Knowledge-Based Modeling Approach for Performance Measurement of Parallel Systems

Amit Chhabra and Gurvinder Singh
Department of Computer Science & Engineering, Guru Nanak Dev University, India

Abstract: *Parallel systems are important computing platforms because they offer tremendous potential to solve inherently parallel and computation intensive applications. Performance is always a key consideration in determining the success of such systems. Evaluating and analyzing parallel system is difficult due to the complex interaction between application characteristics and architectural features. Traditional performance methodologies like experimental measurement, theoretical/analytical modeling and simulation naturally apply to the performance evaluation of parallel systems. Experimental measurement uses real or synthetic workloads, usually known as benchmarks, to evaluate and analyze their performance on actual hardware. Theoretical/analytical models try to abstract details of a parallel system. Simulation and other performance monitoring/visualization tools are extremely popular because they can capture the dynamic nature of the interaction between applications and architectures. Each of them has several types. For example, experimental measurement has software, hardware, and hybrid. Theoretical/analytical modeling has queueing network, petri net, etc. and simulation has discrete event, trace/execution driven, Monte Carlo. All of them have their own advantages and disadvantages. The first part of this paper will concentrate on identifying parameters for carrying out a comparative survey on these techniques and second part will justify the need for some kind of modelling approach which combines the advantages of all the three performance evaluation techniques and lastly paper will be focusing on an integrated model combining all the three techniques and using knowledge-based systems to evaluate the performance of parallel systems. This paper also discusses certain issues like selecting an appropriate metric for evaluating parallel systems; need to select proper workload and workload characterization.*

Keywords*: Knowledge-based system, integrated modelling, metrics, parallel systems, performance evaluation.*

Received May 4, 2007; accepted October 16, 2007

1. Introduction

Generally Performance is always a measure of “how well” a system or constituent component accomplishes its assigned task. There must be some quantitative weightage to the phrase “how well” in order to measure the performance of the parallel system. So performance evaluation can be defined as assigning quantitative values to the indices of the performance of the system under study. There are many aspects that are need to be taken care of while evaluating performance of parallel systems and those are selecting appropriate metric for evaluating parallel system, selecting proper workload, workload characterization and lastly performance evaluation techniques. This paper proceeds in the similar fashion as discussed above.

2. Role of Performance in Parallel Systems

Performance is always a key factor in determining the success of parallel system. Quantitative evaluation and modelling of hardware and software components of parallel systems are critical for the delivery of high performance. Performance studies apply to initial design phases as well as to procurement, tuning, and capacity planning analyses. As performance cannot be expressed by quantities independent of the system workload, the quantitative characterization of resource demands of applications and of their behaviour is an important part of any performance evaluation study. Among the goals of parallel systems performance analysis are to assess the performance of a system or a system component or an application, to investigate the match between applications requirements and system architecture characteristics, to identify the features that have a significant impact on the application execution time, to predict the performance of a particular application on a given parallel system, to evaluate different structures of parallel applications.

3. Selecting an Appropriate Performance Metric

To study the performance of systems or to compare different systems for a given purpose, we must first select some criteria. These criteria are often called metrics in performance evaluation. Different situations need different sets of metrics. Thus, selecting metrics are highly problem oriented. To know metrics, their relationships and their effects on performance parameters is the first step in performance studies. Different metrics may result in totally different values.

Hence, selecting proper metrics to fairly evaluate the performance of a system is difficult.

Characteristics or properties of a good performance metric should be SMART, i.e., specific, measurable, acceptable, realizable and thorough. So far the researchers have proposed many parallel metrics. Execution time is the time interval between the beginning of parallel computation and the time since the last processing elements finishes execution. Another parallel metric is *speedup. Speedup* is defined as the ratio of the time taken to execute a problem on a single processor to the time required to solve the same problem on a parallel computers with *p* identical processing elements. Amdahl and gustafson have proposed a law for speedup metric. The formulation for speed up using Amdahl's law is based on fixed problem size and speed up drops very rapidly with the increase in sequential fraction. So fixed load acts as deterrent in achieving the scalability in performance.

Other variants of speedup are relative speedup, real speedup, absolute speedup and asymptotic relative speedup.

Sun and Ni have generalized Gustafson's scaled speedup to fixed-time relative speedup and proposed another speedup model called memory-bounded relative speedup. In memory constrained sealing, the problem is made to fit in the available memory. Sun and Gustafson proposed two new speedup measures; one of them is generalized speedup and other is sizeup. Generalized speedup is the ratio of parallel execution speed to sequential execution speed. It has been observed that when cost factor is same for all kinds of work, generalized speedup equals relative speedup. The sizeup is the ratio of the serial work represented by the adjusted instance to the serial work represented by the unadjusted instance.

Another parallel metric is efficiency. This measure is very close to speedup. It is the ratio of speedup to the number of processors *P*. Depending on the variety of speedup used; one gets a different variety of efficiency. Carmona and Rice define efficiency as the ratio of Work Accomplished (WA) by a parallel algorithm and the Work Expended (WE) by the algorithm. Also there is one another variant of efficiency known as *incremental efficiency*. The *incremental efficiency* metric [7] looks at the ratio of the successive efficiencies for different number of processors, a quantity that tends to unity in the limit.

Another performance measure is scalability. Scalability refers to the change in the performance of the parallel system as the problem size and machine size increases. Kumar, Nageshwara and Ramesh proposed a scalability measure based on efficiency. In this scalability, or isoefficiency, of a parallel system is defined to be the rate at which workload must increase relative to the rate at which the number of processors is increased so that the efficiency remains the same. Depending on the version of efficiency, different varieties of isoefficiency (e.g., real isoefficiency, absolute isoefficiency and relative isoefficiency) can be obtained. Sun and Rover proposed isospeed that uses the average workload per processor needed to sustain a specified computational speed as a measure of scalability.

Utilization is another measure for evaluating resource utilization and In addition to the above metrics, some other general measures such as CPU time, and CPI (clock cycles for instruction) play important role in measuring the success of parallel implementation of problem.

4. Selecting Proper Workload and Workload Characterization

A system is often designed to work in a particular environment with some specific workload. So it is not a wise step to study the performance of a system without taking into account the workload. Selecting proper workloads is important step in performance evaluation. So after selecting an appropriate metric for evaluation of parallel system, next step is to choose proper workload.

The workload of a computer system is the demand of user from the system being designed. A real workload is often very complex and unrepeatable, so it cannot be used properly for studies. Because of this, a test workload model must be designed. This model must have the following characteristics.

- It must be a representation of the real workload. That is the static and dynamic behaviour of the real workload must be accurately captured. It must be easily reproduced and modified such that the workload can be used repeatedly in different studies;
- It must be compact, such that the workload can be easily ported to different systems. This is quite useful in comparing different systems for the same purpose.

Examples of workload generation methods are instruction mix, synthetic programs like NFSStone and IOStone and various application benchmarks like LINPACK, *etc.*

Then next comes the workload characterization, which is the process of developing a workload model that can be used repeatedly. In order to design a workload model, it is essential to carefully study and understand the key characteristics and to know the limitations of the model. Many statistical analysis techniques can be used in workload characterization like averaging, distribution analysis, principal component analysis, clustering and markov models.

5. Performance Evaluation Techniques

Performance evaluation can be defined as assigning quantitative values to the indices of the performance of the system under study. Evaluating and analyzing parallel system is difficult due to the complex interaction between application characteristics and architectural features. Performance evaluation techniques can be classified into three categories; Experimental measurement, theoretical/analytical modelling and simulation. In this section all of these three techniques are discussed and compared with each other on the basis of some parameters. Four major parameters for selecting a technique for performance evaluation are:

- Stage: this parameter examines which performance evaluation technique should be used at what stage of system development life cycle.
- Output measures: this parameter examines the capabilities of the technique towards providing the desirable metrics.
- Accuracy: this factor evaluates the validity and reliability of the results obtained from the technique.
- Cost/effort: this parameter investigates the cost and effort invested in each performance evaluation strategy in context with computer and human resources.

Various other parameters for selecting a technique for performance evaluation are:

- Resource consumption: this parameter examines the amount of resources consumed/ required by a particular performance evaluation technique.
- Time consumption: this parameter examines the amount of time consumed/ required by a particular performance evaluation technique.
- Tools required: this parameter examines the type of tools required for implementation of any particular performance evaluation technique.
- Trustability/ believability: these parameters reveals that how much one can trust on the results of a particular performance evaluation technique.
- Scalability complexity: this parameter examines the complexity involved in scaling a particular performance evaluation technique.
- Flexibility: this parameter examines the flexibility of a particular performance evaluation technique towards adapting the modifications made to the model of evaluation technique and checking their effect.

5.1. Experimental Measurement

It is based on direct measurements of the system under study using a software, hardware or/and hybrid monitor. It uses real or synthetic workloads and measures their performance on actual hardware. As it uses the actual hardware and the related system software to conduct the evaluation, making it the most realistic model from the architectural point of view. A monitor is a tool, which is used to observe the activities on a system. In general, a monitor performs three tasks: data acquisition, data analysis, and result output. The data recorded by a monitor include hardware related data, e.g. processor usage throughout program run, message latency, and software data, e.g. process times, buffer usage, load balancing overhead. Traditionally, monitors are classified as software monitors, hardware monitors, and hybrid monitors. Software monitors are made of programs that detect the states of a system or of sets of instructions, called software probes, capable of event detection. Hardware monitors are electronic devices to be connected to specific system points where they detect signals characterizing the phenomena to be observed.

A hybrid monitor is a combination of software and hardware. Many examples of software monitors can be found in the literature [6, 8, 9]. Examples of hardware monitors are COMTEN and SPM [5]. Examples of hybrid monitors are diamond and HMS.

Each class of monitors has its own advantages and disadvantages. Selecting an appropriate monitor involves various aspects, e.g., cost, overhead, accuracy, availability, information level, etc. In general, hardware monitors have received less attention than software monitors. This has been shown by the fact that the numbers of existing software monitors are far greater than that of hardware monitors. Application suites such as the perfect club [4], the NAS Parallel Benchmarks [11], and the SPLASH application suite have been proposed for the evaluation of parallel machines. Parameters under consideration:

- Stage: experimental measurement uses the actual hardware, so it cannot be used at the early stage of system design. It can be only possible when the system design has been completed and a real system is being available for evaluation. So experimental measurement is only possible when actual system under study is available.
- Output measures: experimental measurement can give the overall execution time of the application on particular hardware platform. Conducting the evaluation with different problem sizes and number of processors would thus helps to calculate metrics such as speedup, scaled speedup, sizeup, isoefficiency.
- Accuracy: experimental measurement uses real applications to conduct the evaluation on actual machines giving accurate results. But external factors like external instrumentation and monitoring intrusion can affect the accuracy of results.
- Cost/effort: substantial costs are involved in developing the model for experimentation as this technique uses the actual real system to give results. This cost is directly dependent on the complexity of the application. Once the model for experimental measurement is developed, the cost for conducting

the actual evaluation, that is the execution time of the given application, may be very small. Modifications to the application and hardware can thus be accommodated by the experimentation technique at a cost that is totally dependent on the type and the amount of modifications. In other words scalability can be complexier but it totally depends how much one wants to scale.

- Resource consumption: this technique requires actual instruments, so resource consumption will be high.
- Time consumption: as experimentation requires actual system setup which consumes heavy amount of time but once the system comes into reality, time required for results may be very less.
- Tools required: actual instruments are required here to evaluate any parallel application
- Trustability/believability: as it uses actual hardware so results given by it can be highly trustworthy. Here atleast simulation or theoretical/analytical modelling could validate results.
- Scalability complexity: here complexity is totally dependent on the fact that how much scaling of the system is required and this scaling requires time and money.
- Flexibility: it is not highly flexible because it takes lot of time to change a particular experimental setup and it takes lot of time to check the effect of this change.

• Theoretical/Analytical Modelling

Performance evaluation of parallel systems is hard due to the several degrees of freedom that they exhibit. Analytical and theoretical models try to abstract details of a system, in order to limit these degrees of freedom to a tractable level. Such abstractions have been used for developing parallel algorithms and for performance analysis of parallel systems.

Abstracting machine features by theoretical models like the PRAM [2, 3] has facilitated algorithm development and analysis. These models try to hide hardware details from the programmer, providing a simplified view of the machine. Several machine models like Bulk Parallel Random Access Machine (BPRAM), Module Parallel Computer (MPC), Valiant introduces the Bulk-Synchronous Parallel (BSP) model and Log P have been proposed over the years to bridge the gap between the theoretical abstractions and the hardware.

While theoretical models attempt to simplify hardware details, analytical models abstract both the hardware and application details in a parallel system. Analytical models capture complex system features by simple mathematical formulae, parameterized by a limited number of degrees of freedom that are tractable. Such models have found more use in performance analysis than in algorithm development where theoretical models are more widely used. As with experimentation, analytical models have been used to evaluate overall system performance as well as the performance of specific system artifacts. Vrsalovic *et al.* develop an analytical model for predicting the performance of iterative algorithms on a simple multiprocessor abstraction, and study the impact of the speed of processors, memory, and network on overall performance. Parameters under consideration:

- Stage: theoretical/analytical modelling can be used at the early design phase of system development life cycle, as it does not require any actual hardware for evaluation. These models try to hide hardware details from the programmer, providing a simplified view of the machine and also behaviour of these models is very close to that of actual machine.
- Output measures: theoretical/analytical models can directly present statistics for system overheads that are modeled. The values for the hardware and the workload parameters can be plugged into the model corresponding to the system overhead. With models available for each system overhead, overall execution time and all other metrics can be calculated. The drawback is that each system overhead needs to be modeled or ignored in calculating the execution time.
- Accuracy: theoretical/analytical models are useful in predicting system performance and scalability trends as parameterized functions. However, the accuracy of the predicted trends depends on the simplifying assumptions made about the hardware and the application details to keep the models tractable. Theoretical models can use real applications as the workload, whereas analytical models represent the workload using simple parameters and probability distributions. Thus the former has an advantage over the latter in being able to estimate metrics of interest more accurately. But even for theoretical models, a static analysis of application code, which is used to estimate the running time, can yield inaccurate results.
- Cost/effort: substantial effort is involved in the development of models for theoretical/analytical modelling. Simple modifications to the application and hardware can be easily handled with these models by changing the values for the corresponding parameters and re-calculating the results. But a significant change in the hardware and application would demand a re-design of the input models, which can be expensive.
- Resource consumption: not much resources are required to build models for theoretical/analytical modelling.
- Time consumption: fair amount of time is consumed for developing models of

Theoretical/analytical modelling but once models have been developed it takes little time to get results.

- Tools: it involves analyst and various analysis tools.
- Trustability/believability: this method involves assumptions that are being made while developing models. So they are not much trustworthy until there results are validated by atleast simulation or experimental measurement.
- Scalability complexity: it takes time to scale these kinds of models.
- Flexibility: it is flexible as changes can be made to the models easily and effect of change can also be checked easily.

○ Simulation

Simulation is a widely used technique in performance evaluation. It provides a powerful way to predict performance before the system under study has not actually been implemented. It can also be used to validate analytical models. There are a variety of simulations presented in the literature: emulation, Monte Carlo simulation, trace-driven simulation, execution-driven simulation and discrete-event simulation.

An example of emulation is using one available processor (host processor) to emulate the instruction set of another processor (target processor) that is not available or under design. This type of simulation is sometimes called by some author's instruction-level simulation or cycle-by-cycle simulation.

Monte Carlo simulation is a static simulation where the simulated systems do not change their characteristics with time. Computer systems are dynamic systems, and do not belong to this category.

A trace-driven simulation system consists of two components: an event generator (or trace generator) and a simulator. The event generator produces a trace of execution events, mostly addresses, which are used as input to the simulator. The simulator consumes the traced data and simulates the target architecture to estimate the time taken to perform each event on the architecture under study. Discrete-event simulation is used to simulate systems that can be described using discrete event models. Discrete-event simulation is very well suited for studying queuing systems. Parameters under consideration:

- Stage: simulation can not be used at very early stage of system design because of the non-availability of required system details at that point but as the design process goes on and more details about the system are obtained, this technique becomes powerful tool at that point.
- Output measures : simulation provides a convenient monitoring environment for observing details of parallel system execution, allowing the user to accumulate a range of statistics about the application, the hardware, and the interaction between the two. It can give the total execution time and all other metrics discussed earlier.
- Accuracy: the accuracy of results depends purely on the accuracy of input models. Execution-driven simulation can faithfully simulate all the details of a real-world application. It is also possible to simulate all the details of the hardware, though in many circumstances a level of abstraction may be chosen to give moderately accurate results for the intended purposes. The accuracy of these abstractions may also be validated by comparing the results with those obtained from a detailed simulation of the machine or an experimental evaluation on the actual machine.
- Cost/effort: the main disadvantage associated with simulations is the cost and effort involved in simulating the details of large parallel systems. With regard to modifiability, plugging in these values into the model and re-simulating the system may handle a moderate change in hardware parameters. But such a re-simulation, as has been observed, is invariably costlier than a simple re-calculation that is needed for analytical models, or experimentation on the actual machine. A significant change in the machine or application details would also demand a re-implementation of the simulation model, but the cost of re-simulation is again expected to dominate over the cost of re-implementation.
- Resource Consumption: it requires small amount of resources as we are just simulating the parallel environment, no actual instrumentation is being required.
- Time consumption: fair amount of time is consumed for developing the simulation model. But once model has been developed it takes little time to get results.
- Tools: simulation requires high level computer programming languages to build and develop the model.
- Trustability/believability: simulated model is just a replica of the final actual machine but results may little bit vary as this is not an actual machine.
- Scalability complexity: simulated models are very easy scalable as it artificially simulates the actual environment.
- Flexibility: the main advantage of simulation is its flexibility. One can make various modifications to the simulation model and check their effect easily.

• Need for Integrated Modelling of Performance Evaluation Techniques

Each performance evaluation technique has its own role to play in evaluating parallel systems. As shown in Table 1 each technique has its own advantages and disadvantages. So effort can be made in developing an integrated model, which combines the advantages of all the three techniques. This integrated model would have the advantage that it benefits the realism and accuracy of experimentation in evaluating large parallel systems, the convenience and power of theoretical/analytical models in predicting the performance and scalability of the system as a function of system parameters and the accuracy of detailed statistics provided by execution-driven simulation and avoids most of their drawbacks. Also this need for an integrated model is justified by the three rules of validation that says:

- Do not trust the results of a simulation model until they have been validated by at least Theoretical/analytical model or experimental measurements.
- Do not trust the results of a theoretical/ analytical model until they have been validated by at least simulation or experimental measurements.

Table 1. Showing comparison of performance evaluation techniques.

	Performance Evaluation Techniques		
Characteristic	Theoretical/Analytical Modelling	Simulation	Experimental Measurement
a) Stage	Any	Any	After prototype is developed
b) Output Measures	It can give total execution time and all other metrics discussed earlier	It can also give total execution time and all other metrics discussed earlier	It can also give total execution time and all other metrics discussed earlier
c) Accuracy	Low	Medium	High
d) Cost/ Effort	Low	Medium	High
e) Resource Consumption	Low	Medium	High
f) Time Consumption	Low	Medium	High
g) Tools	Analysts	Computer Programming Languages	Instrumentation
h) Trustability/	Low	Medium	High
i) Scalability Complexity	Low	Medium	High
j) Flexibility	High	High	Low

- Do not trust the results of an Experimental measurement model until they have been validated by at least Simulation or Theoretical/analytical modelling.

7. Proposed Integrated Model Using Knowledge Based Systems

This integrated model shown in Figure 1 is going to use the power of stored performance knowledge base systems. User applications are being fed to experimental measurement technique whose stored performance knowledge base system is also attached with it as indicated in Figure 1. This integrated modelling is basically an extension to work done in literature [10].

Based on the user application, the type of workload it is using (real or synthetic) and any other current data, a knowledge set can be extracted from this knowledge base system which will suggest which experimental technique (software, hardware or hybrid) is best or optimal for the current user application. Also when results are obtained after a particular technique is applied on user application, these general results can in turn act as knowledge. This valuable knowledge (general results) can be updated back into the knowledge base system and this procedure of knowledge extraction and updation can be repeated. Knowledge base systems can also be attached with all other evaluation techniques (simulation and theoretical/analytical modelling). These stored knowledge base systems helps in choosing technique for evaluating current user application. Same procedure of knowledge extraction and updation is also true for other measurement techniques. Experimental measurement can be used to implement real-life applications on actual machines, to understand their behaviour and to extract interesting kernels that occur in them. These kernels are fed to an execution-driven simulator, which faithfully and successfully models the dynamics of parallel system interactions. The statistics that are drawn from simulation may be used to validate and refine existing theoretical/analytical models, and to even develop new models. The validated and refined models can help in abstracting details in the simulation model to enhance the speed of simulation. The validity of such a simulation model can in turn be verified by comparing the simulation results with those from an experimental evaluation on the actual hardware. Such a strategy combines advantages of all three techniques, uses the power of knowledge base systems and avoids the shortcomings of the individual evaluation techniques.

7.1. Building of Knowledge-Base

The knowledge base generally contains facts and rules about some specialized knowledge domain. In case of stored performance knowledge base of say experimental measurements it will contains rules and facts about experimental measurement. It is true for other techniques of performance evaluation, i.e., for simulation and analytical modelling. Building knowledge base [1] requires careful planning, accounting and organization of the knowledge structures. It also requires thorough validation and verification of the completed knowledge base operations, which have yet to be perfected. An intelligent editor can simplifies this knowledge base building process.

One of the most difficult tasks in building knowledge base systems is in acquisition and encoding of the requisite domain knowledge. Knowledge for such systems must be derived from expert sources and elicitation of right knowledge can take several men years and cost huge amount of money. This elicitation process can be taken care by knowledge engineers, which build systems by eliciting knowledge from experts, coding that knowledge in an appropriate form, validating the knowledge and ultimately constructing the system using variety of tools.

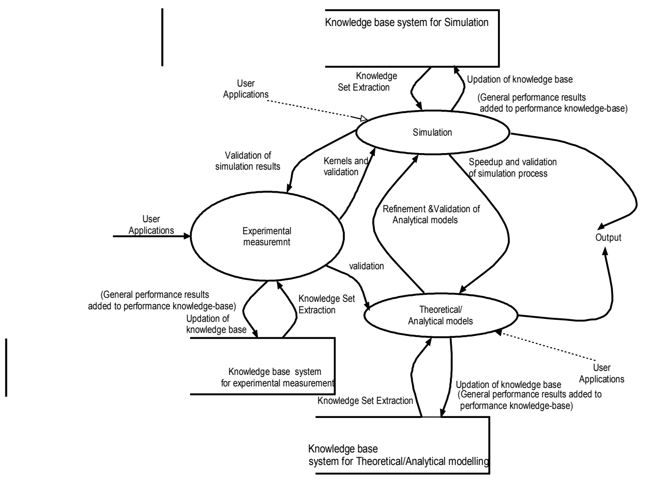
Knowledge base system for Simulation
Knowledge Set Extraction
Updation of knowledge base
(General performance results added to performance knowledge-base)
User Applications
Simulation
Validation of simulation results
Kernels and validation
Speedup and validation of simulation process
Refinement &Validation of Analytical models
User Applications
Experimental measuremnt
Output
validation
(General performance results added to performance knowledge-base)
Updation of knowledge base
Knowledge Set Extraction
Theoretical/ Analytical models
User Applications
Knowledge base system for experimental measurement
Updation of knowledge base (General performance results added to performance knowledge-base)
Knowledge Set Extraction
Knowledge base system for Theoretical/Analytical modelling

Figure 1. Showing integrated modeling of performance evaluation techniques.

8. Conclusion and Future Directions

Performance evaluation as a discipline has repeatedly proved to be critical for design and successful use of parallel systems. At the early stage of design, performance models can be used to project the system scalability and evaluate design alternatives. At the production stage, performance evaluation methodologies can be used to detect bottlenecks and subsequently suggest ways to alleviate them. In these paper three techniques of parallel system performance evaluation are reviewed and compared with each other with the help of four major parameters stage, output statistics, accuracy and cost/effort involved. Other parameters involved in the selection of performance evaluation technique are Resource consumption, time consumption, tools required, trustability, scalability complexity and flexibility. Each of the three techniques has their own pros and cons. So an integrated model that uses the power of knowledge base systems and combines advantages of all the three techniques is discussed. This paper also discusses certain issues like selecting an appropriate metric for evaluating parallel system; need to select proper workload and workload characterization. A future direction for this paper is to test and verify the proposed model by carrying out simulation.

References

[1] Bailey D., Barszcz E., Barton J., Browning D., Carter R., Fatoohi R., Frederickson P., Lasinski T., Simon D., and Venkatakrishnan V., "The NAS Parallel Benchmarks," *International Journal of Supercomputer Applications*, vol. 5, no. 3, pp 63-73, 1991.

[2] Berryetal M., "The Perfect Club Benchmarks: Effective Performance Evaluation of Supercomputers," *International Journal of Supercomputer Applications*, vol. 3, no. 3, pp. 5-40, 1989.

[3] Fortune S. and Wyllie J., "Parallelism in Random Access Machines," *in Proceedings of the 10th Annual Symposium on Theory of Computing*, pp. 114-118, 1978.

[4] Gibbons P., "A More Practical PRAM Model," *in Proceedings of the First Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 158-168, 1989.

[5] Ibbett R., "The Hardware Monitoring of a High Performance Processor," *in Proceedings of Benwell Computer Performance Evaluation*, pp. 274-292, UK, 1978.

[6] Malony A. and Reed D., "Performance Measurement Intrusion and Perturbation Analysis," *IEEE Transactions on Parallel and Distrubuted Systems*, vol. 3, no. 4, pp. 443-450, 1992.

[7] Patterson D., *Introduction to Artificial Intelligence and Expert Systems*, Prentice Hall India, 1998.

[8] Plattner B. and Nievergelt J., "Monitoring Program Execution: A Survey," *IEEE Computer*, vol. 14, no. 11, pp. 76-93, 1981.

[9] Power L., "Design and Use of a Program Execution Analyzer," *IBM Systems Journal*, vol. 22, no. 3, pp. 271-294, 1983.

[10] Sivasubramaniam A., Singla A., Ramachandran U., and Venkateswaran H., *A Comparative Evaluation of Techniques for Studying Parallel System Performance,* Technical Report, 1994.

[11] Worlton J., "Toward a Taxonomy of Performance Metrics," *Computer Journal of Parallel Computing*, vol. 17, no. 10, pp. 1073-1092, 1991.

Amit Chhabra received B.Tech in computer science & engineering and M.Tech from Guru Nanak Dev University, Amritsar. He is currently working as a lecturer in the Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar. His research areas include parallel and distributed systems.

Gurvinder Singh has received MCA and PhD from Guru Nanak Dev University, Amritsar. He is currently working as a reader in the Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar. His research areas include distributed systems, parallel computing, and grid computing.