Representing Task and Machine Heterogeneities for Heterogeneous Computing Systems*

Shoukat Ali[1], Howard Jay Siegel[1§], Muthucumaru Maheswaran[2], Debra Hensgen[3], and Sahra Ali[1]

[1]*School of Electrical and Computer Engineering,*
Purdue University,
West Lafayette, IN 47907-1285 USA
Email: alis, hj, sedigh@purdue.edu
[2]*Department of Computer Science,*
University of Manitoba,
Winnipeg, MB R3T 2N2 Canada
Email: maheswar@cs.umanitoba.ca
[3]*OS Research and Evaluation OpenTV,*
Mountain View, CA 94043 USA
E-mail: dhensgen@opentv.com

Abstract

A distributed heterogeneous computing (HC) system consists of diversely capable machines harnessed together to execute a set of tasks that vary in their computational requirements. Heuristics are needed to map (match and schedule) tasks onto machines in an HC system so as to optimize some figure of merit. An HC system model is needed to simulate different HC environments to allow the study of the relative performance of different mapping heuristics under different circumstances. This paper characterizes a simulated HC environment by using the expected execution times of the tasks that arrive in the system on the different machines present in the system. This information is arranged in an "expected time to compute" (ETC) matrix as a model of the given HC system, where the entry (i, j) is the expected execution time of task i on machine j. The ETC model is used to express the heterogeneity among the runtimes of the tasks to be executed, and among the machines in the HC system. An existing range-based technique to express heterogeneity in ETC matrices is described. A coefficient-of-variation based technique to express heterogeneity in ETC matrices is proposed, and compared with the range-based technique. The coefficient-of-variation-based ETC generation method provides a greater control over the spread of values (i.e., heterogeneity) in any given row or column of the ETC matrix than the range-based method.

***Key Words*:** distributed computing, heterogeneous computing, workload characterization, modeling computer systems heterogeneity, modeling workload heterogeneity, cluster computing, grid computing

*This research was supported by the DARPA/ITO Quorum Program under the NPS subcontract numbers N62271-98-m-0217 and N62271-98-M-0448, and under the GSA subcontract number GS09K99BH0250. Some of the equipment used was donated by Intel.

§Beginning August 2001, H. J. Siegel will hold the endowed chair position of Abell Professor of Electrical and Computer Engineering at Colorado State University.

1. Introduction

A distributed heterogeneous computing (<u>HC</u>) system consists of diversely capable machines harnessed together to execute a set of tasks that vary in their computational requirements. Heterogeneous computing systems range from diverse elements or paradigms within a single computer (e.g., PASM [19]), to a cluster of different types of PCs, to coordinated, geographically distributed machines with different architectures (e.g., a grid [6]). An HC system provides a variety of capabilities that can be orchestrated to execute multiple tasks with varied computational requirements [4, 18]. HC systems are important for efficiently solving collections of computationally intensive problems.

These environments achieve high performance by exploiting the affinity of different tasks to different computational platforms, while considering the overhead of inter-machine communication and the coordination of distinct data sources and administrative domains. In an HC system, tasks need to be matched to machines, and the execution of the tasks must be scheduled. The applicability and strength of HC systems are derived from their ability to match computing needs to appropriate resources.

Heuristics are needed to map (match and schedule) tasks onto machines in an HC system so as to optimize some figure of merit. The heuristics that match a task to a machine can vary in the information they use. For example, the current candidate task can be assigned to the machine that becomes available soonest (even if the task may take a much longer time to execute on that machine than else-where). In another approach, the task may be assigned to the machine where it executes fastest (but ignores when that machine becomes available). Or the current candidate task may be assigned to the machine that completes the task soonest, i.e., the machine which minimizes the sum of task execution time and the machine ready time, where machine ready time for a particular machine is the time when that machine becomes available after having executed the tasks previously assigned to it (e.g., [15]).

The more sophisticated (and possibly wiser) approaches to the mapping problem require estimates of the execution times of all tasks (that can be expected to arrive for service) on all the machines present in the HC suite to make better mapping decisions. One aspect of research on HC mapping heuristics explores the behavior of the heuristics in different HC environments. To use simulation to test the relative performance of different mapping heuristics under different circumstances necessitates that there be a framework for generating execution times of all the tasks in the HC system on all the machines in the HC system. Such a framework would, in turn, require a quantification of heterogeneity to express the variability among the runtimes of the tasks to be executed, and among the capabilities of the machines in the HC system. The goal of this invited paper is to present a methodology for synthesizing simulated HC environments with quantifiable levels of task and machine heterogeneity. This paper characterizes the HC environments so that it will be easier for researchers to describe the workload and the machines used in their simulations based on a common scale.

Given a set of heuristics and a characterization of HC environments, one can determine the best heuristic to use in a given environment for optimizing a given objective function. In addition to increasing one's understanding of the operation of different heuristics, this knowledge can help a working re-source management system select which mapper to use for a given real HC environment.

This research is part of a DARPA/ITO Quorum Program project called MSHN (pronounced "mission") (Management System for Heterogeneous Networks) [9]. MSHN is a collaborative research effort that includes the Naval Postgraduate School, NOEMIX, Purdue, and University of Southern California. It builds on SmartNet, an implemented scheduling framework and system for managing resources in an HC environment developed at NRaD [7]. The technical objective of the MSHN project is to design, prototype, and refine a distributed resource management system that leverages the heterogeneity of resources and tasks to deliver the re-quested qualities of service.

A model for describing an HC system is presented in Section 2. Based on that model, two techniques for simulating an HC environment are described in Section 3. Section 4 briefly discusses analyzing the task execution time information from real life HC scenarios. Some related work is outlined in the Section 5.

2. Modeling Heterogeneity

To better evaluate the behavior of mapping heuristics, a model of the execution times of the tasks on the machines is needed so that the parameters of this model can be changed to investigate the performance of the heuristics under different HC systems and under different types of tasks to be mapped. One such model consists of an expected time to compute (ETC) matrix, where the entry (i, j) is the expected execution time of task i on machine j. The ETC matrix can be stored on the same machine where the mapper is stored, and contains the estimates for the expected execution times of a task on all machines, for all the tasks that are expected to arrive for service over a given interval of time. (Although stored with the mapper, the ETC information may be derived from other components of a resource management system (e.g., [9])). In an ETC matrix, the elements along a row indicate the estimates of the expected execution times of a given task on different machines, and those along a column give the estimates of the expected execution times of different tasks on a given machine.

The exact actual task execution times on all machines may not be known for all tasks because, for example, they might be a function of input data. What is typically assumed in the HC literature is that estimates of the expected execution times of tasks on all machines are known (e.g., [8, 12, 14, 20]). These estimates could be built from task profiling and machine benchmarking, could be derived from the previous executions of a task on a machine, or could be provided by the user (e.g., [3, 8, 10, 16, 22]).

The ETC model presented here can be characterized by three parameters: machine heterogeneity, task heterogeneity, and consistency. The variation along a row is referred to as the machine heterogeneity; this is the degree to which the machine execution times vary for a given task [1]. A system's machine heterogeneity is based on a combination of the machine heterogeneities for all tasks (rows). A system comprised mainly of workstations of similar capabilities can be said to have "low" machine heterogeneity. A system consisting of diversely capable machines, e.g., a collection of SMP's, workstations, and supercomputers, may be said to have "high" machine heterogeneity.

Similarly, the variation along a column of an ETC matrix is referred to as the task heterogeneity; this is the degree to which the task execution times vary for a given machine [1]. A system's task heterogeneity is based on a combination of the task heterogeneities for all machines (columns). "High" task heterogeneity may occur when the computational needs of the tasks vary greatly, e.g., when both time-consuming simulations and fast compilations of small programs are performed. "Low" task heterogeneity may typically be seen in the jobs submitted by users solving problems of similar complexity (and hence have similar execution times on a given machine).

Based on the above idea, four categories were proposed for the ETC matrix in [1]: (a) high task heterogeneity and high machine heterogeneity, (b) high task heterogeneity and low machine heterogeneity, (c) low task heterogeneity and high machine heterogeneity, and (d) low task heterogeneity and low machine heterogeneity.

The ETC matrix can be further classified into two categories, consistent and inconsistent [1], which are orthogonal to the previous classifications. For a consistent ETC matrix, if a machine m_x has a lower execution time than a machine m_y for a task t_k, then the same is true for any task t_i. A consistent ETC matrix can be considered to represent an extreme case of low task heterogeneity and high machine heterogeneity. If machine heterogeneity is high enough, then the machines may be so much different from each other in their compute power that the differences in the computational requirements of the tasks (if low enough) will not matter in determining the relative order of execution times for a given task on the different machines (i.e., along a row). As a trivially extreme example, consider a system consisting of Intel Pentium III and Intel 286. The Pentium III will almost always run any given task from a certain set of tasks faster than the 286 provided the computational requirements of all tasks in the set are similar (i.e., low task heterogeneity), thereby giving rise to a consistent ETC matrix.

In inconsistent ETC matrices, the relationships among the task computational requirements and machine capabilities are such that no structure as that in the consistent case is enforced. Inconsistent ETC matrices occur in practice when: (1) there is a variety of different machine architectures in the HC suite (e.g., parallel machines, superscalars, workstations), and (2) there is a variety of different computational needs among the tasks (e.g., readily parallelizable tasks, difficult to parallelize tasks, tasks that are floating point intensive, simple text formatting tasks). Thus, the way in which a task's needs correspond to a machine's capabilities may differ for each possible pairing of tasks to machines.

A combination of these two cases, which may be more realistic in many environments, is the partially-consistent ETC matrix, which is an inconsistent matrix with a consistent sub-matrix [2, 15]. This sub-matrix can be composed of any subset of rows and any subset of columns. As an example, in a given partially-consistent ETC matrix, 50% of the tasks and 25% of the machines may define a consistent sub-matrix.

Even though no structure is enforced on an inconsistent ETC matrix, a given ETC matrix generated to be inconsistent may have the structure of a partially consistent ETC matrix. In this sense, partially-consistent ETC matrices are a special case of inconsistent ETC matrices. Similarly, consistent ETC matrices are special cases of inconsistent and partially-consistent ETC matrices.

It should be noted that this classification scheme is used for generating ETC matrices. Later in this paper, it will be shown how these three cases differ in generation process. If one is given an ETC matrix, and is asked to classify it among these three classes, it will be called a consistent ETC matrix only if it is fully consistent. It will be called inconsistent if it is not consistent.

Often an inconsistent ETC matrix will have some partial consistency in it. For example, a trivial case of partial-consistency always exists; for any two machines in the HC suite, *at least* 50% of the tasks will show consistent execution times.

3. Generating the ETC Matrices

3.1. Range Based ETC Matrix Generation

Any method for generating the ETC matrices will require that heterogeneity be defined mathematically. In the range-based ETC generation technique, the heterogeneity of a set of execution time values is quantified by the range of the execution times [2, 15]. The procedures given in this section for generating the ETC matrices produce inconsistent ETC matrices. It is shown later in this section how consistent and partially-consistent ETC matrices could be obtained from the inconsistent ETC matrices.

Assume m is the total number of machines in the HC suite, and t is the total number of tasks expected to be serviced by the HC system over a given interval of time. Let $U(a, b)$ be a number sampled from a uniform distribution with a range from a to b. (Each invocation of $U(a, b)$ returns a new sample.) Let R_{task} and R_{mach} be numbers representing task heterogeneity and machine heterogeneity, respectively, such that higher values for R_{task} and R_{mach} represent higher heterogeneities. Then an ETC matrix e [0..(t-1), 0..(m-1)], for a given task heterogeneity and a given machine heterogeneity, can be generated by the range-based method given in Figure 1, where e [i, j] is the estimated expected execution time for the task i on the machine j.

As shown in Figure 1, each iteration of the outer **for** loop samples a uniform distribution with a range from 1 to R_{task} to generate one value for a vector τ. For each element of τ thus generated, the m iterations of the inner **for** loop (Line 3) generate one row of the ETC matrix. For the i-th iteration of the outer **for** loop, each iteration of the inner **for** loop produces one element of the ETC matrix by multiplying $\tau[i]$ with a random number sampled from a uniform distribution ranging from 1 to R_{mach}.

In the range-based ETC generation, it is possible to obtain high task heterogeneity low machine heterogeneity ETC matrices with characteristics similar to that of low task heterogeneity high machine heterogeneity ETC matrices if $R_{task} = R_{mach}$. In realistic HC systems, the variation that tasks show in their computational needs is generally larger than the variation that machines show in their capabilities. Therefore it is assumed here that requirements of high heterogeneity tasks are likely to be more "heterogeneous" than the capabilities of high heterogeneity machines (i.e., $R_{task} \gg R_{mach}$). However, for the ETC matrices generated here, low heterogeneity in both machines and tasks is assumed to be same. Typical values for R_{task} for for high and low heterogeneities are 10^5 and 10, respectively. And similarly for R_{mach}, these values are 10^2 and 10, respectively. Tables 1 through 4 show four ETC matrices generated by the range-based method using the above-mentioned typical values for R_{task} and R_{mach}. The execution time values in Table 3 are much higher than the execution time values in Table 2. The difference in the values between these two tables would be reduced if the range for the low task heterogeneity was changed to 10^3 to 10^4 instead of 1 to 10.

Table 1. A low task heterogeneity low machine heterogeneity matrix generated by the range-based method.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7
t_1	22	21	6	16	15	24	13
t_2	7	46	5	28	45	43	31
t_3	64	83	45	23	58	50	38
t_4	53	56	26	42	53	9	58
t_5	11	12	14	7	8	3	14
t_6	33	31	46	25	23	39	10
t_7	24	11	17	14	25	35	4
t_8	20	17	23	4	3	18	20
t_9	13	28	14	7	34	6	29
t_{10}	2	5	7	7	6	3	7
t_{11}	16	37	23	22	23	12	44
t_{12}	8	66	47	11	47	55	56

With the range-based method, low task heterogeneity high machine heterogeneity ETC matrices tend to have high heterogeneity for both tasks and machines, due to method used for generation. For example, in Table 2, original τ vector values were selected from 1 to 10. When each entry is multiplied by a number from 1 to 100 for high machine heterogeneity this generates a task heterogeneity comparable to machine heterogeneity. It is shown in Section 3.2 how to produce low task heterogeneity high machine heterogeneity ETC matrices which do show low task heterogeneity.

```
for i from 0 to (t-1)
  τ[i] = U ( 1, R_task)
  for j from 0 to (m-1)
    e[i,j] = τ[i]× U ( 1, R_mach)
  endfor
endfor
```

Figure 1. The range-based method for generating ETC matrices.

Table 2. A low task heterogeneity high machine heterogeneity matrix generated by the range-based method.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7
t_1	440	762	319	532	151	652	308
t_2	459	205	457	92	92	379	60
t_3	499	263	92	152	75	18	128
t_4	421	362	347	194	241	481	391
t_5	276	636	136	355	338	324	255
t_6	89	139	37	67	9	53	139
t_7	404	521	54	295	257	208	539
t_8	49	114	279	22	93	39	36
t_9	59	35	184	262	145	287	277
t_{10}	7	235	44	81	330	56	78
t_{11}	716	601	75	689	299	144	457
t_{12}	435	208	256	330	6	394	419

Table 3. A high task heterogeneity low machine heterogeneity matrix generated by the range-based method.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7
t_1	333,304	375,636	198,220	190,694	395,173	258,818	376,568
t_2	442,658	400,648	346,423	181,600	289,558	323,546	380,792
t_3	75,696	103,564	438,703	129,944	67,881	194,194	425,543
t_4	194,421	392,810	582,168	248,073	178,060	267,439	611,144
t_5	466,164	424,736	503,137	325,183	193,326	241,520	506,642
t_6	665,071	687,676	578,668	919,104	795,367	390,558	758,117
t_7	177,445	227,254	72,944	139,111	236,971	325,137	347,456
t_8	32,584	55,086	127,709	51,743	100,393	196,190	270,979
t_9	311,589	568,804	148,140	583,456	209,847	108,797	270,100
t_{10}	314,271	113,525	448,233	201,645	274,328	248,473	170,176
t_{11}	272,632	268,320	264,038	140,247	110,338	29,620	69,011
t_{12}	489,327	393,071	225,777	71,622	243,056	445,419	213,477

Table 4. A high task heterogeneity high machine heterogeneity matrix generated by the range-based method.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7
t_1	2,425,808	3,478,227	719,442	2,378,978	408,142	2,966,676	2,890,219
t_2	2,322,703	2,175,934	228,056	3,456,054	6,717,002	5,122,744	3,660,354
t_3	1,254,234	3,182,830	4,408,801	5,347,545	4,582,239	6,124,228	5,343,661
t_4	227,811	419,597	13,972	297,165	438,317	23,374	135,871
t_5	6,477,669	5,619,369	707,470	8,380,933	4,693,277	8,496,507	7,279,100
t_6	1,113,545	1,642,662	303,302	244,439	1,280,736	541,067	792,149
t_7	2,860,617	161,413	2,814,518	2,102,684	8,218,122	7,493,882	2,945,193
t_8	1,744,479	623,574	1,516,988	5,518,507	2,023,691	3,527,522	1,181,276
t_9	6,274,527	1,022,174	3,303,746	7,318,486	7,274,181	6,957,782	2,145,689
t_{10}	1,025,604	694,016	169,297	193,669	1,009,294	1,117,123	690,846
t_{11}	2,390,362	1,552,226	2,955,480	4,198,336	1,641,012	3,072,991	3,262,071
t_{12}	96,699	882,914	63,054	199,175	894,968	248,324	297,691

3.2. Coefficient-of-Variation Based ETC Matrix Generation

A modification of the procedure in Figure 1 defines the coefficient of variation, V, of execution time values as a measure of heterogeneity (instead of the range of execution time values). The coefficient of variation of a set of values is a better measure of the dispersion in the values than the standard deviation because it expresses the standard deviation as a percentage of the mean of the values [13]. Let σ and μ be the standard deviation and mean, respectively, of a set of execution time values. Then $V = \sigma / \mu$. The coefficient-of-variation-based ETC generation method provides a greater control over spread of the execution time values (i.e., heterogeneity) in any given row or column of the ETC matrix than the range-based method.

The coefficient-of-variation-based (CVB) ETC generation method works as follows. A task vector, q, of expected execution times with the desired task heterogeneity must be generated. Essentially, $q[i]$ is the execution time of task i on an "average" machine in the HC suite. For example, if the HC suite consists of an IBM SP/2, an Alpha server, and a Sun SPARC 5 workstation, then q would represent estimated execution times of the tasks on the Alpha server.

To generate q, two input parameters are needed: μ_{task} and V_{task}. The input parameter, μ_{task} is used to set the average of the values in q. The input parameter V_{task} is the desired coefficient of variation of the values in q. The value of V_{task} quantifies task heterogeneity, and is larger for higher task heterogeneity. Each element of the task vector q is then used to produce one row of the ETC matrix such that the desired coefficient of variation of values in each row is V_{mach}, another input parameter. The value of V_{mach} quantifies machine heterogeneity, and is larger for higher machine heterogeneity. Thus μ_{task}, V_{task}, and V_{mach} are the three input parameters for the CVB ETC generation method.

A direct approach to simulating HC environments should use the probability distribution that is empirically found to represent closely the distribution of task execution times. However, no standard benchmarks for HC systems are currently available. Therefore, this research uses a distribution which, though not necessarily reflective of an actual HC scenario, is flexible enough to be adapted to one. Such a distribution should not produce negative values of task execution times (e.g., ruling out Gaussian distribution), and should have a variable coefficient of variation (e.g., ruling out exponential distribution).

The gamma distribution is a good choice for the CVB ETC generation method because, with proper constraints on its characteristic parameters, it can approximate two other probability distributions, namely the Erlang-k and Gaussian (without the negative values) [13, 17]. The fact that it can approximate these two other distributions is helpful because this increases the chances that the simulated ETC matrices could be synthesized closer to some real life HC environment.

The uniform distribution can also be used but is not as flexible as the gamma distribution for two reasons: (1) it does not approximate any other distribution, and (2) the characteristic parameters of a uniform distribution cannot take all real values (explained later in the Section 3.3).

The gamma distribution [13, 17] is defined in terms of characteristic shape parameter, α, and scale parameter, β. The characteristic parameters of the gamma distribution can be fixed to generate different distributions. For example, if α is fixed to be an integer, then the gamma distribution becomes an Erlang-k distribution. If α is large enough, then the gamma distribution approaches a Gaussian distribution (but still does not return negative values for task execution times).

Figures 2(a) and 2(b) show how a gamma density function changes with the shape parameter α. When the shape parameter increases from two to eight, the shape of the distribution changes from a curve biased to the left to a more balanced bell-like curve. Figures 2(a), 2(c) and 2(d) show the effect on the distribution caused by an increase in the scale parameter from 8 to 16 to 32. The two-fold increase in the scale parameter does not change the shape of the graph (the curve is still biased to the left); however the curve now has twice as large a domain (i.e., range on x-axis).

The gamma distribution's characteristic parameters, α and β, can be easily interpreted in terms of μ_{task}, V_{task}, and V_{mach}. For a gamma distribution, $\sigma = \beta\sqrt{\alpha}$, and $\mu = \beta\alpha$, so that $V = \sigma / \mu = 1/\sqrt{\alpha}$ (and $\alpha = 1/V^2$). Then α_{task} = $1 / V_{task}^2$ and $\alpha_{mach} = 1 / V_{mach}^2$. Further, because $\mu = \beta\alpha$, $\beta = \mu / \alpha$, and $\beta_{task} = \mu_{task} / \alpha_{task}$. Also, for task i, $\beta_{mach}[i] = q[i] / \alpha_{mach}$.

Let $G(\alpha, \beta)$ be a number sampled from a gamma distribution with the given parameters. (Each invocation of $G(\alpha, \beta)$ returns a new sample.) Figure 3 shows the general procedure for the CVB ETC generation.

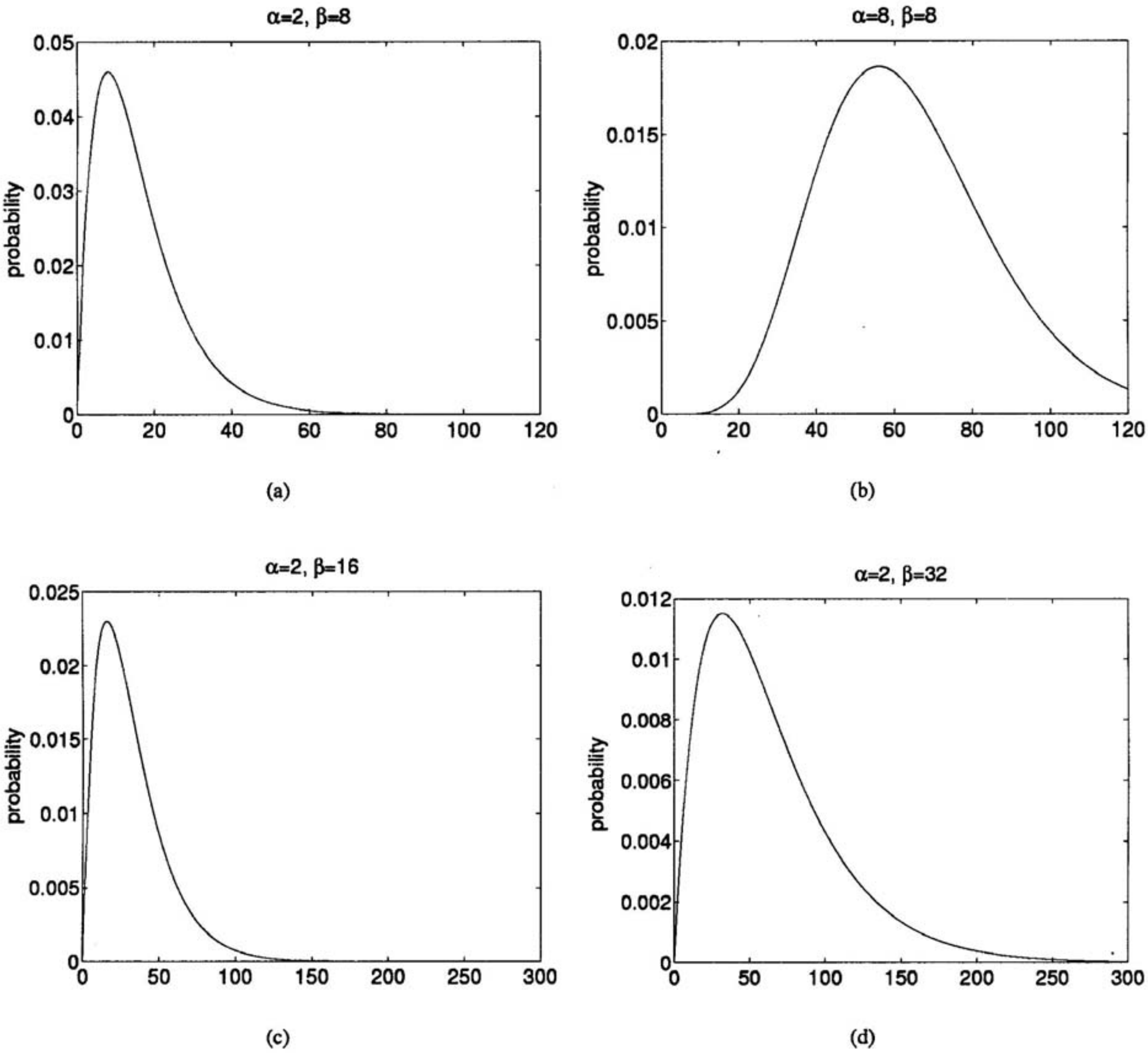

Figure 2. Gamma probability density function for (a) α = 2, β = 8, (b) α = 8, β = 8, (c) α = 2, β = 16, and (d) α = 2, β = 32.

(1) $\alpha_{task} = 1 / V_{task}^{2}$; $\alpha_{mach} = 1 / V_{mach}^{2}$;
    $\beta_{task} = \mu_{task} / \alpha_{task}$
(2) for i from 0 to $(t - 1)$
(3) $q[i] = G(\alpha_{task}, \beta_{task})$
        /* $q[i]$ will be used as mean
        of i-th row of ETC matrix */
(4) $\beta_{mach}[i] = q[i] / \alpha_{mach}$
        /* scale parameter for i-th row */
(5)   for j from 0 to $(m - 1)$
(6) $e[i,j] = G(\alpha_{mach}, \beta_{mach}[i])$
(7)   endfor
(8) endfor

Figure 3. The general CVB method for generating ETC matrices.

Given the three input parameters, V_{task} , V_{mach}, and μ_{task}, Line (1) of Figure 3 determines the shape parameter α_{task} and scale parameter β_{task} of the gamma distribution that will be later sampled to build the task vector q. Line (1) also calculates the shape parameter α_{mach} to use later in Line (6). In the i-th iteration of the outer **for** loop (Line 2) in Figure 3, a gamma distribution with parameters α_{task} and β_{task} is sampled to obtain $q[i]$. Then $q[i]$ is used to determine the scale parameter $\beta_{mach}[i]$ (to be used later in Line (6)). For the i-th iteration of the outer **for** loop (Line 2), each iteration of the inner **for** loop (Line 5) produces one element of the i-th row of the ETC matrix by sampling a gamma distribution with parameters α_{mach} and β_{mach} $[i]$. One complete row of the ETC matrix is produced by m iterations of the inner **for** loop (Line 5). Note that while each row in the ETC matrix has gamma distributed execution times, the execution times in columns are not gamma distributed.

The ETC generation method of Figure 3 can be used to generate high task heterogeneity high machine heterogeneity ETC matrices, high task heterogeneity low machine heterogeneity ETC matrices, and low task heterogeneity low machine heterogeneity ETC matrices, but cannot generate low task heterogeneity high machine heterogeneity ETC matrices. To satisfy the heterogeneity quadrants of Section 2, each column in the final low task heterogeneity high machine heterogeneity ETC matrix should reflect the low task heterogeneity of the "parent" task vector q. This condition would not necessarily hold if rows of the ETC matrix were produced with a high machine heterogeneity from a task vector of low heterogeneity. This is because a given column may be formed from widely different execution time values from different rows because of the high machine heterogeneity. That is, any two entries in a given column are based on different values of $q[i]$ and α_{mach}, and may therefore show high task heterogeneity as opposed to the intended low task heterogeneity. In contrast, in a high task heterogeneity low machine heterogeneity ETC matrix the low heterogeneity among the machines for a given task (across a row) is based on the same $q[i]$ value.

One solution is to generate what is in effect a transpose of a high task heterogeneity low machine heterogeneity matrix to produce a low task heterogeneity high machine heterogeneity one. The transposition can be built into the procedure as shown in Figure 4.

The procedure in Figure 4 is very similar to the one in Figure 3. The input parameter μ_{task} is replaced with μ_{mach}. Here, first a machine vector, p, (with an average value of μ_{mach}) is produced. Each element of this "parent" machine vector is then used to generate one low task heterogeneity column of the ETC matrix, such that the high machine heterogeneity present in p is reflected in all rows. This approach for generating low task heterogeneity high machine heterogeneity ETC matrices can also be used with the range-based method.

(1) $\alpha_{task} = 1 / V_{task}^2$; $\alpha_{mach} = 1 / V_{mach}^2$;
    $\beta_{mach} = \mu_{mach} / \alpha_{mach}$
(2) for j from 0 to $(m-1)$
(3) $p[j] = G(\alpha_{mach}, \beta_{mach})$
        /* $p[j]$ will be used as mean of j-th column of ETC matrix */
(4) $\beta_{task}[j] = p[j] / \alpha_{task}$
        /* scale parameter for j-th column */
(5)   for i from 0 to $(t-1)$
(6) $e[i,j] = G(\alpha_{task}, \beta_{task}[j])$
(7)   endfor
(8) endfor

Figure 4. The CVB method for generating low task heterogeneity high machine heterogeneity ETC matrices.

Tables 5 through 10 show some sample ETC matrices generated using the CVB ETC generation method. Tables 5 and 6 both show high task heterogeneity low machine heterogeneity ETC matrices. In both tables, the spread of the execution time values in columns is higher than that in rows. The ETC matrix in Table 6 has a higher task heterogeneity (higher V_{task}) than the ETC matrix in Table 5. This can be seen in a higher spread in the columns of matrix in Table 6 than that in Table 5.

Table 5. A high task heterogeneity low machine heterogeneity matrix generated by the CVB method. $V_{task} = 0.3$, $V_{mach} = 0.1$.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
t_1	628	633	748	558	743	684	740	692	593	554
t_2	688	712	874	743	854	851	701	701	811	864
t_3	965	1,029	1,087	1,020	921	825	1,238	934	928	1,042
t_4	891	866	912	896	776	993	875	999	919	860
t_5	1,844	1,507	1,353	1,436	1,677	1,691	1,508	1,646	1,789	1,251
t_6	1,261	1,157	1,193	1,297	1,261	1,251	1,156	1,317	1,189	1,306
t_7	850	928	780	1,017	761	900	998	838	797	824
t_8	1,042	1,291	1,169	1,562	1,277	1,431	1,236	1,092	1,274	1,305
t_9	1,309	1,305	1,641	1,225	1,425	1,280	1,388	1,268	1,290	1,549
t_{10}	881	865	752	893	883	813	892	805	873	915

Table 6. A high task heterogeneity low machine heterogeneity matrix generated by the CVB method. Vtask = 0.5 , Vmach = 0.1.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
t_1	377	476	434	486	457	486	431	417	429	428
t_2	493	370	400	420	502	472	475	440	483	576
t_3	745	646	922	650	791	878	853	791	756	788
t_4	542	490	469	559	488	498	509	431	547	542
t_5	625	666	618	710	624	615	618	599	522	540
t_6	921	785	759	979	865	843	853	870	939	801
t_7	677	767	750	720	797	728	941	717	686	870
t_8	428	418	394	460	434	427	378	427	447	466
t_9	263	289	267	231	243	222	283	257	240	247
t_{10}	1,182	1,518	1,272	1,237	1,349	1,218	1,344	1,117	1,122	1,260
t_{11}	1,455	1,384	1,694	1,644	1,562	1,639	1,776	1,813	1,488	1,709
t_{12}	3,255	2,753	3,289	3,526	2,391	2,588	3,849	3,075	3,664	3,312

Table 7. A high task heterogeneity high machine heterogeneity matrix generated by the CVB method. $V_{task} = 0.6$, $V_{mach} = 0.6$.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
t_1	1,446	1,110	666	883	1,663	1,458	653	1,886	458	1,265
t_2	1,010	588	682	1,255	3,665	3,455	1,293	1,747	1,173	1,638
t_3	1,893	2,798	1,097	465	2,413	1,184	2,119	1,955	1,316	2,686
t_4	1,014	1,193	275	1,010	1,023	1,282	559	1,133	865	2,258
t_5	170	444	500	408	790	528	232	303	301	480
t_6	1,454	1,106	901	793	1,346	703	1,215	490	537	1,592
t_7	579	1,041	852	1,560	1,983	1,648	859	683	945	1,713
t_8	2,980	2,114	417	3,005	2,900	3,216	421	2,854	1,425	1,631
t_9	252	519	196	352	958	355	720	168	668	1,017
t_{10}	173	235	273	176	110	127	93	276	390	103
t_{11}	115	74	251	71	107	479	153	138	274	189
t_{12}	305	226	860	554	394	344	68	86	223	120

Table 8. A low task heterogeneity low machine heterogeneity matrix generated by the CVB method. $V_{task} = 0.1$, $V_{mach} = 0.1$.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
t_1	985	1,043	945	835	830	1,087	1,009	891	1,066	1,075
t_2	963	962	910	918	1,078	1,091	881	980	1,009	981
t_3	782	837	968	960	790	800	947	1,007	1,115	845
t_4	999	953	892	986	958	1,006	1,039	1,072	1,090	1,030
t_5	971	972	913	1,030	891	873	898	994	1,086	1,122
t_6	1,155	1,065	800	1,247	980	1,103	1,228	1,062	1,011	1,005
t_7	1,007	1,191	964	860	1,034	896	1,185	932	1,035	1,019
t_8	1,088	864	972	984	736	950	944	994	970	894
t_9	878	967	954	917	942	978	1,046	1,134	985	1,032
t_{10}	1,210	1,120	1,043	1,093	1,386	1,097	1,202	1,004	1,185	1,226
t_{11}	910	958	1,046	1,062	952	1,054	1,020	1,175	850	1,060
t_{12}	930	935	908	1,155	991	997	828	1,062	886	831

Table 9. A low task heterogeneity high machine heterogeneity matrix generated by the CVB method. $V_{task} = 0.1$, $V_{mach} = 0.6$.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
t_1	1679	876	1,332	716	1,186	1,860	662	833	534	804
t_2	1,767	766	1,327	711	957	2,061	625	626	642	800
t_3	1,870	861	1,411	932	1,065	1,562	625	976	556	842
t_4	1,861	817	1,218	865	1,096	1,660	587	767	736	822
t_5	1,768	850	1,465	764	1,066	1,585	663	863	579	757
t_6	1,951	807	1,177	914	939	1,483	573	961	643	712
t_7	1,312	697	1,304	921	1,005	1,639	562	831	633	784
t_8	1,665	849	1,414	795	1,162	1,593	577	791	709	774
t_9	1,618	753	1,283	794	1,153	1,673	639	787	563	744
t_{10}	1,576	964	1,373	752	950	1,726	699	836	633	764
t_{11}	1,693	742	1,454	758	961	1,781	721	988	641	793
t_{12}	1,863	823	1,317	890	1,137	1,812	704	800	479	848

Table 10. A low task heterogeneity high machine heterogeneity matrix generated by the CVB method. $V_{task} = 0.1$, $V_{mach} = 2.0$.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
t_1	4,784	326	1,620	1,307	3,301	10	103	4,449	228	40
t_2	4,315	276	1,291	1,863	3,712	11	91	5,255	200	47
t_3	6,278	269	1,493	1,181	3,186	12	93	4,604	235	46
t_4	4,945	294	1,629	1,429	2,894	14	87	4,724	231	45
t_5	5,276	321	1,532	1,516	2,679	12	102	4,621	205	46
t_6	4,946	293	1,467	1,609	2,661	10	96	3,991	255	39
t_7	4,802	327	1,317	1,668	2,982	10	90	5,090	252	42
t_8	5,381	365	1,698	1,384	3,668	12	99	5,133	242	38
t_9	5,011	255	1,491	1,386	3,061	10	94	3,739	216	42
t_{10}	5,228	296	1,489	1,515	3,632	12	107	4,682	203	38
t_{11}	5,367	319	1,332	1,363	3,393	12	72	4,769	221	43
t_{12}	4,621	258	1,473	1,501	3,124	12	96	4,091	199	44

Tables 7 and 8 show high task heterogeneity high machine heterogeneity and low task heterogeneity low machine heterogeneity ETC matrices, respectively. The execution times in Table 7 are widely spaced along both rows and columns. The spread of execution times in Table 8 is smaller along both columns and rows, because both V_{task} and V_{mach} are smaller.

Tables 9 and 10 show low task heterogeneity high machine heterogeneity ETC matrices. In both tables, the spread of the execution time values in rows is higher than that in columns. ETC matrix in Table 10 has a higher machine heterogeneity (higher V_{mach}) than the ETC matrix in Table 9. This can be seen in a higher spread in the rows of matrix in Table 10 than that in Table 9.

3.3. Uniform Distribution in the CVB Method

The uniform distribution could also be used for the CVB ETC generation method. The uniform distribution's characteristic parameters a (lower bound for the range of values) and b (upper bound for the range of values), can be easily interpreted in terms of μ_{task}, V_{task}, and V_{mach}. (Recall that $V_{task} = \sigma_{task}/\mu_{task}$ and $V_{mach} = \sigma_{mach} / \mu_{mach}$). For a uniform distribution, $\sigma = (b-a)/\sqrt{12}$ and $\mu = (b+a)/2$ [17]. So that

$$a + b = 2\mu \tag{1}$$

$$a - b = -\sigma\sqrt{12} \tag{2}$$

Adding Equations (1) and (2),

$$a = \mu - \sigma\sqrt{3} \tag{3}$$

$$a = \mu(1 - (\sigma / \mu)\sqrt{3}) \tag{4}$$

$$a = \mu(1 - V\sqrt{3}) \tag{5}$$

Also,

$$b = 2\mu - a \tag{6}$$

The Equations (5) and (6) can be used to generate the task vector q from the uniform distribution with the following parameters:

$$a_{task} = \mu_{task}(1 - V_{task}\sqrt{3}) \tag{7}$$

$$b_{task} = 2\mu_{task} - a_{task} \tag{8}$$

Once the task vector q has been generated, the i-the row of the ETC matrix can be generated by sampling *(m* times) a uniform distribution with the following parameters:

$$a_{mach} = q[i](1 - V_{mach}\sqrt{3}) \tag{9}$$

$$b_{mach} = 2q[i] - a_{mach} \tag{10}$$

The CVB ETC generation using the uniform distribution, however, places a restriction on the values of V_{task} and V_{mach}. Because both a_{task} and a_{mach} have to be positive, it follows from Equations (7) and (9) that the maximum value for V_{mach} or V_{task} is $1/\sqrt{3}$. Thus, for the CVB ETC generation, the gamma distribution is better than the uniform distribution because it does not restrict the values of task or machine heterogeneities.

3.4. Producing Consistent ETC Matrices

The procedures given in Figures 1, 3, and 4 produce inconsistent ETC matrices. Consistent ETC matrices can be obtained from the inconsistent ETC matrices generated above by sorting the execution times for each task on all machines (i.e., sorting the values within each row and doing this for all rows independently). From the inconsistent ETC matrices generated above, partially-consistent matrices consisting of an $i \times k$ sub-matrix could be generated by sorting the execution times across a random subset of k machines for each task in a random subset of i tasks.

It should be noted from Tables 9 and 10 that the greater the difference in machine and task heterogeneities, the higher the degree of consistency in the inconsistent low task heterogeneity high machine heterogeneity ETC matrices. For example, in Table 10 all tasks show consistent execution times on all machines except on the machines that correspond to columns 3 and 4. As mentioned in Section 1, these degrees and classes of mixed-machine heterogeneity can be used to characterize many different HC environments.

4. Analysis and Synthesis

Once the actual ETC matrices from a real life scenario are obtained, they can be analyzed to estimate the probability distribution of the execution times, and the values of the model parameters (i.e., V_{task}, V_{mach}, and μ_{task} (or μ_{mach}, if a low task heterogeneity high machine heterogeneity ETC matrix is desired)) appropriate for the given real life scenario. The above analysis could be carried out using common statistical procedures [11]. Once a model of a particular HC system is available, the effect of changes in the workload (i.e., the tasks arriving for service in the system) and the system (i.e., the machines present in the HC system) can be studied in a controlled manner by simply changing the parameters of the ETC model.

This experimental setup can then be used to find out which mapping heuristics are best suited for a given set of model parameters (i.e., V_{task}, V_{mach}, and μ_{task} (or μ_{mach})). This information can be stored in a "look-up table," so as to facilitate the choice of a mapping heuristic given a set of model parameters. The look-up table can be part of the toolbox in the mapper.

The ETC model of Section 2 assumes that the machine heterogeneity is the same for all tasks, i.e., different tasks show the same general variation in their execution times over different machines. In reality this may not be true; the variation in the execution times of one task on all machines may be very different from some other task. To model the "variation in machine heterogeneity" along different rows (i.e., for different tasks), another level of heterogeneity could be introduced. For example, in the CVB ETC generation, instead of having a fixed value for V_{mach} for all the tasks, the value of V_{mach} for a given task could be variable, e.g., it could be sampled from a probability distribution. Once again, the nature of the probability distribution and its parameters will need to be decided empirically.

5. Related Work

To the best of the authors' knowledge, there is currently no work presented in the open literature that addresses the problem of modeling of execution times of the tasks in an HC system (except the already discussed work [15]). However, below are presented two tangentially related works.

A detailed workload model for parallel machines has been given in [5]. However the model is not intended for HC systems in that the machine heterogeneity is not modeled. Task execution times are modeled but tasks are assumed to be running on multiple processing nodes, unlike the HC environment presented here where tasks run on single machines only.

A method for generating random task graphs is given in [21] as part of description of the simulation environment for the HC systems. The method proposed in [21] assumes that the computation cost of a task t_i, averaged over all the machines in the system, is available as $\overline{w_i}$. The method does provide for characterizing the differences in the execution times of a given task on different processors in the HC system (i.e., machine heterogeneity). The "range percentage" (ß) of computation costs on processors roughly corresponds to the notion of machine heterogeneity as presented here. The execution time, e_{ij} , of task t_i on machine m_j is randomly selected from the range, $\overline{w_i} \times (1-\beta/2) \le e_{ij} \le \overline{w_i} \times (1+\beta/2)$. However, the method in [21] does not provide for describing the differences in the execution times of all the tasks on an "average" machine in the HC system. The method in [21] does not tell how the differences in the values of $\overline{w_i}$ for different tasks will be modeled. That is, the method is [21] does not consider task heterogeneity. Further, the model in [21] does not take into account the consistency of the task execution times.

6. Conclusions

To describe different kinds of heterogeneous environments, an existing model based on the characteristics of the ETC matrix was presented. The three parameters of this model (task heterogeneity, machine heterogeneity, and consistency) can be changed to investigate the performance of mapping heuristics for different HC systems and different sets of tasks. An existing range-based method for quantifying heterogeneity was described, and a new coefficient-of-variation-based method was proposed. Corresponding procedures for generating the ETC matrices representing various heterogeneous environments were presented. Sample ETC matrices were provided for both ETC generation procedures. The coefficient-of-variation-based ETC generation method provides a greater control over the spread of values (i.e., heterogeneity) in any given row or column of the ETC matrix than the range-based method. This characterization of HC environments will allow a researcher to simulate different HC environments, and then evaluate the behavior of the mapping heuristics under different conditions of heterogeneity.

Acknowledgment

The authors thank Anthony A. Maciejewski, Tracy D. Braun, and Taylor Kidd for their valuable comments and suggestions. A preliminary version of portions of this material was presented at the 9th Heterogeneous Computing Workshop, sponsored by the IEEE Computer Society and the US Office of Naval Research.

References

[1] Armstrong, R., "Investigation of Effect of Different Run-Time Distributions on Smart-Net Performance," Master's thesis, Naval Postgraduate School (1997) (D. Hensgen, Advisor).

[2] Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L., Maheswaran, M., Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Freund, R. F., and Hensgen, D., "A Comparison Study of Static Mapping Heuristics for a Class of Meta-Tasks on Heterogeneous Computing Systems," *8th IEEE Heterogeneous Computing Workshop (HCW '99)*, pp. 15–29 (1999).

[3] Dietz, H. G., Cohen, W. E., and Grant, B. K., "Would You Run It Here... Or There? (AHS: Automatic Heterogeneous Supercomputing)," *1993 International Conference on Parallel Processing (ICPP '93)*, Vol. II, pp. 217–221 (1993).

[4] Eshaghian, M. M., ed., *Heterogeneous Computing*, Artech House, Norwood, MA (1996).

[5] Feitelson, D. G. and Rudolph, L., "Metrics and Benchmarking for Parallel Job Scheduling," in D. G. Feitelson, L. Rudolph, eds., *Job Scheduling Strategies for Parallel Processin*g, Lecture Notes in Computer Science, Springer-Verlag, New York, NY, Vol. 1459, pp. 1–15 (1998).

[6] Foster, I., Kesselman, C., eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Fransisco, CA (1999).

[7] Freund, R. F., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D., Keith, E., Kidd, T., Kussow, M., Lima, J. D., Mirabile, F., Moore, L., Rust, B., and Siegel, H. J., "Scheduling Resources in Multi-User, Heterogeneous, Computing Environments with SmartNet," *7th IEEE Heterogeneous Computing Workshop (HCW '98)*, pp. 184–199 (1998).

[8] Ghafoor, A. and Yang, J., "Distributed Heterogeneous Supercomputing Management System," *IEEE Computer*, Vol. 26, No. 6, pp. 78– 86 (1993).

[9] Hensgen, D. A., Kidd, T., John, D. S., Schnaidt, M. C., Siegel, H. J., Braun, T. D., Maheswaran, M., Ali, S., Kim, J.-K., Irvine, C., Levin, T., Freund, R. F., Kussow, M., Godfrey, M., Duman, A., Carff, P., Kidd, S., Prasanna, V., Bhat, P., and Alhusaini, A., "An Overview of MSHN: The Management System for Heterogeneous Networks," *8th IEEE Heterogeneous Computing Workshop (HCW '99)*, pp. 184–198 (1999).

[10] Iverson, M. A., Özgüner, F., and Follen, G. J., "Statistical Prediction of Task Execution Times Through Analytic Benchmarking for Scheduling in a Heterogeneous Environment," *8th IEEE Heterogeneous Computing Workshop (HCW '99)*, pp. 99–111 (1999).

[11] Jain, R., *The Art of Computer Systems Performance Analysi*s, John Wiley & Sons, Inc., New York, NY (1991).

[12] Kafil, M. and Ahmad, I., "Optimal Task Assignment in Heterogeneous Distributed Com-putting Systems," *IEEE Concurrenc*y, Vol. 6, No. 3, pp. 42–51 (1998).

[13] Lapin, L. L., *Probability and Statistics for Modern Engineerin*g, Waveland Press, Inc., Prospect Heights, IL, Second edn. (1998).

[14] Lopez-Benitez, N. and Hyon, J.-Y., "Simulation of Task Graph Systems in Heterogeneous Computing Environments," *8th IEEE Heterogeneous Computing Workshop (HCW '99)*, pp. 112–124 (1999).

[15] Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., and Freund, R. F., "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems," *Journal of Parallel and Distributed Computin*g, Vol. 59, No. 2, pp. 107–131 (1999).

[16] Maheswaran, M., Braun, T. D., and Siegel, H. J., "Heterogeneous Distributed Computing," in J. G. Webster, ed., *Encyclopedia of Electrical and Electronics Engineering, Vol.* 8, John Wiley, New York, NY, pp. 679–690 (1999).

[17] Papoulis, A., *Probability, Random Variables, and Stochastic Processe*s, McGraw-Hill, New York, NY (1984).

[18] Siegel, H. J. and Ali, S., "Techniques for Mapping Tasks to Machines in Heterogeneous Computing Systems," *Journal of Systems Architecture*, Vol. 46, No. 8, pp. 627–639 (2000).

[19] Siegel, H. J., Braun, T. D., Dietz, H. G., Kulaczewski, M. B., Maheswaran, M., Pero, P. H., Siegel, J. M., So, J. J. E., Tan, M., Theys, M. D., and Wang, L., "The PASM Project: A Study of Reconfigurable Parallel Computing," *2nd International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN'96)*, pp. 529–536 (Invited paper) (1996).

[20] Singh, H. and Youssef, A., "Mapping and Scheduling Heterogeneous Task Graphs Using Genetic Algorithms," *5th IEEE Heterogeneous Computing Workshop (HCW '96)*, pp. 86–97 (1996).

[21] Topcuoglu, H., Hariri, S., and Wu, M.-Y., "Task Scheduling Algorithms for Heterogeneous Processors," *8th IEEE Heterogeneous Computing Workshop (HCW '99)*, pp. 3–14 (1999).

[22] Yang, J., Ahmad, I. and Ghafoor, A., "Estimation of Execution Times on Heterogeneous Supercomputer Architecture," *1993 International Conference on Parallel Processing (ICPP '93)*, Vol. I, pp. 219–225 (1993).