Analysis & Integrated Modeling of the Performance Evaluation Techniques for Evaluating Parallel Systems

Amit Chhabra chhabra_amit78@yahoo.com
Department of Computer Science & Engineering,
Guru Nanak Dev University,
Amritsar, 143001,India

Gurvinder Singh gsbawa71@yahoo.com
Department of Computer Science & Engineering,
Guru Nanak Dev University,
Amritsar, 143001,India

Abstract

Parallel computing has emerged as an environment for computing inherently parallel and computation intensive applications. Performance is always a key factor in determining the success of any system. So parallel computing systems are no exception. Evaluating and analyzing the performance of parallel systems is an important aspect of parallel computing research. Evaluating and analyzing parallel system is difficult due to the complex interaction between application characteristics and architectural features.

Experimental measurement, Theoretical/Analytical modeling and Simulation are the most widely used techniques in 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. Each of these three techniques has their own pros and cons.

The purpose of this paper is firstly to present a qualitative parametric comparative analysis of these techniques based on parameters like stage, output statistics, accuracy, cost, resource consumption, time consumption, flexibility, scalability, tools required, trustability and secondly to justify the need for an integrated model combining the advantages of all these techniques to evaluate the performance of parallel systems and thirdly to present a new integrated model for performance evaluation . This paper also discusses certain issues like selecting an appropriate metric for evaluating parallel systems.

Keywords: Integrated model, Metrics, Parallel systems, Performance, Evaluation

1.INTRODUCTION TO PARALLEL COMPUTING SYSTEMS

For the last three decades, researchers in the area of parallel processing have proclaimed that parallel computing is the wave of the future. The reason most widely cited for this wave is the rapid approach that resolves the speed limit of serial computing. This limit needs to cracked in order to solve many computing intensive applications in an acceptable amount of time. The only way to overcome the speed limit of serial computers is to harness the power of several serial computers to solve a single problem in a parallel fashion.

The advents in the today's micro-electronic technology have resulted in the availability of fast, inexpensive processors and advancement in the communication technology has resulted in the availability of cost-effective and highly efficient computer networks.

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. Different metrics may result in totally different values. Hence, selecting proper metrics to fairly evaluate the performance of a system is difficult. Another problem with selecting metrics is that in a real system different metrics may be relevant for different jobs. For example, response time may be the most suited metric for interactive jobs, while system utilization is more important for batch jobs. So issue here is to select an appropriate metric, which is suitable for every kind of systems (batch, interactive, closed and open systems).

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 [1] have proposed a Law for *Speedup* metric. The formulation for speed up using Amdahl's law are 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.

John Gustafson [1] proposed a fixed law using time concept to scale the speed up model and remove the fixed load restriction. This law states that problem size scales with the number of processors and idea is to keep all processors busy by increasing the problem size. Other variants of *speedup* are *Relative speedup* [6][7][8], *Real speedup* [3][5], *Absolute speedup* [6][7][8] and *Asymptotic relative speedup* [4].

Sun and Ni [6][8] 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 [7] 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*. In *sizeup* metric, the instance being solved is adjusted as for fixed-time 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 [2] define efficiency as the ratio of work accomplished (*wa*) by a parallel algorithm and the work expended (*we*) by the algorithm. Here work accomplished (*wa*) by the parallel algorithm is the work that is being done/performed by the "best" serial algorithm and work expended (*we*) is the product of the parallel execution time, the speed (*S*) of an individual parallel processor and the number P of processors. Also there is one another variant of efficiency known as *Incremental efficiency*. It is an approach to look at change of the efficiency of the implementation as the number of processors increases. The *Incremental efficiency* metric [11] 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. Intuitively, a parallel system is scalable if its performance continues to improve as we scale the size of the system.

Kumar, Nageshwara and Ramesh [10] 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. Isoefficiency is generally a function of the instance and number of processors in the parallel computer. 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 [9] 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 may be defined as ratio of total usage time over the total available time of the resources (processors, memory). 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. 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:

a) *Stage*-this parameter examines which performance evaluation technique should be used at what stage of system development life cycle.
b) *Output statistics*-this parameter examines the capabilities of the technique towards providing the desirable metrics.
c) *Accuracy*-this factor evaluates the validity and reliability of the results obtained from the technique.
d) *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:
e) *Resource consumption*-this parameter examines the amount of resources consumed/required by a particular performance evaluation technique.
f) *Time consumption*- this parameter examines the amount of time consumed/required by a particular performance evaluation technique.
g) *Tools required*- this parameter examines the type of tools required for implementation of any particular performance evaluation technique.
h) *Trustability/Believability*- this parameters reveals that how much one can trust on the results of a particular performance evaluation technique.
i) *Scalability complexity*-this parameter examines the complexity involved in scaling a particular performance evaluation technique.
j) *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.

4.1. 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 [12]. 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 [13].

A hybrid monitor is a combination of software and hardware. Many examples of software monitors can be found in the literature [14][15][16]. Examples of hardware monitors are COMTEN and SPM [17]. [18] describes the Paragon performance-monitoring environment that uses a hardware performance monitoring board. And examples of hybrid monitors are Diamond [20] and HMS [19].

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 [15]. This has been shown by the fact that the number of existing software monitors are far greater than that of hardware monitors. [21] describes and compares software and hardware monitors in more detail. Application suites such as the Perfect Club [22], the NAS Parallel Benchmarks [23], and the SPLASH application suite [24] have been proposed for the evaluation of parallel machines.

Parameters under consideration

a) *Stage*- As 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.

b) *Output statistics*-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.

c) *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.

d) *Cost/Effort*-Substantial costs are involved in developing the model for experimentation as experimental 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.

e) *Resource consumption*-This technique requires actual instruments, so resource consumption is high.

f) *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.

g) *Tools required*-Actual instruments are required here to evaluate any parallel application

h) *Trustability/Believability*-As it uses actual hardware so results given by it can be highly trustworthy. Here results could be validated by atleast simulation or theoretical/analytical modelling.

i) *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.

j) *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.

4.2 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 [25][26] has facilitated algorithm development and analysis. These models try to hide hardware details from the programmer, providing a simplified view of the machine. The utility of such models towards developing efficient algorithms for actual machines depends on the closeness of the model to the actual machine. Several machine models like Bulk Parallel Random Access Machine (BPRAM)[29], Module Parallel Computer (MPC)[30], Valiant [28] introduces the *Bulk-Synchronous Parallel (BSP)* model and LogP[27] 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. [31] 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-

a) *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.

b) *Output statistics* – 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.

c) *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.
d) *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.
e) *Resource consumption*-Not much resources are required to build models for theoretical/analytical modelling.
f) *Time consumption*- Good 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.
g) *Tools*- It involves analyst and various analysis tools.
h) *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.

i) *Scalability complexity*-It takes time to scale these kind of models.
j) *Flexibility*-It is flexible as changes can be made to the models easily and effect of change can also be checked easily.

4.3. 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 been implemented. It can also be used to validate analytical models, as is done in [32][34]. 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 authors *instruction-level simulation* [35] or *cycle-by-cycle simulation*. Programs written for the target processor are simulated by the simulator running on the host processor to study the behavior of the programs if they were executed on the target processor.

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 token 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 queueing systems.

Parameters under consideration-

a) *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.

b) *Output statistics*-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.

c) *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.
d) *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 we 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.
e) *Resource Consumption*-It requires small amount of resources as we are just simulating the parallel environment, no actual instrumentation is being required.
f) *Time consumption*-Good amount of time is consumed for developing the simulation model. But once model has been developed it takes little time to get results.
g) *Tools*-Simulation requires high level computer programming languages to build and develop the model.
h) *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.
i) *Scalability complexity*-Simulated models are very easy scalable as it artificially simulates the actual environment.

j) *Flexibility*-The main advantage of simulation is its flexibility. One can make various modifications to the simulation model and check their effect easily.

5. 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 knowledge based integrated model, which combines the advantages of all the three techniques.

	Performance Evaluation Techniques		
Characteristic	**Theoretical/Analytical Modelling**	**Simulation**	**Experimental Measurement**
a) Stage	Any	Any	After prototype is available
b) Output Statistics	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	Small	Medium	High
f) Time Consumption	Small	Medium	High
g) Tools	Analysts	Computer Programming Languages	Instruments
h) Trustability/ Believability	Low	Medium	High
i) Scalability Complexity	Small	Medium	High
j) Flexibility	High	High	Low

TABLE1: Showing comparison of performance evaluation techniques

This integrated model has 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 avoid some of their drawbacks. Also we need an integrated model that takes care of 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.
- Do not trust the results of a Experimental measurement model until they have been validated by at least Simulation or Theoretical/analytical modelling.

6. PROPOSED INTEGRATED MODEL

This integrated model shown in Fig. 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 Fig. 1. Based on the user application, the type of workload it is using (real or synthetic) and any other current data, a knowledge set

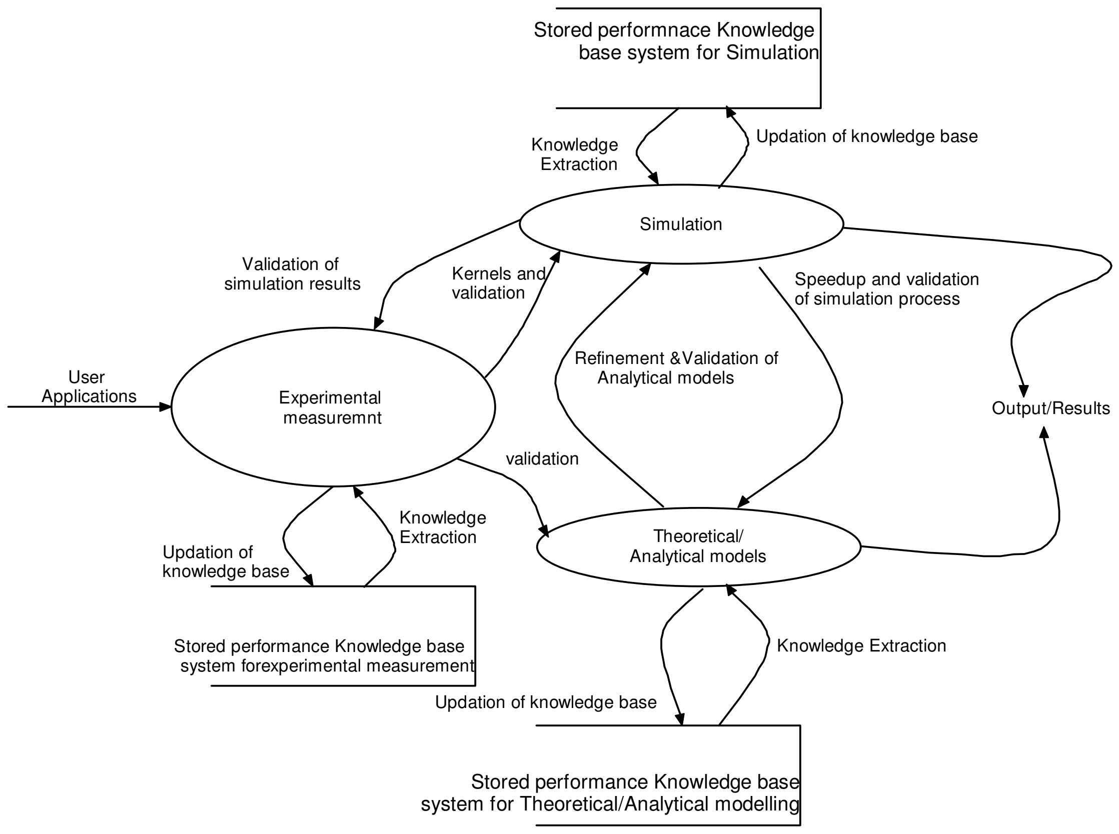
Stored performnace Knowledge base system for Simulation
Knowledge Extraction
Updation of knowledge base
Simulation
Validation of simulation results
Kernels and validation
Speedup and validation of simulation process
Refinement &Validation of Analytical models
User Applications
Experimental measuremnt
Output/Results
validation
Knowledge Extraction
Updation of knowledge base
Theoretical/ Analytical models
Stored performance Knowledge base system forexperimental measurement
Knowledge Extraction
Updation of knowledge base
Stored performance Knowledge base system for Theoretical/Analytical modelling

FIGURE 1: Diagram showing proposed integrated model for performance evaluation

can be extracted from this stored 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 results can in turn act as knowledge. This valuable knowledge can be updated back into the knowledge base system and this procedure of knowledge extraction and updation can be repeated. Stored 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 stored knowledge base systems and avoids the shortcomings of the individual evaluation techniques.

7. CONCLUSION

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 this 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.Issue like selecting an appropriate metrics for performance evaluation is also discussed.

8.REFERENCES

[1]J.Gustafson, "*Reevaluating Amdahl's Law*", CACM, 31,5,532-533,1988.
[2]E.Caromona ,M.Rice, "*Modelling the serial and parallel fractions of a parallel algorithm*",Journal of Parallel and Distributed Computing,13,286-298,1991.
[3]J.JaJa, " *An introduction to parallel algorithms*",Addison Wesley,1992.
[4]D.Nussbam and A.Agrawal, " *Scalability of parallel machines*",CACM,34,3,57-61,1991.
[5]S.Ranka, S.Sahni, "*Hypercube algorithms*",Springer-Verlag,New York,1990.
[6]X.Sun, L.Ni, "*Another view on parallel speedup*",Proceedings Supercomputing 90,324-333,1990.
[7]X.Sun ,J.Gustafson, "*Towards a better parallel performance metric*",Parallel Computing,17,1093-1109,1991.
[8]X.Sun ,L.Ni, " *Scalable problems and memory-bouneded speedup*", Journal of Parallel and Distributed Computing,19,27-37,1993.
[9]X.Sun ,D.Rover, "*Scalability of parallel algorithm-machine combinations*",IEEE Transactions Of Parallel and Distributed systems,5,6,599-613,1994.
[10]V.Kumar,V.Nageshwara and K.Ramesh, "*Parallel depth first search on the ring architecture*", Proc.1988 International Conference on Parallel Processing,Penn. State Univ. Press,128-132,1988.
[11]J.Worlton,"*Toward a taxonomy of performance metrics*", Parallel Computing 17(10-11): 1073-1092 (1991) .
[12]Jelly, I. ,Gorton, I., "*Software engineering for parallel systems*", Information and Software Technology, vol. 36, no. 7, pp. 381-396, 1994.
[13]Ferrari, D., "*Considerations on the insularity of performance evaluation*", Performance Evaluation Review, vol. 14, no. 2, pp. 21-32, August 1986.
[14]Plattner, B. ,Nievergelt, J., "*Monitoring program execution: A survey,*" IEEE Computer, vol. 14, pp. 76-93, November 1981.
[15]Power, L. R., "*Design and use of a program execution analyzer,*" IBM Systems Journal, vol. 22,no. 3, pp. 271-294, 1983.
[16]Malony, A. D., Reed, D. A. ,Wijshoff, H. A. G., "*Performance measurement intrusion and perturbation analysis,*" IEEE Transactions on Parallel and Distrubuted Systems, vol. 3, no. 4, pp. 443-450, July 1992.
[17]Ibbett, R., "*The hardware monitoring of a high performance processor,*" in: Benwell, N. (ed), Computer Performance Evaluation, Cranfield Institute of Technology, UK, pp. 274-292, December 1978.
[18]Ries, B., Anderson, R., Auld, W., Breazeal, D., Callaghan, K., Richards, E. and Smith, W., "*The Paragon performance monitoring environment,*" Proceedings of the conference on Supercomputing'93, pp. 850-859, 1993.

[19]Hadsell, R. W., Keinzle, M. G. and Milliken, K. R., "*The hybrid monitor system*," Technical Report RC9339, IBM Thomas J. Watson Research Center, New York, 1983.

[20]Hughes, J. H., "*Diamond ¾ A digital analyzer and monitoring device*," Performance Evaluation Review, vol. 9, no. 2, pp. 27-34, 1980.

[21]Jain, R., *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, John Wiley & Sons, New York, 1991.

[22]M.Berryetal. *The Perfect Club Benchmarks: Effective Performance Evaluation of Supercomputers*. International Journal of Supercomputer Applications, 3(3):5–40, 1989.

[23]D. Bailey et al. *The NAS Parallel Benchmarks*. International Journal of Supercomputer Applications, 5(3):63–73, 1991.

[24]J. P. Singh, W-D. Weber, and A. Gupta. SPLASH: Stanford Parallel Applications for Shared-Memory. Technical Report CSL-TR-91-469, Computer Systems Laboratory, Stanford University, 1991.

[25] S. Fortune and J. Wyllie. Parallelism in random access machines. In *Proceedings of the 10th Annual Symposium on Theory of Computing*, pages 114–118, 1978.

[26]P. B. Gibbons. A More Practical PRAM Model. In *Proceedings of the First Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 158–168, 1989.

[27]D. Culler et al."*LogP: Towards a realistic model of parallel computation*"In Proceedings of the 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pages 1–12, May 1993.

[28]L. G. Valiant. "*A Bridging Model for Parallel Computation*" Communications of the ACM, 33(8):103–111, August 1990.

[29]A. Aggarwal, A. K. Chandra, and M. Snir " *On Communication Latency in PRAM Computations*" In Proceedings of the First Annual ACM Symposium on Parallel Algorithms and Architectures, pages 11–21, 1989.

[30]H. Alt, T. Hagerup, K. Mehlhorn, F. P. Preparata " *Deterministic Simulation of Idealized Parallel Computers on More Realistic Ones*" SIAM Journal of Computing, 16(5):808–835, 1987.

[31] D. F. Vrsalovic, D. P. Siewiorek, Z. Z. Segall, and E. Gehringer "*Performance Prediction and Calibration for a Class of Multiprocessors*" IEEE Transactions on Computer Systems, 37(11):1353–1365, November 1988.

[32]Agarwal, A., *"Performance tradeoffs in multithreaded processors*," IEEE Transactions on Parallel and distributed Systems, vol. 3, no. 5, pp. 525-539, September 1992.

[33]Menasce, D. A. ,Barroso, L. A., "*A methodology for performance evaluation of parallel applications on multiprocessors,*" Journal of Parallel and Distributed Computing, vol. 14, pp. 1-14,1992.

[35]Covington, R. G., Dwarkadas, S., Jump, J. R., Sinclair, J. B. ,Madala, S., "*The efficient simulation of parallel computer systems,*" International Journal in Computer Simulation, vol. 1, pp.31-58, 1991.