insight in preparing a preliminary version of this material. The preliminary version was presented at the 5th Heterogeneous Computing Workshop.

REFERENCES

Addison, C. A., Getov, V. S., Hey, A. J. G., Hockney, R. W., and Wolton, I. C., "Benchmarking for Distributed Memory Parallel Systems: Gaining Insight from the Numbers," *Parallel Computing*, Vol. 20, 1994, pp. 1653-1668.

Ambrosius, S. L., Byram, J. K., Campbell, M., Freund, R. F., Kidd, T., Klich, D., Kramer, G., Weissman, M., Wellington, R., "Dynamic Segment Scheduling in JMCIS," *Proceedings of the 1996 Command and Control Research and Technology Symposium*, Monterey, California, June 1996, pp. 669-685.

Driscoll, M. A., and Daasch, W. R., "Accurate Predictions of Parallel Program Execution Time," *Journal of Parallel and Distributed Computing*, Vol. 25, 1995, pp. 16-30.

Freund, R. F., "Optimal Selection Theory for Superconcurrency," *Proceedings of Supercomputing '89*, Reno, Nevada, November 1989, pp. 261-264.

Freund, R. F., "SuperC or Distributed Heterogeneous HPC," *Computing Systems in Engineering*, Vol. 2, No. 4, 1991, pp. 349-355.

Freund, R., Kidd, T., Hensgen, D., Moore, L., “SmartNet: A Scheduling Framework for Heterogeneous Computing,” *Proceedings of 2nd International Symposium on Parallel Architectures, Algorithms, and Networks*, Beijing, China, June 1996, pp. 514-521.

Ghafoor, A., and Yang, J., “Distributed Heterogeneous Supercomputing Management System,” *IEEE Computer*, Vol. 26, No. 6, June 1993, pp. 78-86.

Gustafson, J. L., “The Consequences of Fixed-Time Performance Measurement,” in *Proceedings of the 25th Hawaii Conference on System Sciences*, Kauai, Hawaii, January 1992, pp. 113-124.

Gustafson, J. L., “Teraflops and Other False Goals,” *IEEE Parallel and Distributed Technology*, Vol. 2, No. 2, Summer 1994, pp. 5-6.

Harrison, P. G., and Patel, N. M., *Performance Modeling of Communication Networks and Computer Architectures*, Addison-Wesley, New York, 1993.

Herrarte, V., and Lusk, E., *Studying Parallel Program Behavior with Upshot*, Technical Report ANL-91115, Argonne National Laboratory, Argonne, Illinois, 1991.

Jain, R., *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, New York, 1991.

Kane, J. W., and Sternheim, M. M., *Physics*, 3rd edition, John Wiley & Sons, New York, 1988, pp. 139-141.

Khokhar, A., Prasanna, V. K., Shaaban, M., and Wang, C. L., “Heterogeneous Computing: Challenges and Opportunities,” *IEEE Computer*, Vol. 26, No. 6, June 1993, pp. 18-27.

Siegel, L. J., Siegel, H. J., and Swain, P. H., “Performance Measures for Evaluating Algorithms for SIMD Machines,” *IEEE Trans. Software Engineering*, Vol. SE-8, No. 4, July 1982, pp. 319-331.

Siegel, H. J., Antonio, J. K., Metzger, R. C., Tan, M., and Li, Y. A., "Heterogeneous Computing," in *Parallel and Distributed Computing Handbook*, A. Y. Zomaya ed., McGraw-Hill, New York, 1996, pp. 725-761.