Realistic Modeling and Synthesis of Resources for Computational Grids

Yang-Suk Kee
Computer Science & Engineering
University of California at San Diego
yskee@csag.ucsd.edu

Henri Casanova
San Diego Supercomputer Center,
Computer Science & Engineering
University of California at San Diego
casanova@sdsc.edu

Andrew A. Chien
Computer Science & Engineering,
Center for Networked Systems
University of California at San Diego
achien@ucsd.edu

ABSTRACT
Understanding large Grid platform configurations and generating representative synthetic configurations is critical for Grid computing research. This paper presents an analysis of existing resource configurations and proposes a Grid platform generator that synthesizes realistic configurations of both computing and communication resources. Our key contributions include the development of statistical models for currently deployed resources and using these estimates for modeling the characteristics of future systems. Through the analysis of the configurations of 114 clusters and over 10,000 processors, we identify appropriate distributions for resource configuration parameters in many typical clusters. Using well-established statistical tests, we validate our models against a second resource collection of 191 clusters and over 10,000 processors, and show that our models effectively capture the resource characteristics found in real world resource infrastructures. These models are realized in a resource generator, which can be easily recalibrated by running it on a training sample set.

Keywords
Computational Grid, resource modeling, clusters, resource discovery, resource selection, resource management

1. INTRODUCTION
Grid technologies enable the sharing and utilization of widespread resources in a coordinated fashion. Numerous research and development activities are ongoing in the Grid computing area, and a common concern is the analysis and evaluation of the resulting techniques and algorithms in "representative'' scenarios; these include typical software, physical resource, and network environments for current or future Grid deployments. Current Grid environments [1-9] exhibit a wide range of software, physical resource, and network configurations. However, common themes are present, such as the prevalence of clustered commodity systems.

To see the importance of understanding current and future Grid configurations, consider, for instance, the wealth of activity in scalable and efficient resource discovery and monitoring. Grid information services (GIS) are core components of the middleware infrastructure. The MDS [10] is often used for initial resource discovery, and resource monitoring tools like NWS [11], Ganglia [12], and Hawkeye [13] are used for measuring dynamic resource characteristics. Understanding the performance properties and evaluating these systems requires the exploration of a wide range of representative resource configuration scenarios.

Similarly, consider resource selection systems such as Matchmaking/ClassAds [14] and Redline [15] that enable applications to express complex resource requirements for sets of resources. Understanding how useful these constraints are, and how they affect the complexity of selections depends directly on the actual resource environments in which they are used. More specifically, researchers are exploring new resource discovery and selection techniques that selectively consider resource information – scoping or sampling. For example, the systems described in [16,17] use a variety of mechanisms to reduce the information that an application needs to retrieve from the Grid Information service. [17] suggests that even searching a randomly selected subset can improve performance by reducing server load. Understanding how well such a technique will work in a broad range of future grids requires the generation of representative resource configurations that can form the basis for performance studies. Further, scoping and random selection may also negatively impact the quality of the discovered resources, but quantifying this impact is only possible with experiments against a variety of representative real-world Grid resource sets.

In short, it is important to evaluate the utility, cost, and scalability of the above systems with a variety of realistic resource configurations. In addition to the evaluation of resource monitoring and discovery mechanisms, realistic Grid resource configuration synthesis is also critical in the evaluation of resource management techniques and policies as well as application scheduling. This cannot be achieved with real hardware Grids, but synthesizing resource sets and evaluation by the means of a simulation system such as MicroGrid [18] is one promising alternative.

Grid environments consist of both network and endpoint resources. In the networking community, various network topology generators are available to synthesize structured

SC'04, Nov. 6–12, 2004, Pittsburgh, PA, U.S.A.

Table 1. Probabilities of processor architectures for the sample set and predicted sample sets (%)

	Pentium2	Celeron	Pentium3	Pentium4	Itanium	Athlon	AthlonMP	AthlonXP	Opteron
Samples	1.4	4.1	40.3	34.6	3.9	0.0	12.4	1.3	2.0
1 year	0.8	2.5	24.5	46.1	5.2	0.0	16.5	1.8	2.6
2 years	0.5	1.6	15.9	52.3	5.9	0.0	18.7	2.0	3.0
3 years	0.4	1.1	10.9	56.0	6.3	0.0	20.0	2.1	3.2

topologies obeying certain power-laws (e.g. Brite [19] and Tier [20]). These network topologies can be used in a Grid emulator, like MicroGrid. In this work, we focus on the compute resources that are interconnected by such networks.

To the best of our knowledge, GridG [21] is the only prior work focusing on the synthesis of realistic Grid resource configurations. GridG uses an annotation scheme to generate the characteristics of single hosts that are individually placed into an Internet network topology. In contrast, we focus on clusters of commodity microprocessors (often called Beowulf clusters), which increase popularity and importance in the Grid. To simplify the task, we exclude specialized computer systems such as vector, SIMD, and some MPP machines. These could be added to a future extension of our model. This simplification is realistic because the rapid advance of commodity systems and networking suggest that such commodity clusters will be the dominant component in Grids. For instance, recent Top500 lists [22] show that clusters are rapidly growing in popularity in the supercomputing market, and they are already dominant in commercial systems.

Our approach to resource generation uses distributions inferred from existing configurations of Grid resources. We use these probability distributions to generate typical configurations for current Grids of arbitrary sizes. Specifically, we build a model for each resource characteristic and generate possible resource configurations accordingly. For some resource characteristics that are not reported in the sample set of real-world systems that we observed, we use a technology-based analysis for performance trends and system configurations provided by system vendors. The resulting models capture the key trends, and can be used to automatically adjust parameters when new empirical data becomes available. Our approach obviates the need for a user to build new models by hand. We use a wide range of existing Grid configurations to generate our statistical models, and then validate them against a second disjoint set of Grid configurations. Finally, we use these statistical models and an analysis of technology trends to extrapolate to future Grid resource configurations. Our contributions are as follows:

- **New models for Grid resource configurations**: based on an extensive study of configurations, totaling over 10,000 processors, we derive appropriate distributions for various resource characteristics. For the number of processors in a node, memory size per node, and the number of nodes per cluster, we find a normal distribution to be a good model. For processor architecture, a multinomial distribution is appropriate. For clock speed, a uniform distribution is a good candidate. For processor cache size, a step function is realistic. Intra-cluster networks are split evenly between Ethernet and one of several proprietary networks (e.g. Myrinet). We combine these attributes into a single statistical model for Grid resource generation.
- **Validation of the resource models**: the models for processor architecture, clock speed, and number of processors are validated by comparing statistical properties of the generator output to another set of sample resource configurations for 191 clusters and over 10,000 machines.
- **Extrapolation to future Grid resource configurations**: all the models except processor architecture and processor clock speed are time-independent or have implicit time-dependent factors. We show how they can be used to predict future grid system configurations in a straightforward fashion.
- **Implementation of the models**: we implemented a resource generator to realize the models. It is currently used to produce resource configurations for evaluating Grid resource discovery systems being developed in our lab.

The remainder of this paper is organized as follows. We describe our methodology for generating models and models for specific resource entities in Section 2. These models are validated against another sample resource set in Section 3. We highlight key implementation issues for our resource generator and its design in Section 4. Finally, we discuss related work in Section 5, and conclude and describe future directions in Section 6.

2. RESOURCE MODELS

The major goals of our study are to develop the models for resources available in current Grids and to extrapolate from these models to future systems. The resource entities of interest are the major contributors to application performance, which include processor architecture, clock speed, L2 cache size, per-node memory size, per-node disk size, per-node number of processors, per-cluster number of nodes, and system area network technology. Especially, processor architecture is crucial to application performance. CPU vendors try to achieve higher performance by optimizing processor microarchitecture, extending instruction set, and increasing processor clock speed. Ideally, our processor models might include microarchitecture, clock rate, cache hierarchy, etc. However, we are limited by the resource information that is readily available. As a result, we use processor architecture, clock speed, and cache size as a basis for our processor resource model.

To estimate the probability distribution of the resource characteristics mentioned above, we gathered a large sample set of existing cluster resources, and use this set as the basis for statistical models for our resource generator. We gathered system configuration data for 114 clusters totaling over 10,000 processors

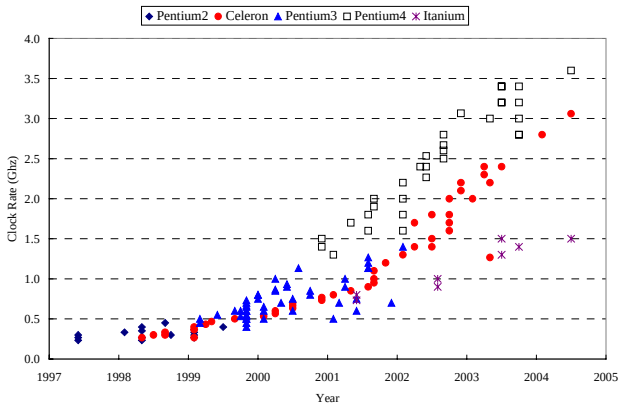

Figure 1. Processor clock speed of Intel processors (Ghz)

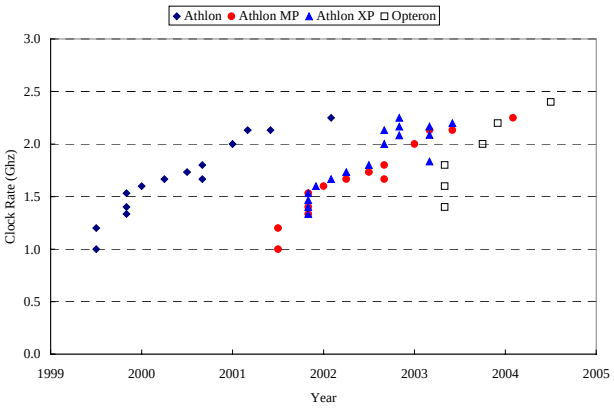

Figure 2. Processor clock speed of AMD processors (Ghz)

from 12 web sites for cluster and Grid computing. Some representative sites include Lawrence Berkeley National Lab[1], Ganglia demo[2], U.C. Berkeley Millennium project[3], and NPACI[4]. Although many clusters are not publicly available, we believe that our sample set is representative because it includes clusters with a variety of popular microprocessors from old Pentium 2 processors to state-of-the-art Opteron processors, and because the cluster system sizes range from single machines to over one thousand nodes. In the next sections, we describe our models for each considered attribute.

2.1 Processor Architecture

Among the factors contributing to computing power, processor architecture is among the most crucial. The first step is to estimate the distribution of processor architectures in the population. For this purpose, we counted the total number of processors per architecture in the sample set (not the number of systems). The sample set includes 10,179 processors. The ratio of processors for each architecture is shown in the first row of Table 1. Two architectures, Intel's Pentium 3 and Pentium 4, are dominant, accounting for 40.3% and 34.6% of all processors, respectively. We assume the sample set is representative and use the sample ratios as the population distribution in the resource generator for current cluster systems.

However, the resource population never stands still, and new systems with state-of-the-art processors are being deployed while old systems are being retired. To extrapolate to future systems, we would ideally know exactly which new processor architectures would be introduced, when they would be available, and their performance. However, we do not have such detailed information. As an alternative, we use an extrapolation of processor clock speed to predict the release dates of new processors. Processor clock speed increases as microprocessor technology advances and users tend to install the newest (or most cost effective) processors available on the market. Hence, the processor clock speed of a system within each processor family is a good indicator of its release year. For instance, the clock speeds of Intel and AMD processors released since 1997 are shown in Figure 1 and Figure 2. The observed trend can be fitted linearly as follows:

$$clock_p = a_p * (release_p - yr_p) + b_p, \quad (1)$$

where $release_p$ is the release year of processor p, yr_p is the release year of the first processor of this architecture, and a_p and b_p are constants. We use a simple linear regression analysis to find a line with a least mean-square error. The slope and intercept are determined independently for each architecture. For example, the slope of the Pentium 4 processor's group is 0.661, the intercept is 0.759, and the coefficient of determination (R-square) is 0.9. The coefficient of determination is used to estimate how well the fitted line is matched to data. When the coefficient value is close to 1, the line is well fitted. Since the R-square value is close to 1, this line is quite well matched to the trend. We can then estimate the release year of a given system by its processor clock speed using the inverse function of equation (1) as follows.

$$release_p = \frac{clock_p - b_p}{a_p} + yr_p \quad (2)$$

Figure 3 shows, for our sample data, the number of processors that were released each year, as computed by equation (2). The trend in this histogram can be fitted by a polynomial with degree 2 as follows:

$$Count_{yr} = a * \Delta_{yr}^{\ 2} + b * \Delta_{yr} + c \quad (3)$$

where Δ_{yr} is the difference of release year based on the year when the first Pentium 2 processor was announced (e.g., 1997). For our sample set, *a* is equal to 190.6, *b* to 437.3, and *c* to 320.2, and the R-square value of the fitted curve is 1.0.

Because we are extrapolating, we assume that future systems will adopt one of the processor architectures currently available in the market: Pentium 4, Itanium 2, Athlon XP/MP, and Opteron. We estimate the sample ratios of these processors and how many processors will be available in the future by equation (3). This allows us to calculate the sample ratios of the processor architectures at points in the future. Because the relative fraction of older processors decreases as more new processors are added to the system, we need not explicitly model system retirement. Our extrapolations for several years are shown in the second, third, and fourth row of Table 1.

[1] http://www.lbl.gov

[2] http://ganglia.sourceforge.net/

[3] http://www.millennium.berkeley.edu/

[4] http://www.npaci.edu/

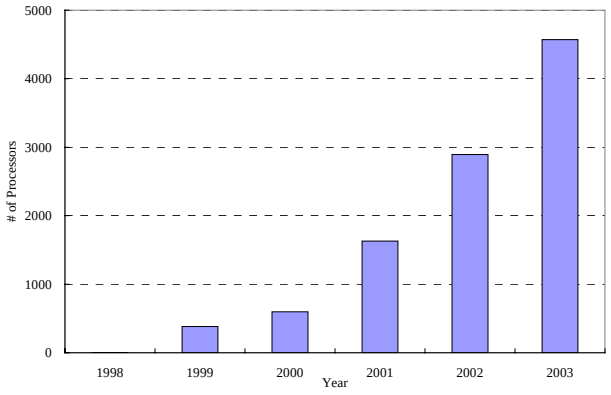

Figure 3. Number of sample processors released every year

2.2 Processor Clock Speed

A key-contributing factor to a processor's computing power is its clock speed. As an approximation of the distribution of clock speeds that actually occur for a given processor architecture, we use a uniform distribution. We postulate that the cause of the dense distribution of certain clock speeds is due to the fact that several processors with different clock speeds are often announced at the same time and some processors with the same clock speed are released several times. In addition, the advent of new processors affects the purchase of existing processors. However, it is very difficult to capture these factors in a simple unified model.

For older processors out of production, we calculate the possible clock speed values using the average clock increment ($\Delta_{(p,clk)}$), the slowest clock speed (c_p), and the fastest clock speed (C_p). Then, there are $(C_p - c_p)/\Delta_{(p,clk)}$ possible clock speeds, one of which is selected randomly between c_p and C_p by the unit of $\Delta_{(p,clk)}$. For example, the slowest clock speed of Pentium 3 processor was 400Mhz, the fastest was 1.4Ghz, and the average clock increment was 50Mhz. As a result, there are 20 possible clock speed values by the units of 50Mhz. Meanwhile, for new processors still in production, we calculate the possible clock speed values using the average clock increment of the processor architecture ($\Delta_{(p,clk)}$), the interval (I_p) in which new versions are released, and the elapsed time after the first one was announced (T_p). Then, one between c_p and $c_p + \Delta_{(p,clk)} * T_p / I_p$ can be randomly selected by the units of $\Delta_{(p,clk)}$. For instance, the slowest clock speed of the Pentium 4 processor was 1.4 Ghz, the average clock increment has been 115Mhz, and new versions have been released every 3 months.

2.3 Processor Cache

Caches are crucial components that affect application performance and programmers often try to optimize cache reuse. Since cache size is determined by processor architecture and manufacturing technology, there seems to be no simple rule for cache size. The L2 cache size of Intel processors and AMD processors are shown in Figure 4 and 5, respectively. The cache size increment is mainly related to the advent of new processors. Celeron processors have 128 KB or 256 KB L2 cache, Pentium 3 processors have 256 KB or 512 KB, and Pentium 4 processors have 256 KB, 512 KB, or 1024 KB. Meanwhile, Athlon, Athlon MP, Athlon XP processors have 256 KB or 512 KB L2 cache, and Opteron processors have 1 MB cache. Due to multithreading support, the recent processors tend to have larger L2 cache.

Table 2. Processor performance factors

Factor	Processor architecture
1	Pentium 2, Celeron, Pentium 3, Athon
2	Pentium 4, Athlon MP, Athlon XP, Opteron
4	Itanium, Itanium 2

Observing historical data over several years, one can see that cache sizes increases following a clear step function. For Intel processors, increases occur every odd year and the average step duration is 3.5 years. There seems to be no obvious trend in AMD processors. From this observation, we model cache size as follows, simply assuming that the step duration (T_{step}) is twice as long as the step-up period (T_{up}).

$$\log_2(Cache) = \begin{cases} Cache_{base} + \left\lfloor \dfrac{release - yr_{base}}{T_{up}} \right\rfloor \\ \qquad\qquad or \\ Cache_{base} + \left\lfloor \dfrac{release - yr_{base}}{T_{up}} \right\rfloor - 1 \end{cases} \quad (4)$$

where yr_{base} is the base year, $release$ is the estimated release year, and $Cache_{base}$ is the cache size of the base year. This step function matches 80% of the cache size data points for Intel processors when the step-up period is 2, the step duration is 4, the base year is 1999.8, and the base cache size is 8 (256 KB). Meanwhile, this function matches 77% of the cache size data points for AMD processors with the same parameter values as Intel processors.

This model has time-dependent terms, so it can be used for future systems as well as current systems. Based on equation (4), we expect that the possible cache sizes for future processor architectures are 512KB or 1MB for the foreseeable future. For existing systems, however, we can determine the exact cache size of a processor by referencing processor datasheets.

2.4 Number of Processors per Node

Multiprocessor systems can provide cheap inter-processor communication and exploit lightweight parallelism through threads. Although more processors deliver more computing power, the dual-processor configuration is widely accepted as the most cost effective since cost and system complexity increase with the number of processors hyper-linearly. This is seen in our sample set. Figure 6 counts the cluster systems according to the number of processors per node in a log scale. As expected, clusters with dual-processors are dominant and account for about 70 percent of the samples. This histogram can be fitted by a normal distribution as follows.

$$\log_2(Count_{SMP}) \sim N(\mu, \sigma) \quad (5)$$

The samples passed a normality test for goodness-of-fit and showed a normal distribution in which the mean is 0.98 and the

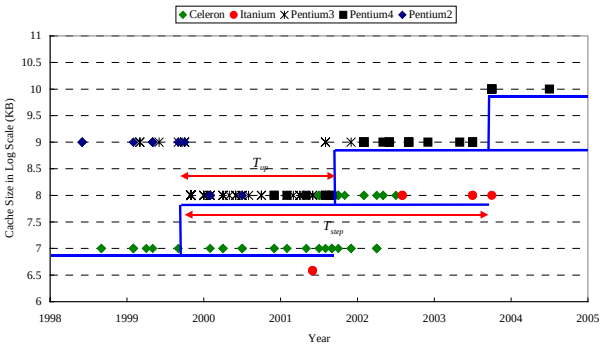

Figure 4. L2 cache size of Intel processors in a log scale (KB)

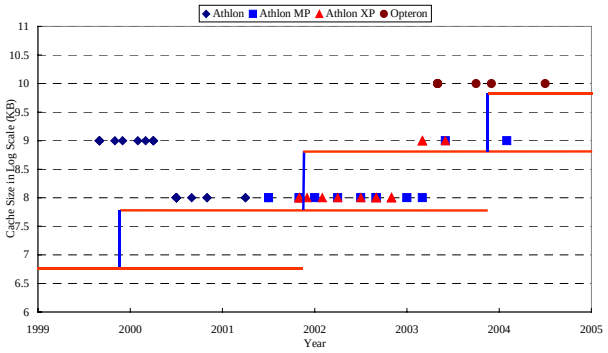

Figure 5. L2 cache size of AMD processors in a log scale (KB)

standard deviation is 0.55. We should note that this distribution is only applied to clusters for high-performance computing. A survey about hosts in a Desktop Grid environment managed by Entropia [23] shows quite different results as most desktop systems have a single processor.

Since this trend is independent of processor architecture, this model can be applied to any processor even though high-end microprocessors like Itanium and Opteron target server systems with more than 2 processors. In addition, since this trend seems time-independent, we expect that it will continue for the foreseeable future.

2.5 Memory Size per Processor

Memory size is a critical performance element for a wide range of applications, particularly for data-intensive Grid computing. For these applications, memory, network, and disk performance directly impacts Grid performance. Because the price of memory has fallen steadily with time, it is reasonable to correlate higher performance processors with larger memory configurations. To develop an appropriate rule, we surveyed the vendor specifications for minimum memory configurations. IBM xSeries machines with Pentium 4 processors have a minimum of 256MB, Xeon processors have a minimum of 512MB, and Itanium processors have 2GB memory. Meanwhile, HP desktops and workstation with Pentium 4 and Athlon XP processors have 128MB and those with Itanium processors have a minimum of 2GB.

From these observations, we hypothesize that memory size increases linearly with the number of processors per node and geometrically with processor performance. Applying these correlation factors to the installed memory sizes of the sample set, we get the histogram shown in Figure 7. This histogram can be fitted by a normal distribution as follows:

$$\log_2(Size_{memory}/(Count_{SMP} * 2^{F_P})) \sim N(\mu, \sigma) \quad (6)$$

where $Count_{SMP}$ denotes the number of processors per node and 2^{F_P} denotes the correlation due to the processor performance factor presented in Table 2. These factors are determined by the performance of floating-point operations per second. The samples passed a normality test and showed a normal distribution in which the mean value is 8.41 and the standard deviation value is 1.00. Since the time-dependent part is hidden under the relative performance factor, this model can be used for both current and future systems.

2.6 Disk Capacity per Node

Many Grid applications produce large amount of data and store it to disk. The local disk capacity and the I/O rate are critical to this kind of application. Regardless of its importance, disk information is rarely available: only 34 of the systems in our sample set provided disk information and vendor specifications give only limited guidelines. As an alternative, we use the well-known rule of thumb that disk capacity doubles every year [24]. We also assume that most cost-effective disks are installed. For example, 120GB EIDE hard drives were the most cost effective at the end of 2003 and 250 GB drives are available in the market as of April 2004. Meanwhile, 36GB SCSI drives were popular at the end of 2003 and 74GB and 150GB drives are in widespread use now. We assume that only a single disk is installed per node. In the past, a single disk capacity was not sufficient to store a large amount of data, so many systems had multiple disks. However, this is increasingly uncommon and most systems have only one disk (although they may have multiple disks to support RAID). For a given year, we assume that disk size follows a uniform distribution. From these assumptions, we can formulate disk size of system as follows.

$$Disk = Disk_{(type, yr_{base})} * 2^{(yr - yr_{base})} \quad (7)$$

where $Disk_{(type, yr_{base})}$ represents the base size of a drive type at year yr_{base}; for EIDE, it is 120 at 2003 and for SCSI, 74 at 2003. We assume that the EIDE and SCSI are chosen with same probability. We get the release year and month information of a given system from the clock speed trend.

2.7 Number of Nodes per Cluster

Cluster users have a keen interest in the number of processors available for their parallel applications. Because of the increasing cost, there are fewer cluster systems with larger numbers of cluster nodes. This intuition suggests a power-law model. However, as shown in Figure 8, the distribution observed in our sample set is quite different. Actually, the distribution of the number of nodes in a log scale is close to a normal distribution and it can be modeled as follows:

$$\log_2(Count_{node}) \sim N(\mu, \sigma) \quad (8)$$

This distribution supports the notion that cluster sizes that range from 8 to 64 nodes are preferred. The samples passed a normality test and showed a normal distribution in which the

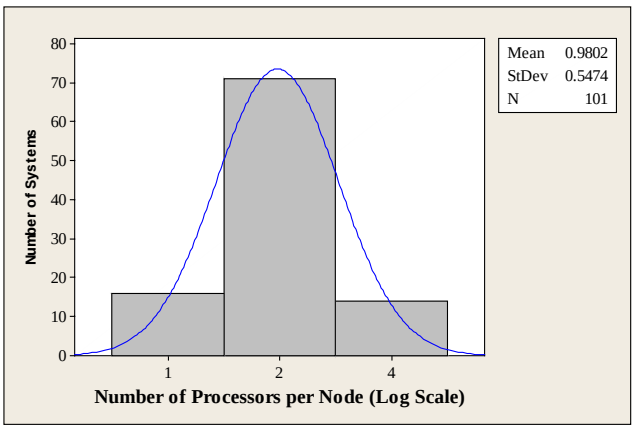

Figure 6. Distribution of number of processors per node in a log scale

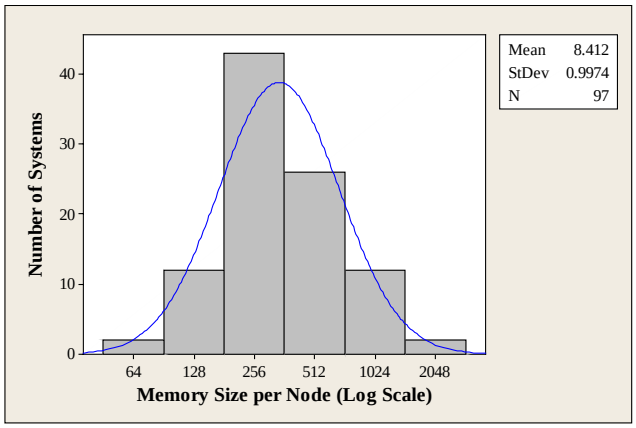

Figure 7. Distribution of memory size per node in a log scale applying the correlation factors due to processor architecture and number of processors per node (MB)

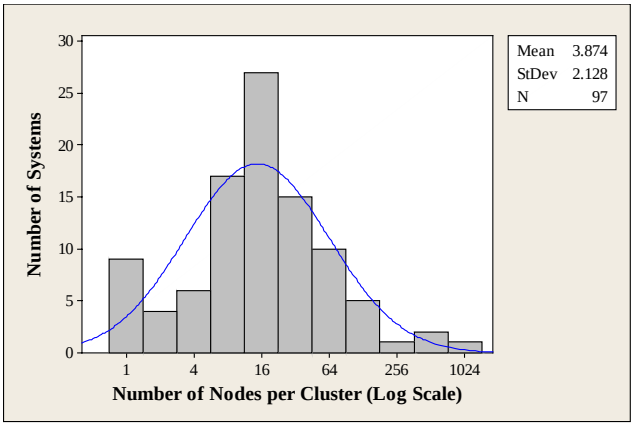

Figure 8. Distribution of the number of nodes per cluster in log scale

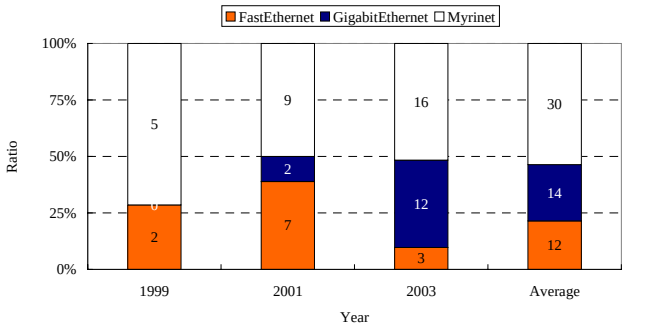

Figure 9. Distribution of cluster System Area Network (SAN): the numbers in the boxes represent the number of systems with that SAN type.

mean is 3.87 and the standard deviation is 2.13. According to our observation of the samples, there is no correlation between the number of nodes and the year of release even though the total number of processors increases. We believe that this model is robust and can be applied to future systems.

2.8 System Area Network

In addition to the computing power, another important attribute of clusters is their internal communication performance. Clusters with high performance communication often use specialized System Area Network (SAN) technology. In many parallel-processing applications, latency is as important as bandwidth, so lightweight communication networks like Myrinet [25] and VIA [26] have been adopted. However, Gigabit Ethernet products are cheap and powerful, so many cluster systems are deployed with a single or even dual Gigabit Ethernet network. For simplicity, we categorize SANs into Ethernet and non-Ethernet network types and model them using a binomial distribution. However, SAN information is available in only 56 clusters in our sample set and more samples would be required to build a more rigorous model.

To understand the current distribution, we calculated the ratio of Ethernet to non-Ethernet for our sample set. For 56 samples, Ethernet accounts for 46% and Myrinet for 54%. This result suggests that Ethernet and Myrinet evenly share the market. However, to build a model that can extrapolate future configurations, we need to know the trend over time. We count the number of systems released every two years and the results are shown in Figure 9. The graph says that the ratio of Fast Ethernet decreases and the ratio of Gigabit Ethernet increase. However, the ratio of Myrinet to Ethernet has been almost constant for the last four years. From this observation, we expect this trend to continue and the sample ratios can be used for the probabilities of a binomial distribution.

3. MODEL VALIDATION

We use a second sample set to validate our resource models. This second sample set is from NPACI Rocks clusters [27]. The Rocks software provides cluster configuration management. As of March 2004, there were 191 registered Rocks clusters totaling 10,073 processors. Because of its recent release and popularity, the population of Rocks clusters is a good representative of recent cluster configurations. Unfortunately, because the Rocks registry only includes a subset of modeled attributes (but arguably the most critical ones, at least for compute intensive applications), we can only validate our models for processor architecture, clock speed, and total number of nodes.

Table 3. Distribution of processor architectures in Rocks clusters and 1 year predicted samples (%)

	Pentium2	Celeron	Pentium3	Pentium4	Itanium	Athlon	AthlonMP	AthlonXP	Opteron
Rocks	0.7	0.0	18.8	57.9	1.1	0.2	13.4	2.0	5.8
This year	0.0	0.5	15.1	53.5	6.3	0.0	19.4	2.1	3.2

3.1 Processor Architecture

Since several models developed in this paper are dependent on the processor architecture distribution, an accurate model for it is essential. The processor architecture distribution for Rocks clusters is shown in the first row of Table 3. The major difference between the sample set (the "training samples") used to develop our models and the Rocks cluster sample set is the proportions of Pentium 3 and Pentium 4 processors. Pentium 3 and Pentium 4 account for 40.3% and 34.6% in the training samples and they account for 18.8% and 57.9% in Rocks clusters. The major cause of this difference is that most Rocks clusters were released in the past two years. When the processors released in only the last two years are considered in our sample set, however, the extrapolated model for this year is well matched to those derived from Rocks. As shown in the last row of Table 3, Pentium 3 and Pentium 4 account for 15.1% and 53.3% in the extrapolated samples.

Although there are discrepancies in the ratios for each processor, the overall trend is quite similar. Let's look at the ratio of Intel processors to AMD processors first. The ratio of Intel to AMD in the training samples is 84 to 16 while that in Rocks clusters is 79 to 21. Roughly, we can say that the ratio of Intel to AMD is 8 to 2 in both sets. Another aspect is the ratio of 32-bit architecture to 64-bit architecture. The percentage of 64-bit architectures in the training samples is about 6% while that in Rocks clusters is about 7%. When we consider the systems released in the past two years, the gap between two sample sets becomes significantly lower. In summary, the multinomial distribution for processor architecture and the sample ratios derived from the training samples are representative and form a sound basis for a model.

3.2 Processor Clock Speed

Since Rocks clusters were deployed recently, higher clock speed processors are dominant. To understand the distribution of clock speeds, we counted the number of Athlon MP/XP and Pentium 4 processors released in the last two years. Among 191 Rocks clusters, about 40 clusters have Athlon MP/XP processors and about 70 clusters have Pentium 4 processor. The distribution is shown in Figure 10. Athlon MP/XP processors at 1.6 Ghz and Pentium 4 processors at 2.8 Ghz are responsible for the spikes. A goodness-of-fit test using mean square error says that the error is too large to match this distribution to a uniform distribution. The major reason of this discrepancy is that we oversimplified the model. Thus, it may still possibly be a uniform distribution. Since these processors are in production and new systems with higher clock speed will be released continuously, we expect that the spikes will diminish and the overall trend will be close to a uniform distribution.

3.3 Number of Processors per Cluster

In Section 2.7, we developed a model for the number of nodes per cluster. However, only the number of processors per cluster is available in the Rocks cluster lists. As most systems have dual-processor nodes and the number of nodes in a cluster follows a normal distribution, we can assume that the number of processors also follows a normal distribution. Figure 11 shows the distribution of the number of processors in a log scale for the training samples and the Rocks clusters. Each histogram can be fitted by a normal distribution and both passed a normality test. The number of processors for the training samples follows a normal distribution in which the mean is 4.86 and the standard deviation is 2.00 while the Rocks clusters follows a normal distribution in which the mean is 4.50 and the standard deviation is 1.78.

To answer the question, "Is the training sample set equivalent to the Rocks cluster sample set?" we performed an independent sample t-test and an f-test. An f-test is used to test if the standard deviations of two groups are equal. The formula for f-test is a ratio between the standard deviations of the two groups. The test result at the 5% significance level indicates that there is no evidence that the deviations of the two sets are different. Then, we performed a t-test under the assumption that the standard deviations are equal. A t-test assesses whether the means of two groups are statistically different from each other. This analysis is appropriate whenever one want to compare the means of two groups. The formula for t-test is a ratio of the differences between two group means to variability of groups. The test result with the 5% risk level also indicates that there is no evidence that the means are different. Even though these tests do not guarantee that both sets are same, the consistent results of two tests mean it is highly probable that two groups are statistically the same.

4. IMPLEMENTATION

In the previous sections, we developed resource models and validated them. In this section, we discuss the implementation of a resource generator that implements these models. The resource generator consists of three components: a processor architecture description file, a training sample file, and a modeler. The processor architecture description file contains the data available in processor datasheets, which include processor architecture name (e.g. Pentium 4, Itanium), processor clock speed, L2 cache size, and current availability. It is mainly used to determine the values of the model parameters for clock speed and cache size. In addition, it is used to check if the clock speed and cache size of the training samples and synthesized resource entities are valid. For instance, it can detect that Pentium 3 processor with 2.0 Ghz clock or Itanium 2 processor with 1 MB L2 cache are invalid. Users can add new processor data whenever they become available.

To build models for resource entities, the resource generator reads representative data for clusters. This training sample file contains the configurations of our sample clusters (over 10,000 processors). This architecture allows the models to be updated seamlessly as more cluster data becomes available. Finally, the modeler implements the models for each resource attribute as we have described at length and determines the model parameters by

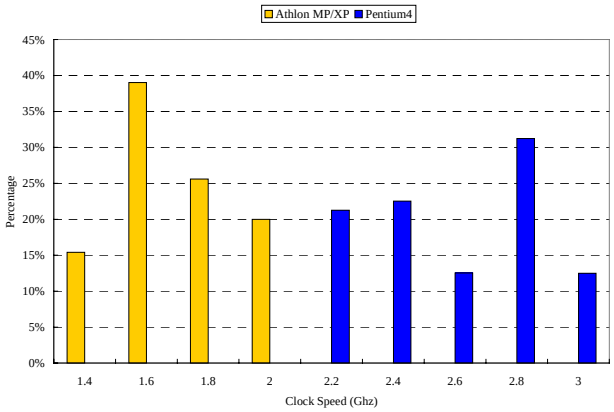

Figure 10. Distribution of clock speed for Rocks clusters released in the past two years (%)

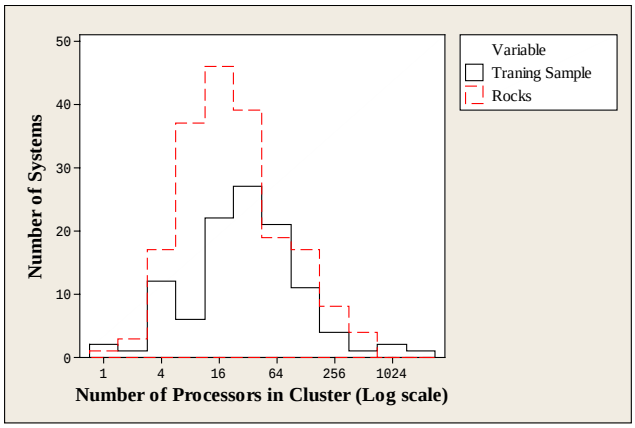

Figure 11. Distribution of the number of processors per cluster for the training samples and Rocks clusters

referencing the architecture description and the training sample file. It performs a linear regression analysis, implements the uniform and normal distribution models, and determines the model parameters for the resource entities.

5. RELATED WORK

Most closely related to ours is the work by Lu and Dinda's grid resource generator, GridG [21]. GridG synthesizes resource information through two steps: topology generation and node annotation. In the topology generation step, GridG produces structured network topologies obeying Internet power laws using Tier. Our model presumes a similar technique. GridG then annotates the nodes in the network with the attributes of computing resource entities according to the user supplied rules and empirical resource distribution information. In the annotation step, they captured two correlations between computing resource entities. The first correlation can be used to capture dependencies between number of processors, clock speed, memory size, and disk size. The second correlation, which the authors call the OS concentration, can be used to model the typical phenomenon of a dominant operating system in a local area network.

Their observation of these correlations is valuable to understand the characteristics of computing resources in the computational Grid environment. However, they focus only on the procedure of resource synthesis and do not evaluate the accuracy of their rules. Hence, there is little evidence that the synthesized resource configurations are representative; only that impossible configurations are eliminated. For example, our study shows that their argument that the memory size is proportional to the number of processors is correct, but they missed another factor of processor performance. In addition, OS concentration does not fully capture the fact that the Grid consists primarily of clusters, not individual hosts. Finally, GridG does not support any way to predict configurations for future grid systems.

Our resource generator focuses on clusters as the dominant resource type in Grids. In contrast to GridG, our resource generator uses a general statistical framework for modeling resource attributes, allowing uniform treatment of a wide range of attributes and easy addition of future ones. In addition, the statistical approach allows automatic parameter tuning through the incorporation of new cluster data as well as extrapolation for future cluster configurations.

6. SUMMARY & FUTURE WORK

We have discussed models for the resources in computational Grids and a novel resource generator for the dominant resource in Grids, commodity clusters. Our major contributions are the development of realistic models for the resource entities in a computational Grid. Using a statistical approach, we capture the characteristics of a sample data set automatically and generate models that enable representative resource generation for contemporary and future clusters (and thereby grids). Using a second sample set for validation, we demonstrate that the models capture the distributions of current systems.

Some attributes of our models include the number of processors in a node, memory size per node, and the number of nodes per cluster. These follow a normal distribution. Meanwhile, processor architecture follows a multinomial distribution, clock speed follows a uniform distribution, cache size follows a step function, and intra-cluster networking is split evenly between Ethernet and Myrinet.

Due to the a paucity of data, however, our models for disk size per node and SAN type are not yet complete. In addition, we need more study about the distribution of clock speed. To develop more realistic and accurate models, we need more sample data. This is certainly an area of active continuing work, and we are now accumulating cluster configurations from various sources.

To evaluate the robustness of our models more deeply, it would be good to evaluate the models using a bootstrapping technique. This deeper validation would build on the work we have presented here, which validates the models using two disjoint, independent sample sets. As we collect more sample data, we pursue these further experiments.

Our resource generator has been fully implemented and is being used to develop a resource discovery system for a range of projects – the OptIPuter's DVC [28] and the VGrADS project. Moreover, it will be integrated with a variant of Brite network topology generator in MicroGrid [29] to perform various experiments in a virtual grid environment with respect to scalability, resource optimization, and performance prediction.

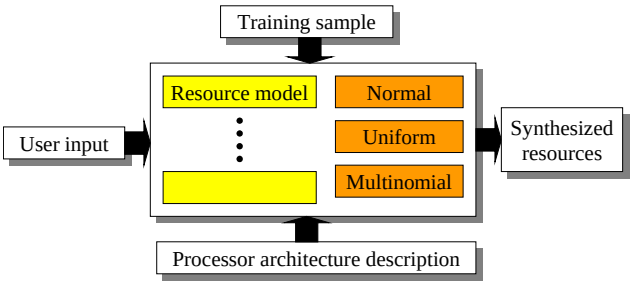

Figure 12. Resource generator structure

7. ACKNOWLEDGEMENTS

The authors and research described here are supported in part by the National Science Foundation under awards NSF EIA-99-75020 Grads and NSF Cooperative Agreement NSF CCR-0331645, NSF NGS-0305390, and NSF Research Infrastructure Grant EIA-0303622. Support from Hewlett-Packard, BigBangwidth, Microsoft, and Intel is also gratefully acknowledged. The Ministry of Information and Communication, Republic of Korea also supports this research study.

8. REFERENCES

[1] Charlie Catlett, The TeraGrid: A Primer, http://www.teragrid.org/about, Sep. 2002.

[2] William E. Johnston, Dennis Gannon, and Bill Nitzberg, Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid, In the Proceedings of IEEE International Symposium on High Performance Distributed Computing (HPDC'99), Aug. 1999

[3] Japans National Research Grid Initiative (NAREGI), Asian Technology Information Program Report, ATIP03.016, http://www.atip.org/REPORTSMATRIX/public/year2003_total.html, Dec. 2003.

[4] Geoffrey Fox and David Walker, e-Science Gap Analysis, UK e-Science Technical Report, UKeS-2003-0, http://www.nesc.ac.uk/technical_papers/uk.html, Jun. 2003.

[5] William E. Johnston, et. al, The DOE Science Grid, http://www.doesciencegrid.org/Grid/papers, 2003.

[6] J. Waldo, The Jini architecture for network-centric computing, Communications of the ACM, 42(7): 76-82, 1999.

[7] L. Gong, JXTA: A Network Programming Environment, IEEE Internet Computing, Vol. 5, pp. 88-95, June 2001

[8] Butterfly.net Inc., Butterfly Grid Solution for Online Games, http://www.butterfly.net, 2003.

[9] Oliver Storz, Adrian Friday and Nigel Davies: Towards 'Ubiquitous' Ubiquitous Computing: an alliance with the Grid, System Support for Ubiquitous Computing Workshop at the Fifth Annual Conference on Ubiquitous Computing (UbiComp 2003), Oct. 2003. Apr. 2003.

[10] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, Grid Information Services for Distributed Resource Sharing, In the Proceedings of IEEE International Symposium on High-Performance Distributed Computing (HPDC'01), Aug. 2001.

[11] Rich Wolski, Neil Spring, and Jim Hayes. "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," Journal of Future Generation Computing Systems, 15(5-6): 757-768, Oct. 1999.

[12] Federico D. Sacerdoti, Mason J. Katz, Matthew L. Massie, David E Culler, "Wide Area Cluster Monitoring with Ganglia," In the Proceedings of IEEE International Conference on Cluster Computing (Cluster'03), Dec. 2003.

[13] Todd Tannenbaum, HawkEye: A Monitoring and Management Tool for Distributed Systems, http://www.cs.wisc.edu/condor/hawkeye, Mar. 2002.

[14] Rajesh Raman, Miron Livny, and Marvin Solomon, Matchmaking: Distributed Resource Management for High Throughput Computing, In the Proceedings of IEEE International Symposium on High Performance Distributed Computing (HPDC'98), Jul. 28-31, 1998.

[15] Chuang Liu; Ian Foster, A Constraint Language Approach to Grid Resource Selection, TR-2003-07, Mar. 2003.

[16] D. Lu, P. Dinda, J. Skicewicz, Scoped and Approximate Queries in a Relational Grid Information Service, In proceedings of International Workshop on Grid Computing (Grid'03), pp.192-201, Nov. 2003.

[17] Peter Dinda and Dong Lu, Nondeterministic Queries in a Relational Grid Information Service, In the Proceedings of Supercomputing 2003 (SC'03), Nov. 2003.

[18] Song, H. J., Liu, X., Jakobsen, D., Bhagwan, R., Zhang, X., Taura, K., and Chien, A. A., The microgrid: a scientific tool for modeling computational grids, In the Proceedings of Supercomputing 2000 (SC'00), Nov. 2000.

[19] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: An Approach to Universal Topology Generation. In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'01), Aug. 2001.

[20] Doar, M. A better model for generating test networks. In the Proceedings of IEEE Global Internet, pp. 86-93, 1996.

[21] D. Lu and P. Dinda, Synthesizing Realistic Computational Grids, In the Proceedings of Supercomputing 2003 (SC'03), Nov. 2003

[22] Erich Strohmaier and Jack Dongarra, Highlights of the 23rd TOP500 List/Awards for the #1 System Worldwide and the #1 System in Europe, In the Opening Session of International Supercomputer Conference, Jun. 2004.

[23] Fran Berman, Geoffrey C. Fox, and Anthony J. G. Hey, Architecture of a Commercial Enterprise Desktop Grid: The Entropia System, Grid Computing: Making the Global Infrastructure a Reality, pp.337-350, Dec. 2002, John Wiley & Sons.

[24] D. A. Patterson and J. L. Hennessy, Computer Architecture - A Quantitative Approach, (second edition), Morgan Kaufmann, 1996

[25] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W.-K. Su. Myrinet- A Gigabit-per-Second Local-Area Network, IEEE Micro, 15(1): 29-38, Feb. 1995.

[26] Dave Dunning, Greg Regnier, Gary McAlpine, Don Cameron, Bill Shubert, Frank Berry, Anne Marie Merritt, Ed Gronke, Chris Dodd, “The Virtual Interface Architecture,” IEEE Micro, 18(2): 66-76, Mar./Apr. 1998.

[27] Philip M. Papadopoulos, Mason J. Katz, Greg Bruno, “NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters,” In the Proceedings of IEEE International Conference on Cluster Computing (Cluster'01), pp. 258-270, Oct. 2001.

[28] Nut Taesombut and Andrew Chien, Distributed Virtual Computer (DVC): Simplifying the Development of High Performance Grid Applications, In the Proceedings of workshop on Grids and Advanced Networks (GAN ’04), Apr. 2004.

[29] Xin Liu, Huaxia Xia, and Andrew Chien, Validating and Scaling the MicroGrid: A Scientific Instrument for Grid Dynamics, To appear, Journal of Grid Computing.