Goals of this Dagstuhl Seminar

Günter Haring, Christoph Lindemann, Martin Reiser

Performance Evaluation is a discipline of Computer Science for some thirty years. It seems time to take stock of what we were doing. That is, provide answers to the following questions:

- What are its scientific contributions?
- What is its relevance in industry and business?
- What is its standing in academia?
- Where is the field headed?
- What are its success stories and failures?
- What are its current burning questions?

At this Dagstuhl seminar, we were taking the term workshop literally and have broken up in five working groups. These groups worked in parallel on different areas of Performance Evaluation. Each group prepared a written report during the stay at Dagstuhl. These reports constitute the remainder of this text. The ultimate goal of this seminar is to publish a White Book on Performance Evaluation that will become a catalyst for the advancements of this field as well as a book of reference.

Furthermore, each group discussed *Success Stories, Failures, Impact of Methods,* and *Burning Questions* of Performance Evaluation. A summary of this discussion is given in the following:

Success Stories

- Computer architecture, in particular RISC, cache coherence, and multiprogramming.
- IEEE 802 suite of protocols, in particular medium access control level.
- Data networks, in particular TCP/IP.
- Storage systems, in particular RAID and virtual memory.
- Operating systems, in particular timesharing, MVS, and scheduling.
- Realization and implementation of methodological results in easy-to-use software packages.

Failures

- Late consideration of performance issues in system design process.
- Bad quality of measured input data.
- Wrong level of abstraction.
- Educational failures
  - Lack of making known methods to customers.
  - Lack of training of modeling skills.

Impact of Methods

- BCMP theorem made impact to e.g., capacity planning, multiprogramming, and communication networks.
- MVA algorithm substantially increased the applicability of BCMP queueing networks with several customer classes for supporting the design of computer and communication systems.
- Particular stochastic models with closed-form solutions made impact to design of IEEE 802 medium access control protocols.
- Trace-driven simulation made impact to cache and processor architecture design.

Burning Questions

- Complexity of today's systems:
  - Scalability of performance evaluation techniques.
  - Validation
  - Parameter estimation, large amount of measured data.
  - Develop methods for handling very large models with sufficient level of detail.
- Bridging the gap between theoretical and applied work:
  - Seamless integration of performance models into design process.
  - Application-specific parametrizable performance models.
- Academic education and university research:
  - Curriculum for computer scientists and engineers has not enough exposure to Performance Evaluation methods.
  - Faculty positions and research funding is inadequate for Performance Evaluation methodology. I.e., is pure Performance Evaluation research an endangered species?
- Acceptance in industry:
  - Performance Evaluation experts must be routinely be involved in system design process.
  - Short development cycle, short live cycles of technologies.
  - Education of university graduates.

Report of the Performance Evaluation Techniques and Tools Group

Peter Buchholz, Jeffrey Buzen, Hans Daduna, Boudewijn Haverkort, Ullrich Herzog, Helena Szczerbicka

1 A brief history

1.1 Techniques

Performance evaluation, an important topic for telephone network planning since the beginning of this century, got a major stimulation with the advent of computer-communication networks and time sharing computers in the 1960's when engineers realized that there were no adequate methods for the planning of these increasingly complex (queueing) systems.

In the period **1960-70** the first important methods were imported from the Operations Research area, most prominently, open Jackson networks to describe packet switching networks. Also the modeling of time-shared computer systems by discrete-time queueing systems popularized queueing theoretical methods. The invention of closed Gordon-Newell networks was a further step towards more advanced modeling techniques.

During **1970-80**, the performance evaluation community developed further models tailored for various applications: Processor sharing as an approximation of time-sharing disciplines, models of multiprogrammed systems in more general terms than Gordon-Newell networks, and BCMP and Kelly networks as general purpose modeling techniques for networks of multiclass queues. Shortly thereafter, the convolution algorithm and Norton's theorem have become popular as easy to use methods for parametric system analysis. At the same time, in the field of simulation, the regenerative simulation method was developed as being well-suited for the statistical evaluation of queueing applications in the computer and communications field.

The developments in the period **1980-90** started with proving the arrival theorem and the development of the Mean Value Analysis algorithm. This was followed by constructing several classes of approximation algorithms for first order quantities in queueing networks, which mostly depend on MVA or convolution algorithms. A great variety of aggregation/disaggregation techniques and algorithms which depend on these algorithms and often on Norton's theorem (and its variants) appeared at this time. These modelling principles have been influencing the development of techniques and tools up to now. Refinements of the queueing models by using matrix-analytic methods and applying uniformization to study transient phenomena broadened the application of queueing models further. With the unification of performance and reliability features in a common class of models the notion of performability was established. A new class of models was introduced when the need of synchronising processes over time became an important requirement which leads to stochastic Petri nets and generalized stochastic Petri nets.

In the simulation area, trace- and execution-driven simulation became popular, and in stochastic simulation variance reduction techniques were developed, especially importance sampling. Finally, stochastic graph models were developed to allow for performance evaluations of interdependent tasks in multiprocessor systems.

Since **1990**, parallel simulation of processes and systems has rapidly been developed, and is still in progress today. This clearly holds as well for the following topics in the field of analytical methods: state space reduction methods, non-Markovian stochastic models, fixed-point methods for general systems, methods for hierarchical modelling, tensor representation, exploiting model symmetries and the development of new modelling concepts like combining

paradigms from queueing networks with stochastic Petri nets. Renewed interest from emerging applications find discrete-time queues as well as fluid approximations for first-order quantities (e.g., fluid Petri nets), and diffusion approximations for second-order descriptions. Furthermore, techniques borrowed from artificial intelligence to support, among others, model building and experimentation are being incorporated in performance evaluation.

Finally, there is a clear trend to expand the initial ideas of stochastic Petri nets and stochastic graph models; specification techniques for distributed systems and protocol design (such as SDL and LOTOS) are enhanced with stochastic (timing) information, thus allowing for the combined analysis of functional as well as performance properties.

1.2 Tools

Shortly after the popularization of queueing network models which incorporate detailed customer and system behaviour (BCMP and Kelly networks) these models became the theoretical foundation of performance evaluation tools. New tools also lead to the development of new classes of queueing networks, including the capabilities to handle non-separable networks. Also using simulation and approximation algorithms, they offered a versatile class of modeling features for evaluating large systems. In the middle of the eighties the first tool using stochastic Petri nets was build. A large number of different tools has been developed since then, following the increase of computing power, and this is still going on today. Modeling and evaluation tools of a new era came on with the beginning of the nineties: Modeling environments which offer versatile classes of queueing and synchronization features to the user in a friendly and convenient way. At the same time hierarchical modeling tools appeared which increased the size of the systems which can be handled considerably. A new point of increasing importance now considered is the automatic optimization of systems in all their complex modeling features.

With the introduction of user-friendly front ends the tools can and are now tailored towards specific application areas. Typically in these tools all the mathematical detail of general performance evaluation techniques are hidden, thus letting the user concentrate on his application problems. It should be pointed out that during the, mostly research-driven development, of the packages as described above, commercial tools for specific application areas have been developed as well. These used special purpose methods selected for a specific field of applications with an user-interface specified by the requirements of the field.

2 Future developments

Existing techniques and tools have now been successfully used in research and development and have found several commercial applications as well. However, there is still a need for their extension in order to improve their efficiency, applicability and acceptance, as well as to cover domain specific needs.

Especially methods dealing with complexity, and particulary with complexity in model specification, are of great importance. Further compositional and hierarchical techniques, multi-paradigm techniques, specification techniques directly related with those used in application areas and methods to transform models from a domain-specific to a solution-oriented specification are to be developed. To cope with increasing system complexity, we need more advanced analytical and numerical techniques, in particular: fast bounding techniques for complex systems, techniques to handle large state spaces, techniques to handle adaptive systems, time-dependent analysis techniques, and techniques for analyzing systems subject to real-time requirements. Further challenges lie in the area of queues with complex inputs, such as self-similar traffic, and

techniques to compute resource demands of hierarchically structured software systems. (Noteworthy: most of these requirements come from the group discussing communication networks' developments.)

Simulation is still the most frequently used and the best accepted technique. However, all applications suffer from insufficient performance of large simulation studies. To overcome this problem we need hierarchical and decompositional methods for simulation, abstraction in mapping problems to simulation models and the possibility of combining simulation and analytical/numerical techniques. Speed up of general simulation procedures by enhanced variance reduction techniques also need to be developed further.

Additionally, requirements on existing and future techniques for easy usage of tool come from the users' side. Hiding mathematical details of performance evaluation methods makes these methods available in a variety of application areas. Making techniques available in tools, or packaging performance evaluation algorithms can essentially improve the acceptance and usability of these techniques. For efficiently parameterizing models, the availability and interpretation of data must be enhanced considerably. Currently, there is a deficit with respect to measuring existing systems, predicting data for planned systems, handling large data sets arising from measurements, and representing and understanding measured data.

Developing new tools, we are faced with the problem of capturing methods with the capabilities mentioned above, and with fulfilling requirements coming from the application domain. The discussions during the workshop clearly indicate that tools are only applied successfully if they meet the requirements of the specific application areas. Interfaces specific to the domain of the system to be modelled, seamless integration of performance evaluation techniques with development tools from other areas (e.g., VLSI, software engineering, etc.), integration of performance evaluation techniques with other analysis goals (e.g., functional analysis, optimisation) are some examples of this type of requirements. The complexity problem should also be addressed from the tools' perspective. Tools not only have to support hierarchical specification and the possibility of expressing several levels of details in a model, but also should allow for the use of libraries of components.

3 Conclusions

The discussions with the application-oriented groups indicated that there are urgent needs for extension of functionality of performance evaluation tools in order to integrate performance modelling and evaluation in the more general development and/or analysis process. This comprises, among others, data collection and model parametrization, optimization, interpretation of results and support in experiment design. Challenges for the performance evaluation community in developing new methods and tools comprise especially the following:

- Responding to the development of systems which feature new applications and new dimensions of complexity by providing new methods and theories.
- Enhancing existing tools and tailoring them to particular application domains.
- Broadening the scope of methods in order to meet the multi-skill character of performance evaluation work for real-world problems.
- Reporting about important performance evaluation results in application-oriented journals and conferences.

Report of the Communication Networks Group

Marco Ajmone Marsan, Jonathan Billington, Edmundo de Souza e Silva, Raymond Marie, Hideaki Takagi, Satish Tripathi

The above group of 6 researchers met to list important achievements in the area of performance evaluation of communication networks. The organizers had suggested that we consider the issues of success stories, impact, failures, and burning questions. After some discussion it was agreed to brainstorm and document the results under the headings of success stories and the impact of performance evaluation in the field, important techniques, and future challenges.

1 Success Stories and Impact

The computer communication area was broken into different major applications and for each we listed techniques that have been found to be useful in modeling such systems with a view to understanding their overall behavior. These areas are given below.

The design of *circuit switching networks* was probably the first application where performance evaluation made a major impact. Important techniques used are: Erlang formulas (infinite population); "Generalized" Erlang formulas; Engset formulas (finite population); graph theory (shortest path, minimum spanning tree); and optimization (network flow, minimum cut/maximum flow).

Reliability was already a main concern in the early telephone networks. Recently performability analysis was applied to the design of a new generation of circuit switches.

Packet switching networks were introduced to improve efficiency for data communications, and thus right from the beginning were aimed at performance improvements. The major techniques used for its design and analysis are: Jackson networks together with Kleinrock's independence assumption and BCMP networks for the analysis of more complex queueing networks. Efficient solution techniques followed and MVA also provided the basis for many approximations. Flow deviation is an optimization technique used in the design process, and fluid models proved to be valuable.

Protocols both for circuit and packet switching networks were an area of important application for performance evaluation techniques. Some techniques for protocol design that concern specification, verification and testing (LOTOS, ESTELLE, SDL, Temporal Logic, Process Algebras, Petri Nets) served as the basis for the development of approaches to combine quantitative and qualitative analysis. An example is provided by Stochastic Petri Nets.

The Internet TCP protocol is an excellent example where analysis influenced its successive refinement. Random access protocols were only accepted because of the performance analysis that supported the feasibility of the probabilistic approach. For *satellite networks* (such as Aloha and subsequent reservation schemes) Markov chains and Markov decision processes were extensively used in the analysis.

The milestone paper on the Ethernet is also an excellent example of the use of basic modeling techniques to support the proposal of a new protocol that later became an international and industry standard for the LAN market. Markov chains and Markov decision processes were used in many papers that followed and helped with understanding the main issues of CSMA type protocols. The relative merits of different proposals (such as the token ring versus Ethernet), were mostly decided according to performance measures. These performance studies led to the proposal of new protocols such as the token bus and tree-based protocols. Besides Markov chain theory, polling models played a major role in the analysis of token passing networks.

In the case of MANs, FDDI and DQDB networks are good examples of protocols with complex behavior. Performance analysis was used to understand critical issues. Markov chain analysis, bounding analysis and polling models have been used to increase understanding of time-limited token protocols, such as FDDI. These analyses were instrumental in establishing fairness and response time guarantees of the protocol. In the case of DQDB, performance analysis uncovered a fairness problem in the standard and was used to find a solution. Exploratory networks such as Meta Ring and ATM LANs are also being developed to enhance performance.

Performance analysis has had a key role in the standardization of ATM technology. In the ATM forum, new proposals must be supported by performance studies, based on comprehensive models. In these studies, discrete time models, matrix geometric analysis, fluid models, large deviation theory and rare event simulation (variance reduction techniques) were used. The integrated service requirements have given rise to a new class of scheduling disciplines, for example the fair queueing discipline and virtual clock. Performance models were used to prove hard bounds for these approaches. These bounds are essential for determining the quality of service required by new multimedia applications. The whole class of models developed are used not only for ATM but also for the many resource allocation problems associated with high speed networking technology.

The Poisson arrival model, largely used in the design of telephone networks, has turned out not to be appropriate for characterizing the traffic generated by multimedia sources. This led to a new class of traffic models that better represent the burstiness and correlation of these traffic streams. For example, Markov modulated Poisson processes, fluid (gradual input) models and self similarity provide better representations of the traffic.

In the seventies, Markov analysis was used to study major issues related to *packet radio networks*. Recently *wireless networks* have spawned a whole set of new challenges such as channel allocation and mobility management (protocols/signaling/databases). Performance models are at the core of solutions which can achieve the efficiency necessary to make wireless computing ubiquitous. Mostly Markovian models are used for analysis.

2 Important Techniques

Modeling techniques have played a major role in the design and evaluation of protocols. As early as 1907 *Erlang's formula* was used to configure the number of communication lines in telephone networks. Even today these basic results are employed to dimension modern telephone switching networks.

The development of *queueing network* theory by Jackson, combined with Kleinrock's *independence assumption*, was central to the initial design of the ARPANET (which later became today's internet). Important extensions (such as the *BCMP* theorem) to Jackson's results and efficient algorithms (Buzen) expanded the types of the networks that could be analyzed (e.g. multiclass open and closed networks). *Mean Value Analysis* (MVA) results provided a framework to compute the average delays efficiently and became a basis for a number of approximations to analyze complex communication systems. For example, these results were used to model flow control, admission control, buffer allocation and scheduling algorithms.

Graph theory and optimization techniques contributed significantly in configuring network capacity to handle increased offered load. *Network flow* and *flow deviation* techniques provide the mathematical foundation for optimization and topological design. Shortest path algorithms (e.g. Dijkstra, Bellman-Ford) are extensively used in the design of routing protocols.

The advent of local area networks, and in particular token passing protocols, motivated the development of *polling models* to take into account complex interdependencies that exist in the

system's behavior. Modern medium access protocols for metropolitan area networks and some multiplexing techniques for multimedia traffic have key features that can be analyzed by polling models.

Markov models which had been used earlier for a variety of applications, continue to be important for the performance evaluation of communication systems. Their capability has been extended by techniques such as *Matrix Geometric* and *Kronecker algebra* to efficiently solve a broad class of large models. The introduction of *Stochastic Petri Nets* (SPN) allowed more complex models of systems to be developed and from which Markov chains could be automatically generated.

Most of the Markov model studies have focused on the steady state behavior of systems. In order to evaluate system *reliability* time dependent analysis is an issue. Recently, a lot of effort has been devoted to finding transient measures from Markov models. *Transient analysis* also plays an important role in the design and evaluation of control policies in communication networks. The need to design systems that degrade gracefully stimulated the definition of new metrics and the development of techniques which take into account the combined effect of both performance and dependability (*Performability*).

Fluid models were introduced to manage large state space cardinalities. More recently, fluid approximations of bursty traffic have been extensively used to evaluate admission and flow control policies of multimedia networks.

3 Future Challenges

The continuing increase in the number of technologies and services in future generations of networks will offer a number of challenges for performance evaluation in the years to come. Some of the topics that appear most interesting today are listed below. They include areas such as Quality of Service (QoS) in multiservice networks, wireless networks, photonic networks, and general techniques for performance analysis.

3.1 Quality of Service in multiservice networks

QoS is an important issue for multiservice networks and is independent of the adopted technology. In particular, it applies to both IP-based and ATM-based networks. Today, the problems that appear to be most relevant in this field are: analysis of closed loop control schemes (Available Bit Rate (ABR) in particular, and its interactions with TCP/IP); design of effective traffic shaping algorithms; identification of realistic source models for the different types of services; design and analysis of efficient multicast protocols; identification of effective approaches for the provision of real time guarantees; media synchronization for multimedia services; detailed models of user equipment and switches; solution of end-to-end multilayer models capturing several aspects of end-user equipment behavior as well as network behavior; and analysis of request/reservation protocols (e.g. RSVP).

3.2 Wireless networks

The extraordinary success of mobile telecommunication services is generating a number of analysis and design problems based on performance and QoS issues. The future introduction of integrated services for mobile users will further increase the number of problems to be analyzed and solved. Among the current problems, the ones that appear to be the most challenging are: analysis of mobility management protocols, the associated signaling schemes, and of the

interactions with databases; identification of accurate non-stationary traffic models; transient analysis of network models, for instance, to cope with the high mobility of users and the consequent changes in traffic patterns; static and dynamic optimization of the network configuration (number of cells, allocation of frequencies to cells and transmission power control); design and performance analysis of air interface access protocols (eg TDMA and CDMA); and interoperability of land/radio/satellite networks (for example for Universal Personal Communications).

3.3 Photonic Networks

The use of photonic technologies for the design of all-optical networks is an approach that has potential to significantly increase communication network speeds. Several technological problems remain to be solved in order to design photonic switching nodes and all-optical networks. The choice among alternative designs must be guided by careful performance analysis in order to exploit, to the maximum possible extent, the photonic components that will be available as technology progresses. Relevant performance problems in this field are development of accurate switch models and development of models of multiservice photonic networks.

3.4 Analysis Techniques

The challenges posed by emerging networking architectures and protocols can be met by performance analysis only if some of the existing approaches can be extended to cope with the new environments, and if some new analysis techniques can be developed. Important theoretical questions that are raised by networking applications include: accurate traffic characterization; development of modeling approaches that can cope with self similarity; (for instance, we do not understand well the impact of user behavior and policing over the quality of service provided, and the QoS sensitivity to changes in the model parameters;) solution of models with very large state spaces; solution of complex non-Markovian models (in steady state, as well as transient); developments in sensitivity/perturbation analysis; development of variance reduction techniques for general simulation models; analysis of deterministic models; use of stochastic geometry approaches; exact and approximate solutions of discrete time queueing networks; and development of hierarchical decomposition/aggregation approaches. These techniques should be developed on the basis of exact analysis or legitimate approximation and validation by measurements.

Report of the Computer Architecture Group

Sarita Adve, Margaret Martonosi, David Nicol,
Herb Schwetman, Mary Vernon

This paper describes the position of the computer architecture group at the Dagstuhl Workshop. Performance evaluation is ubiquitous in the computer architecture community. Much of this work, however, is based on instruction-level simulation models. The key thesis of this paper is that low-level simulation-based analysis is running out of gas, and that the computer architecture area is ripe for different evaluation methodologies. We make the case that high-level analytical models, possibly used in conjunction with simulation, can become the key drivers of future computer architecture performance analyses. We conclude with a call to arms to the performance evaluation community, to increase their involvement with computer architecture systems researchers and ensure that the above transition occurs in a timely and sustainable fashion.

1 Benefits of Modeling

One cannot overestimate the utility of modeling when designing an architecture (or any other complex system). The process of structuring the model, the process of thinking through abstractions, the process of describing how submodels interact and the process of identifying the metrics of interest all sharpen one's understanding of the system. Then, when that model is expressed in some form that can be evaluated, one achieves the additional benefit of exploring the quantitative effects of changing parameters, design alternatives, or operating environment. Before committing to building a system, one can come to some understanding of its projected behavior and check that behavior against design constraints and requirements.

In the area of computer architecture modeling is used extensively. However, the modeling is mainly done at a level that directly expresses the instructions and operations on functional units, address generation, caching behavior, and so on. These models by needs are evaluated by discrete-event simulation. With a few important exceptions the architecture community does not employ mathematical models at a higher level of abstraction.

2 Simulation Methodologies

We use the example of shared-memory multiprocessor architecture evaluations in the last ten years. During this period, analytical models have been used with success for individual parts of the system (e.g., the interconnection network), and in a few cases, for full system evaluation. Most full system evaluation studies, however, have used low-level simulation models. Focusing on these simulation studies, we can divide the period of the last decade into the following phases in chronological order:

- *Trace-driven simulation:* In the last decade, the first quantitative studies of shared-memory multiprocessors used trace-driven simulation, a natural extension of one of the most widely used techniques for uniprocessor simulation. However, in multiprocessors, trace-driven simulation generally does not capture the effects of synchronization and contention, since it does not allow these effects to alter the course of the execution.

- *Direct-execution simulation:* In contrast to trace-driven simulation, execution-driven simulation uses application executables to drive a simulator. This allows effects such as

synchronization and contention to change the course of the simulation, providing higher accuracy than trace-driven simulation. Execution-driven simulation is generally slower than trace-driven simulation. To improve simulation speed, a form of execution-driven simulation known as direct execution has been widely used for shared-memory simulation. The key property of direct execution simulation is that it simulates only some parts of the application, and executes the remaining parts directly on the simulation host machine. Most direct execution simulators for shared-memory systems only simulate the memory system. The contribution to simulation time due to non-memory instructions is approximated through static analysis and some dynamic information collected through application instrumentation.

- *Detailed execution-driven simulation:* Recent advances in processors have made currently used direct execution simulation techniques inadequate. Current processors exploit high levels of instruction-level parallelism, using complex features such as multiple issue, out-of-order issue, non-blocking loads, and speculation. These features lead to two difficulties for direct execution: (1) the contribution of non-memory instructions to execution time can no longer be statically determined as with previous direct execution simulators, and (2) the extent to which the processor exploits non-blocking loads directly impacts the memory system performance, but current direct execution simulators only model blocking loads. Therefore, there is a resurgence of detailed execution-driven simulators which model the entire processor in detail.

3 Simulation is Running out of Gas

Reliance on low-level simulation has to change, because exhaustive low-level simulation is "running out of gas". At the same time as Moore's Law is doubling the speed at which simulations can execute every 18 months, the overall complexity of the systems being built is increasing at an even faster rate. It is a race that complexity is winning. Simulating the performance of a new ILP (instruction level parallel) architecture on a SPEC95 benchmark by running the benchmark to completion is nearly unthinkable. The execution time required to simulate one CPU for a few seconds of simulation time is the better part of an entire day. The problem is exacerbated when one considers additional complexity such as parallel CPUs or complex I/O subsystems. The essential problem is that performance is affected by features at very different time-scales than the CPU clock, e.g., virtual memory thrashing, bursty demands from external sources, and the simple fact that programs have phases and the few seconds simulated may not intersect with critical ones. Moreover, simulations are complex programs that are hard to verify and validate. All of this points to a need to explore more abstract models—including ones analyzed mathematically rather than by simulation—for modern computer architectures.

4 A Case for Analytic Modeling

There are many examples of the use of analytic models in the analysis of computer (and system) architectures. We list a few of these, illustrating the diversity of both the kind of models and the areas of application.

Amdahl's Law (and Gustafson's Corollary) Amdahl's Law relates, with an explicit formula, the total speedup of an application to the speedup factor of an enhancement to the system and the percentage of the execution time of that application which can utilize the

enhancement. In the case of multiprocessor systems, Amdahl's Law shows that the speed which can be expected is limited by the percentage of the execution time which must be executed serially (i.e., on a single processor). In most cases, Amdahl's Law predicts that parallel processing (execution on a multiprocessor) will have only limited impact on reducing execution time. Gustafson's Corollary shows that if the data size of an application is increased as the number of processors in a multiprocessor system is also increased, then the corresponding increased availability of parallelism diminishes the relative limitation on performance induced by the serial bottleneck.

Asymptotic bounds analysis Asymptotic bounds analysis is a technique for producing upper bounds on throughput rates (or lower bounds on response times) in systems consisting of a number of interconnected components (queues). This analysis was used by researchers at IBM Research to analyze the impact of hot spots on multi-stage interconnection network performance. Special combining hardware was added to the NYU Ultracomputer to address this problem.

Analysis of bus-arbitration protocols Researchers at Wisconsin uncovered a flaw in the Futurebus arbitration protocol using analytic performance models, leading to the design of new higher-performance protocols. One of the new protocols was adopted in the Futurebus+ specification as well as in the experimental high-speed XUNET switch at AT&T.

CPU performance formula (Hennessy & Patterson) The CPU performance formula presented in the book Computer Architecture: a Quantitative Approach (J. Hennessy and D. Patterson, Morgan-Kaufman) states that the execution time for an application program is the product of the number of instructions executed and the clock cycles per instruction divided by the CPU clock rate. This formula has guided the design of many modern processors. In particular, it has been used to show that reductions in the clock cycles per instruction can lead to significant gains processor performance.

5 The Two Parachute Philosophy

In many projects, the system (sub-system, architectural feature) being modeled is very complex. As a result, the model of this system is also complex. With analytic models, a number of assumptions or simplifications may be required. In these cases, it is often valuable to have a "second opinion" on the validity of the results from the model. Simulation models are frequently developed to validate these results. Once validated, the analytic model can investigate regions of the problem space that are inaccessible to the simulation.

The simulation program expressing a complex system is itself large and complex, and (like any complicated program) subject to errors in design or implementation. The existence of an analytic model can serve to validate the answers produced by the simulation. Once validated, a simulation can be extended in complexity to investigate regions of the problem space where the analytic model Our point is that using two different models, validated against each other in problem regions where both apply, is like sky-diving with two parachutes. Wherever you fall, it is likely that one chute will open.

6 Other Evaluation Domains

As computer systems become more complex, new types of analysis problems continue to arise. These newly arising problems represent opportunities for performance engineers. Such emerging

problem domains include power consumption analysis and analysis for reliability and fault tolerance. While both areas have been of interest to specialized communities for some time, they are increasingly relevant to all architects as systems are scaled and begin to strain power and reliability limits.

Performance predictability, for real-time systems, is another opportunity for analytic techniques. In this domain, tight bounds on worst-case execution time are sought and analytic techniques are better-suited for bounding than many execution-based measurement or simulation techniques.

Finally, increasingly important problem domains call for performance analysis as well. These include such areas as application-specific architectures and hardware-software codesign.

7 A Call to Arms!

In response to simulation's severe limitations in the context of current and likely future architectural trends, we see an opportunity for analytical techniques to make significant contributions. For this reason, we issue the following call to arms:

- Members of the performance community can have significant impact by applying both new and existing models to the design issues that are currently disadvantage if they do not have an understanding of analytic modeling techniques.
- Direct collaboration between performance modelers and and increases the likelihood of "technology transfer" of modeling techniques from the performance to architecture communities.
- Another strategy to facilitate technology transfer is to publish important performance analysis results in systems conferences where system architects will see them. These include the International Symposium on Computer Architecture (ISCA) and the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS).
- A final aspect of improving the link between the computer architecture and the performance community will involve improving the instructional materials and techniques used in teaching performance modeling classes. In particular, textbooks and classes should motivate theory through examples and case studies that emphasize the relevance of analytical techniques to architecture and other systems areas.

Report of the Computer Resource Management Group

Rick Bunt, Derek Eager, Leana Golubchik, Gabriele Kotsis, Shikharesh Majumdar
Dick Muntz, Emilia Rosti, Giuseppe Serazzi, Evgenia Smirni

Research in resource management concerns the enhancement of system performance through effective management of resources such as CPU, memory, secondary storage, and network. The focus of this summary is the use of performance analysis in the evaluation and devising of appropriate software techniques for the effective management of these resources. Systems ranging from uniprocessors to networks of workstations (NOW), SMP's and MPP's are considered. A brief discussion of the use of performance evaluation in past research on resource management is presented in the following section. This is followed by a discussion of the future directions for performance evaluation-based research in computer resource management.

1 Past Impacts

In the late sixties and early seventies performance evaluation played an important role in the development of resource management strategies for single-CPU-based time-sharing systems. Some of the important contributions include the evaluation of CPU scheduling policies, the application of job shop scheduling techniques to computer systems, and techniques for virtual memory management. The use of operational analysis led to the development of tools that were used extensively by the industry and academia for performance evaluation of computer systems. Trace-driven simulation was a popular method for studying the paging behavior of virtual memory systems. Studies of memory reference strings were examples of early work in workload characterization.

Performance analysis has been instrumental in more recent research efforts as well. Examples include routing algorithms for inter-connection networks, task allocation, load balancing in distributed systems, scheduling in multiprogrammed parallel systems, and techniques for management of parallel I/O. Performance analysis was central to the development of scheduling techniques for hard real-time systems. Examples include rate-monotonic and dynamic scheduling techniques.

Various different performance analysis techniques have been used in the past for research on resource management. Analytic models have been successfully utilized in the performance evaluation of databases, operating systems (particularly CPU scheduling policies), and in the evaluation of alternatives for data management. Stochastic simulation has been used in the evaluation of many resource management strategies. A recent example is the analysis of processor scheduling policies for multiprogrammed parallel systems. Evaluation of management policies for memory hierarchies has typically been based on simulation. For example, trace-driven simulation has been used extensively in the design of cache management and page replacement strategies for virtual memory systems. Evaluating system management alternatives by running benchmarks and making measurements is effectively deployed in transaction processing systems. Recent research on parallel and distributed file systems have adopted the prototyping approach. Experimental file systems are implemented and measurements are made on the system for understanding their performance. An important aspect of performance analysis in resource management research is workload characterization. For determining their usefulness, the resource management strategies must be evaluated under a representative workload. As a result a significant amount of research on workload characterization has resulted in different problem domains. Examples of current efforts in this direction include characterization of Web traffic, of parallel I/O performed by scientific computations, and of multimedia traffic.

2 Potential Areas for Current and Future Research

Five different domains of resource management in which performance can be an important goal and performance evaluation can form an integral part of research on resource management are identified. These include:

- parallel processing (scientific applications in particular)
- multimedia systems
- transaction processing
- internet computing
- data mining

Issues that are worthy of research are identified in each of these domains. A number of these issues cut across the domains and are summarized in the immediately following paragraphs. This is followed by five sections, each of which provides a more detailed discussion of research issues that are specific to a given problem domain.

Common Research Issues

Handling Heterogeneity: Many different environments are characterized by a variety of different platforms. Resource management strategies have to be able to cope with this heterogeneity.

Adaptive Policies: Strategies that adapt to changes in system state and in workload behavior seem to be promising. Adaptive strategies for resource management warrant investigation.

Caching, Prefetching, and Buffer Management: Input/Output is an important component of system execution in all of the five domains. The impact of different strategies for caching, prefetching, and buffer management on system performance needs investigation: existing strategies need to be re-examined in the context of newer paradigms and new techniques are to be invented when necessary.

Data Distribution and Replication: Data access is an important part of system operation in many of these problem domains. Replication of data and use of appropriate techniques for distribution of data on multiple devices (sites) can speedup data access. Strategies for controlling data replication and managing data consistency with minimal performance cost as well as effective data distribution techniques are promising topics for future research.

Workload Characterization: All the five domains are relatively new application areas. Characterization of workload in each of these domains is necessary for resource management as well as for development of models for performance evaluation. Two important factors, heterogeneity and mixed applications, will contribute to the challenges for researchers in this area.

2.1 Parallel Processing of Scientific Applications

Parallel processing has been employed successfully for running a variety of different computation-and-I/O-intensive scientific applications. Traditionally a single scientific computation was run in isolation on a parallel system. Recent research is concerned with running multiple applications

on a multiprogrammed parallel environment. Economic reasons have been the driving force behind sharing expensive parallel machines among various users and behind the shift towards networks of commodity-off-the-shelf (COTS) platforms for parallel scientific computation.

Processor management and application scheduling in such an environment is a central problem. The processor allocation policies have to address heterogeneity of processors, interconnect, and workload. Adaptability, both in the application and in the resource management algorithms is a key factor for effective use of networks of workstations. In addition, techniques for performance tuning of parallel applications remains critical. Together with processors, the other critical resource is storage. Data distribution and layout on the disks, caching and prefetching policies that adapt to changing workload requirements, and data replication techniques are among the fundamental issues that current and future research should address. Workload characterization techniques that are able to capture the specific features of a multiprogrammed environment are needed.

A significant part of the existing research in the area is based on queueing models of systems and stochastic simulation. More recent studies have verified the relative performance of resource management policies and characterized the execution behavior of scientific programs through measurements made on experimental systems.

2.2 Multimedia Systems

Multiple media servers must support access to both continuous and non-continuous media. For continuous media, this requires satisfaction of real-time constraints on retrieval and delivery. In comparison to traditional information systems, additional challenges arise from the need to support real-time operations on large objects. On the other hand, additional flexibility arises due to the inherent redundancy within video, audio, and image data. Some of the important issues include QoS metrics, data layout, scheduling, admission control, fault tolerance, adaptivity, content based retrieval, data sharing, interactive workloads, synchronization of media streams, caching, prefetching, and buffer management. Some key questions concern the scalability of various resource management techniques and the economics and feasibility of the potential applications.

Performance evaluation techniques that have been applied in this domain primarily include simple quantitative models and stochastic simulation. Quantitative models include simple queueing models, Markov chains, resource allocation optimization models, and use of algebraic calculations (as in path length analysis, computation of utilizations etc.). The multimedia server area appears to be one in which performance analysis is providing valuable insight and is playing a major role in resolving the various resource management issues.

2.3 Transaction Processing

Although transaction processing is a fairly established field, recent transitions from a more centralized view to a multi-level client-server paradigm have led to the emergence of new performance concerns. Issues such as where to store data, where to process the data, replication of data, concurrency control/locking, synchronization, and the concomitant costs require reexamination in light of changing technology. The increase in size of systems results in a need for appropriate performance evaluation methods, for instance in demand analysis.

2.4 Internet Computing

One of the challenges in this domain is the existence of a workload that is a mix of application classes that may not have the same performance requirements: for example ftp is batch whereas realaudio is (soft) real-time. Application-level performance depends on the unreliable "best effort" technology on the network platform upon which it runs. How can the overall performance objectives be stated and achieved? For appropriate resource management we need accurate workload forecasts and good data on resource availability so that resources can be allocated effectively. This is easier in a closed environment but much more difficult in an open environment such as the internet. Appropriate techniques for measurement need to be developed for handling such an open environment.

Performance enhancements are possible through a number of mechanisms that include the following: proxies/replication/caching techniques for controlling network performance (admission control, flow control, congestion control etc.) Research is required in each of these topics to achieve high performance. All the different performance evaluation techniques, analytic modelling, simulation, and measurement are currently being applied to this task.

Security is also a major concern for internet computing applications, but there is a resource consumption (performance) cost for the user. What price are users prepared to pay for security? Modelling studies are required to explore the elasticity of this relationship. Application-level transactions may require the co-operation of multiple lower-level components. Who has the responsibility for ensuring that they perform as needed? Techniques for performance management of such multi-component applications need investigation.

2.5 Data Mining

Data mining (DM) is the search for patterns in (large) data spaces. The objective is to summarize information and to obtain knowledge in a human understandable form. Algorithms supporting this type of search can be both I/O intensive (e.g., OLAP) or computation intensive (e.g., market basket analysis). Another characteristic of data mining systems is the variability in response time demands. While some applications in DM are rather of a batch processing type (e.g., searching large data spaces off-line), others require short interactive response times (e.g., generating reports in OLAP).

Techniques applied in DM come from the field of statistics and/or machine learning and have to be adapted to cope with massive amounts of data. Particular performance problems include: design of mass storage, data layout, parallel (distributed) databases, and use of/modification to existing databases (e.g., delegating part of the DM job using SQL).

Report of the Performance and Software Group

Alois Ferscha, Daniel Menascé, Jerry Rolia,
Bill Sanders, Murray Woodside

We first identify the goals of *Software Performance Prediction* (SPP). Some of the issues discussed here have been first addressed in the book "Software Performance Engineering" by Connie Smith. We define *Software Engineering* as the systematic development of software and systems. Typically, these software and systems development teams are faced with many functional and qualitative issues that determine their design and development decisions. However for reasons discussed in this report, it is difficult or impossible to anticipate the performance behaviour of the resulting system.

The purpose of SPP is to provide timely quantitative feedback that: supports software and system design decisions; ensures performance requirements are captured and met; and anticipates performance bugs and helps overcome them. We define *Performance Engineering* as the techniques for: describing abstract models, performance evaluation techniques for predicting the performance of models, and techniques for measurement. SPP research has as its goals the development of useful PE methods for specific classes of software and systems.

1 Challenges of SPP

The following subsections identify several challenges that face researchers working on software performance.

1.1 Integration of PE and SE Processes

The successful integration of Performance Evaluation and Software Engineering poses several challenges. The first is due to the sheer size and complexity of large software systems with interacting components distributed over a large number of heterogeneous networks. SPP methods have to scale well with system size and complexity to be of practical use. They must also deal gracefully with the incompleteness and evolution of system descriptions.

To impact software development from its early stages and influence architectural design choices, SPP techniques and tools must complement the formal and sometimes informal software development processes used for specific projects. This informality and incompleteness raises barriers to performance evaluation.

Software designers must be able to use SPP techniques and tools in a seamless and transparent manner, all the way to the end of development. The same languages, formalisms, and abstractions used to describe software and the mapping of its components to hardware objects, should be used in the derivation and generation of performance models.

For the integration of PE and SE to become a reality, software engineers must be educated in some basic performance concepts and principles of software performance. The software development technologies they learn as students should incorporate SPP practices and methods. The students of today will be the software developers of tomorrow and the managers of tomorrow. They need to understand the importance of SPP.

SPP techniques must be fast and cheap to use. Most software projects are behind schedule. So, the incremental effort to include PE in the software development process should be minimal so that the risk of not using SPP techniques far outweighs any possible delays in the software development process.

1.2 System Size and Complexity

A complex system is an assembly of components which may come from many sources and may require different performance characterizations. Complex systems exhibit complex interactions within and between components. Some examples include interactions of transactions with cache effects and database locks. These interactions may modify in non-predictable ways transaction service demands.

Large complex systems lead to: limited perspective for any one designer — different perspectives must be integrated and conflicts resolved to obtain a complete view of the entire system; the need for performance prototypes with results to be integrated in models for the entire system; difficulties in managing and deriving meaning from the huge amounts of data that could be collected after system deployment.

1.3 Data Gathering/Interpretation

Data gathering to determine parameters for performance models, including service demands for processors, disks, networks, and other resources, requires measurements on an existing system. Software systems under development pose a particularly difficult challenge since measurements cannot be taken.

Various techniques could be used to estimate service demands for software systems under development. Measurements of similar existing applications can be used to estimate demands on new applications. Parameterized benchmarks on software components (such as database management systems, transaction processing monitors, and protocols) can provide estimates for service demands by setting the proper values for the parameters. Prototyping and taking measurements of critical components is a valuable technique to estimate demands if a small portion of the components are responsible for a large percentage of the service demands. Another approach is based on estimating the service demands from transaction descriptions combined with analytic models of components such as database management systems and middleware components. Finally, some guessing can be done based on past experience.

Some of the problems with data gathering for software systems come from the fact that developers hate guessing and prefer to work with exact data. Another problem is that too much data may be available to be understood. Data may be collected at the wrong level of abstraction or the only available data may be inappropriate for SPP. Lastly, software developers do not have enough time to talk to performance analysts because they do not feel they should spend time on these kind of activities.

Data gathering for the purpose of evaluation poses its own problems in large systems. These are due to large volumes, many types, multidimensional data spaces, sparsely populated data spaces, multiple time scales, and non-stationarity.

2 Emerging Software Features

New system features pose challenges to analysis. Examples include security mechanisms, reliability and fault tolerance, agent technology, brokers, and adaptive systems.

2.1 Modelling Techniques

System models are often composed of a mixture of submodels that capture the behaviour of specific parts of systems. The integrated model must capture the primary and secondary effects of system behaviour.

The following are some of the challenges that face the development and integration of the submodels, and their subsequent evaluation:

- transient system behaviour and the possible lack of a steady state;
- feature interactions between submodels;
- data and load dependent resource demand requirements; and
- interaction effects due to shared resources such as caches.

2.2 Model Building

Model building is the process of gathering information from software development artifacts, prototypes, and deployed systems, for the purpose of defining a models structure and parameters.

Challenges for model building include: incomplete or contradictory descriptions; keeping performance models in step with each other and the many other artifacts that describe the viewpoints of a design; the need to develop many types of models to support different kinds of decisions; and automation, to recognize classes of models and methods for deducing their structure and parameters.

3 Current Status

The following subsections provide a brief review of approaches by those conducting software performance research.

3.1 Model Building

A number of techniques have been developed for building models, which have been successfully used by developers. These have been used on quite large information systems projects and include: modelling from scenarios, modelling from design descriptions, and modelling from prototypes. The first two approaches use parameters obtained either by estimates based on experience (educated guesses) or from a resource demand data base for known components. The third relies on measurement to capture unknown structure, behaviour or parameter values.

Scenarios are used to define paths followed by the execution, and workload parameters are attached to the paths. The parameters must be estimated from experience.

Various software design (CASE) tools which record the definition of a design in a high-level form can also generate a performance simulation model. The user of the simulation must add resource parameters such as CPU resources and disk I/Os, and other workload parameters such as input rates. While simulation models are the most common output of such a model-generation step there are also research tools that generate analytic queueing models.

Measurement and emulation/simulation can also be used to help with the model building process. A performance prototype can be developed to capture key performance data used to create a model. The model is then modified to extrapolate for the purpose of understanding system behaviour.

3.2 Special Models for Software Systems

We have models that work for some problems. Ordinary queueing models have been used successfully for software systems that have rather simple one-at-a-time resource demands. Simultaneous resource demands and parallel sub-paths are examples of features that require more

sophisticated model such as extended queueing networks, Performance Petri nets, Stochastic Activity Networks, and Stochastic Process Algebras. Layered Queueing networks are a kind of extended queueing network defined especially to represent the fact that software servers execute on top of other layers of servers and processors, giving complex combinations of simultaneous resources.

Queueing-based models scale up well; however state-based models are better at representing intricate logical interactions and protocols. The state-based models suffer from state explosion, which may be reduced by using decomposition and hierarchical techniques. The reduction of state explosion problems and the extension of the queueing models are active research areas.

3.3 Data Problems

Dealing with mountains of data from large complex systems requires instrumentation conventions which are essentially non-existent, and may require data mining and statistical techniques which are beginning to be reported. Data gathering for resource demands of programs or components has much in common with running functional tests, and it may be possible to integrate it with software test environments. Resource demands may be stored in a repository and accessed for model building, when components are being used or re-used. Instrumentation is a continuing intricate problem, with relatively little being reported in the research literature. A determined effort, and cooperation with operating systems and run-time environment specialists, are needed.

3.4 Adaptive Systems and Environments

Self-managing systems are of growing interest and present many problems for SPP. Dynamic re-configuration for functional reasons and with respect to time-varying workload mixes and the availability of resources is likely to become more common. Feedback from performance evaluation techniques are needed to assess the impact of such re-configurations. To guarantee timeliness of reconfiguration activities is particularly challenging. A whole new class of methods supporting "*pro-active*" adaptivity based on the predicted system state at reconfiguration instant are needed to alleviate the shortcomings of "*re-active*" systems, which bring reconfigurations in effect after the system state has changed.

Examples of further challenges for SPP in this area include support for: identifying representative sets of control parameters for such systems — to avoid over-engineering with too many parameters; dealing with sensor data that may be statistically inaccurate — yet still used to make reconfiguration decisions; developing methods for Quality of Service (QoS) reservation and allocation.

3.5 Risk Analysis and Special Payoffs

When is it important to analyze the performance of a design? Virtually no work is being done on this, and it should be.

Performance analysis of software could offer more than just evaluations of a given design, and it might be able to support systematic methods for improving designs. Once a model is made, it can (in principle) be used to explore the design space and discover or rank optimizations. Very little has been done on this yet.