M.I.T. Laboratory for Computer Science
Computer Systems Research Division

March 16, 1978
Request for Comments No. 161

PERFORMANCE PROBLEMS IN DISTRIBUTED SYSTEMS

by Liba Svobodova

This paper will be presented at CIPS '78, the conference of the Canadian Information Processing Society. The conference takes place at the University of Alberta, Edmonton, Alberta, in May 1978. I had very little time to put this paper together, and as a result, it is rather shallow. Although this is the final form as far as this particular conference is concerned, I would appreciate comments, since I may do some additional work in this direction.

PERFORMANCE PROBLEMS IN DISTRIBUTED SYSTEMS

Liba Svobodova
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract

There is a strong indication that for many applications distributed computer systems will replace conventional computer systems built around a large central processor, and that new applications will emerge based on distributed information processing. This move towards distributed processing has become feasible mainly because of the rapidly dropping cost of computer hardware and the increasing power and flexibility of mini and microcomputers. Lower cost of computer hardware also ought to have an impact on the evaluation of computer system performance, making utilization of hardware resources a much less important factor than it has been so far. Performance related efforts should concentrate on the aspects visible to the user, that is, response time, reliability and availability.

Due to the distributed nature of these new systems, new problems are introduced that require new measurement and analytical approaches. A primary problem in distributed processing is proper partitioning of computations and data bases among the processors in the system. Correspondingly, measurement tools need to be distributed, based on the characteristics of the measured process. This paper discusses some of the problems in detail and examines how the conventional computer performance evaluation tools and techniques need to be expanded to serve the requirements of distributed systems.

Introduction

Analysis and evaluation of computer systems has become progressively more difficult as more sophisticated systems were introduced, and more emphasis put on interactive use. Today, computer performance evaluation is a demanding field that requires a deep understanding of many factors and their mutual interaction: the inner system mechanisms, both hardware and software, the processing requirements, the operating environment, but also users' habits, preferences, and adaptability to system changes, and the effects of poor performance on the user's productivity and well-being. To a great degree, computer performance evaluation is still an art. Yet, significant progress has been made in understanding the performance problems and developing techniques and tools to aid in performance evaluation.

However, as the techniques and tools evolve to solve existing problems, new problems emerge due to evolution of computer systems. For several years, there has been an indication that the natural evolution leads towards distributed systems. The primary new problem in this type of systems is how to distribute computations and data bases. More generally, it is necessary to understand the measurement tools needed for distributed systems, generation of test workloads, and applicability of analytical modeling and simulation.

While for many applications a distributed system is a more natural realization than a system based on a single large central processor, undoubtedly there are applications that do require just such a centralized

system. The decision of what type of system is more appropriate is a difficult one. It concerns more than just performance and cost of the hardware and software. In particular, it may have a strong impact on how people in an organization perceive and relate to their computer system; this in turn has an impact on the effectiveness of the people involved, both the users and the managers.*

The area of "distributed systems" has become a popular source of systems research projects. It has also become an important term in marketing computer equipment. Unfortunately, because of this popularity, the terms "distributed systems" and "distributed processing" are frequently misused, often referring to such conventional concepts as remote job entry, use of terminal concentrators, or multiprocessor organizations. Thus before it is possible to talk about performance problems in distributed systems and the applicable tools, it is necessary to discuss what distributed systems are and what motivates their use.

Characteristics of distributed systems

Distributed systems discussed in this paper can be defined loosely as organizations of highly autonomous information processing modules that cooperate in a manner that produces an image of a coherent system on a certain defined level. The autonomy is the key characteristic that eliminates most multiprocessor organizations from the class of distributed systems. Certainly, a distributed system has more than one processor; however, in a distributed system, the processors are highly independent, each having its own

---

* Of course, managers can also be users.

primary memory, possibly even some secondary storage, and its own interface through which it communicates with its environment (e.g., user terminals, sensors). These individual subsystems, called members in this paper,* are connected by a communication network. Each processor has access to its own memory only; that is, in a distributed system inter-processor communication is possible only by explicitly exchanging messages, not through shared memory. Finally, in a distributed system, control of the system's internal resources is decentralized.**

The type of communication network used in a distributed system will vary widely with the georgraphical dispersion of the system, the purpose of the system, and the performance requirements. Computer networks are often categorized as global networks and local networks, where global networks span large geographical distances and local networks connect computers located in the same building or a building compound. The best known example of a global network is ARPANET [24], a store-and-forward packet switching network. Communication processors on the ARPANET (IMPs) receive packets from the ARPANET hosts or other IMPs, and forward them to the destination host or another IMP on the path towards the destination. This technique is unnecesarily expensive for local networking. In situations where individual nodes are sufficiently close, a single communication channel (bus) accessible to all nodes is sufficient. Since it is assumed that a distributed system

---

* In computer networks, such subsystems are usually called "hosts". The term "member" is used here to suggest participation and cooperation.

** There still may be a "central" component in such a system for a certain class of decisions or services (e.g., central file system), but it is assumed that most of the decisions concerning requests coming from the member's own terminals are made locally.

does not have a central control element, each member has to be able to decide for itself when it can use the communication channel. Local networking has become quite popular in recent years. The two best known local networks are the Ethernet [19], a cable that operates as a bidirectional broadcast contention (random access) channel,* and the DCS ring [9], where a control token is passed from one member to the next member to indicate when a member is allowed to transmit a message. In a local network, it is possible to use a channel with a very high bandwidth such that the delays due to the communication network are practically negligible. In geographically distributed systems, the delays in the communication network cannot be overlooked. The significance of these delays is especially felt when it is necessary to synchronize operations of several distant members.

A computer system cannot be viewed separately from the application it is to support. The hardware and software organization ought to reflect the structure and the needs of the containing system (bank, factory, university, etc.). As said earlier, for many applications, a distributed system represents a more natural realization than a system built around a single large central processor. The reasons are many, but the primary forces are functional separability and a non-uniform distribution of the use of data bases. Functional distribution means that different processors support different services. Such systems seem natural for control of industrial

---

* The "random access" approach (that is, a member wanting to transmit a message can do so immediately, regardless of the plans of the other members) is used only when the Ethernet is found to be idle. Each member monitors the state of the Ethernet, and defers to passing messages. Also, once it starts transmitting, it checks for possible collisions of its own message with messages from other members. If a collision is detected, the transmission is aborted.

processes, where different processors control different parts of a process, or in such systems as aircraft, where different processors process information from different sensors. However, this approach seems to be also advantageous in service sectors such as banking. For example, the First National City Bank (based in New York City) separated different kinds of services provided for the corporate customers on different computers systems.* This move resulted in a lower cost for computer services, since separation greatly reduced the complexity of information processing within the bank.

Another category of distributed systems is a system where individual processors support the same services but on a different part of a data base. A typical example is a bank with many branch offices. Each branch has its local accounts, but it should be able to serve a bank's customer whose account is at another branch. Since such remote requests are much less frequent than manipulation of the local accounts, partitioning of the bank's accounts data base (that is, maintaining accounts on a computer at their local branch) is a natural approach. It needs to be said, however, that the division between functional distribution and data base distribution is not clean; in most cases, a distributed system will to some extent include both.

In addition to the "naturalness" of distributed systems for some applications, distributed systems offer several important advantages over central processor systems. Availability of information can be increased by replicating it in several members. This arrangement not only increases the

---

* The computer facilities in the First National City Bank do not represent a distributed system, since individual computers dedicated to specific services are not connected. However, the bank is building a computer communication network that will tie together all the existing systems, thus setting the ground for distributed processing.

access bandwidth to the information, but in case of a failure of one of the members or some communication links, the information remains accessible. It is also believed that distributed systems provide a better environment for protecting information stored in the system and coping with run-time errors resulting from hardware failures or residual design and implementation errors. In support of this claim, two factors should be considered:

i. the actual physical separation of indepedent or loosely coupled computations and information that belongs to different users

ii. as a possible side effect of such separation, a reduction in the software complexity

The physical boundaries of individual members provide "firewalls" that (if properly designed) will prevent spreading of errors originating from a particular member to the rest of the system and protect information stored at individual members from unauthorized access or modification by other members. As the most severe protection measure, a self-contained member can be guaranteed privacy during some sensitive operations by physically detaching it from the rest of the system. Also, distribution reduces the level of hardware resource sharing, and consequently, may reduce the complexity of software for resource allocation, scheduling, and protection. Lower software complexity makes verification of design and implementation more feasible.

Finally, distributed systems offer expandability. As more users join the system or new services are added, it is not necessary to make any physical replacements; rather, one or more new members need to be added to the system. Distributed systems can grow more gradually than systems with a large central processor.

Distributed systems have been emerging in three different ways:

i. Distribtion of an existing information processing (or process control) system: new hardware (and software as necessary) replaces the old organization based on a large central processor to support (initially) the same set of applications.

ii. New systems: a distributed system is designed to support a new application, for example, office automation.

iii. Integration of existing computer systems: self-contained computers supporting specific (possibly diverse) applications are connected by a communication network with the aim of sharing hardware resources and information.

The last category often represents a very difficult problem. Since individual self-contained computers evolved in their own ways, building a coherent system that shields the user from the idiosyncrasies of individual members may be an impossible task. Yet there seems to be a great need for building new systems through integration. The ARPANET project demonstrated the feasibility of a computer network composed of diverse computer systems. The Resource Sharing Executive (RSEXEC) [29] and the National Software Works (NSW) [20] are distributed systems built on the top of the ARPANET. Computer manufacturers feel the pressure to support computer networking; DECNET [30] was developed to facilitate building of networks that may include any type of system manufactured by the Digital Equipment Corporation. However, many underlying problems of making a network of highly autonomous computer systems look like a coherent system for queries or computations that require participation of several such systems have not yet been mastered. One problem that repeatedly

comes up in decisions what mechanisms should be provided to the programmers to support implementation of distributed applications is the lack of understanding of such applications. This is of course a well known problem, that also hinders evaluation of computer system performance.

Performance evaluation issues

The very fundamental problem of computer performance evaluation is the problem of defining "performance". Performance of a system can be discussed from two different positions:

i. how effective the system appears to its users

ii. how efficiently the system uses its internal resources in order to satisfy the demands of its users

"Effectiveness" is very difficult to measure, not speaking about trying to express it in a quantitative form. Thus performance evaluation usually concentrates on measurable quantities that reflect the system efficiency. However, effectiveness has to be considered, especially when choosing a system for a new applicaton. Effectiveness is a result of the set of tools provided to the users, characteristics of the actual physical interface between the user and the system, system reliability and response time to user requests, but it also may be influenced by such intangibles as how much control the user feels to have over the execution of his programs. Distributed systems seem to offer definite advantages regarding the effectiveness [6].

The question whether a centralized or a distributed system should be adopted for a particular environment and application has been addressed by many people, including professionals from both the computer field and

management. Advantages and disadvantages concerning management of system development and system operations, personnel, cost, and political problems are usually discussed [25]. Differences between distributed and centralized systems in the above categories make the selection process very difficult and definitely beyond the scope of the conventional tools (benchmarks, simulation) used to select a system from among similar system organizations offered by different manufacturers. Finally, even if it is determined that a network of small scale computers will serve the needs of a particular environment better than a system based on a large central processor, different applications executed on the system may need a different degree of distribution, that is, each application ought to be analyzed separately for its "distributability".

Tuning and load sharing

The widest class of performance evaluation projects is concerned with system "tuning". Tuning on conventional computer systems generally strives for the highest possible utilization of all resources (maximize throughput), while guaranteeing reasonable response time to the users. This is done by balancing the load either externally (the order in which jobs are submitted to the system) or internally (the order in which jobs are made eligible for multiprogramming, the order in which jobs are assigned to a processor). The other aspect is the criticality of individual jobs. The more critical a job, the higher priority it should be given when assigning resources. This policy of course works against the load balancing attempts. The importance of the speed of the response vs. system throughput has been stressed by several people in the past, based mostly on the "cost" of a programmer compared with the cost of the system [8]. With the cost of the hardware rapidly decreasing, it makes sense to place even more emphasis on response time (and other

effectiveness aspects). That is, individual members should have sufficient capacity of provide good response time; it should not matter if they are underutilized.*

One incentive for building distributed systems by integration is the possibility of load sharing. That is, if one member is overloaded, requests from its local users can be sent to another (lightly loaded) member for execution. How dynamic such a load sharing scheme can be made is a difficult question. In a geographically distributed system, shipping an entire job to another member may be costly. Also, because of the delays, it is difficult to know the current state of all suitable members; the member that was chosen to come to the rescue can suddenly become heavily loaded, or may even fail. Tuning this kind of system may be an impossible task. However, in a more static situation, load sharing might be clearly desirable and easy. For example, computers in different time zones experience peak loads at different times. It might be possible to work out a schedule such that during certain hours some of the capacity of a specific computer would be made available to users from the busier time zone. Still, the communication cost of shipping all needed programs and data to another member may render this strategy infeasible.

The crucial problem in design or selection of a computer system, and also in later tuning efforts, is understanding how the system will be or is used.

---

* Another class of distributed systems that is coming into existence is a network of personal computers. A personal computer can be used by a single individual only; a personal computer is always available, and ought to provide good response to tasks performed by the user. Utilization of a personal computer is completely irrelevant. Availability and quick response is what matters.

From the performance evaluation point of view, the problem is to understand the demands on system resources, that is, the workload. Characterization of workload is possibly the most difficult and certainly the least understood part of a performance evaluation project. This is because each user community, each environment, will represent a different workload. Workload is usually characterized by the distributions of requests for individual resources and the length of time needed to satisfy a request. In distributed systems, because of potentially long communication delays, the frequency of requests that cannot be satisfied locally* and must be sent to a remote member is a very important parameter.

Reliability vs. speed

Another influencial factor in computer system evaluation is reliability. In the literature on performance evaluation, reliability (mostly in the sense of availability, that is, the percentage of time the system is available to its users) is sometimes listed as a measure of performance. Reliability, however, is more than that the system is operational and available to its users. Reliability also means that while the system is operational, it operates correctly, and that the information entrusted to it is not lost or damaged. Reliability often represents a tradeoff in terms of the efficiency of a centralized system, and the significance of this tradeoff seems to be more prominent in distributed systems, in particular, systems supporting distributed data bases. Physical distribution of computational tasks and information is both a means for achieving robustness and a source of new

---

* A request may not be satisfiable locally because the particular member does not have the data requested. Note: this is conceptually an entirely different problem than load sharing.

reliability problems requiring new solutions. It can be assumed that the most common failure in a distributed system is that one of the members involved in a communication does not respond any more. A failure of a particular member should not affect the rest of the system, that is, cause a failure of another member or unnecessarily delay those operations of other members that do not depend on the failed member. By properly engineering the system, the first possibility can be eliminated. For example, communication protocols have been designed that are prepared to deal with the problem of a non-responding member [23]. Paradoxically, the possibility of a delay caused by a failure of some member may be a result of an attempt to enhance reliability, in particular, availability of information. As mentioned earlier, availability can be increased by providing multiple copies of files at different members. These copies ought to be kept consistent in face of concurrent asynchronous update requests originating from different members. The protocols that update all of the copies simultaneously are very complex and the possibility that a failure occurring during such an update will cause a long delay (possibly infinite, if no external means for recovery are provided) to unrelated requests is very real. A similar problem exists when it is necessary to atomically update different files maintained by different members. An atomic update means that either all or none of the changes requested are done before anybody else is allowed to see the affected data. Current research in this area concentrates on minimiznig the time window in a distributed update during which a failure of one of the participating members will lead to a situation when it is impossible to make a decision whether the update should be completed or aborted, thus delaying other requests until this situation is resolved [15], [10].

Partitioning of computations and data bases

Partitioning of computations and data bases is definitely the most fundamental problem in a distributed system. In fact the very decision whether a distributed or a centralized approach should be used depends on the feasibility of partitioning to more less independent units. If it is not possible to identify modules such that frequency of communication between modules is relatively low compared to the work done by individual modules, there is no point to think about distribution.* Similarly, when information in the data base is used approximately with the same intensity by the entire user population, the data base should remain centralized. It has been suggested that distribution of a data base is preferable if 80% of the access requests to each partition is generated locally [11]. This rough rule has not been verified yet; also, the cutoff may be different for different applications.

Existing systems are an invaluable source of information for designers of new systems. Once it is understood how computer systems are used (and misused), better hardware and software support can be planned. Measurements of various applications run on existing computer systems provide a very good insight. Such empirical studies are also needed to assess the "distributability" of an application. One such study was performed on the Multics system at M.I.T.; a series of experiments was conducted to determine the degree and type of information sharing among the Multics users [21].**

---

* That is, the type of distribution considered in this paper.

** Unfortunately, the degree of sharing in the Multics system is extremely difficult to measure; only rough estimates can be obtained.

Mathematical models have been developed that attempt to find the optimal allocation of files in a distributed system [1], [3], [5], [16]. The usual considerations are the cost of the communication and the cost of storage. The models assume that the costs (measured in delays) of reading and updating individual files is known. These costs will depend not just on the location of the files, but also on the type of communication network used and the update strategy employed. Even if these costs are known, solving such models is a non-trivial problem. Some helpful simplifications can be made for a localized distributed system, such that for some update strategies, the model is easy to use. However, for more complex update strategies, for example, the atomic updates described earlier, the costs that are the parameters of the model are very difficult to estimate.

More comprehensive analysis is needed to aid in the decision whether the data base should be centralized or distributed. In such a decision, not just the cost of performing an update, but also the cost of implementing update protocols for a distributed system should be considered. In addition, the very fact that the system supports a complex update protocol may result in an overhead that is independent of the actual use of that protocol. If it is preferable to use a distributed data base, it is quite possible that the "optimal" distribution is obvious, that is, following the theory of "naturalness" of distribution, the files that should be local to a particular member are clearly distinct from local files of other members, where this distinction is determined by the responsibilities of and the work done at each of the members.

Performance evaluation techniques and tools

Performance evaluation methods traditionally have been divided into three categories: measurement, analytical modeling, and simulation. In most performance evaluation projects, some combination of all three types is usually used. In particular, measurement and modeling are complementary processes. A model provides a framework for measurement, while measurement provides data for validating the model and verifying correctness of predictions made with the aid of the model. As it is well known from performance studies of conventional systems, it is usually impossible to incorporate all the relevant aspects into a single model. Performance evaluation has to be performed on may different levels, incorporating the results from one level into a model of a higher level. For example, it is necessary to combine the characteristics of the primary memory and the paging device with a particular paging algorithm and particular page demand pattern to be able to understand how much primary memory is needed per user and how to model the usage of the processor, given that rescheduling takes place at an occurrence of a page fault. The processor may then be a part of another model that includes peripheral devices and user terminals, such that user demands on the processor are the results derived from the previous model. Such a hierarchical approach of course is needed in distributed systems too. In addition to various levels within individual members, it is necessary to incorporate into the final model the communication network.

Analytical models

An analytical model is a mathematical expression of the behavior of a system derived from a model of the system and a model of the system's

workload. Analytical studies of computer performance require many simplifying assumptions about the modeled system and its workload, since the class of problems that is solvable with existing mathematical methods is rather limited. Despite the forced simplifications, analytical models play an important role in performance evaluation: they provide insight and a relatively quick first-order approximation of system performance. Analytical models are more suitable for studies of individual system components. In distributed systems, a good candidate for mathematical analysis is the communication network. The communication network is a shared medium and its performance depends on its physical organization, bandwidth of the physical communication medium and the multiplexing strategy used. A large body of analytical models and results for computer networks is assembled in [12].

Although new techniques have been found that facilitate solution of large queueing networks, this does not seem to help much with performance analysis of distributed systems, since the queueing networks solvable with these techniques are not sufficiently general [4]. In a distributed system, each member has its own source of input and may generate requests for services provided by other members; that is, requests to each individual member are coming from outside the system (local requests) and from other members in the system via the communication network (remote requests). In addition, a reply (which results in a creation of a new task at the member that sent the original request) has to be sent back in response to a remote request. This situation for a two-member network is depicted in Figure 1. In a more general case, queueing would also occur for the communication network. It may be possible to estimate throughput and response time of individual members by reducing the model to a single member and the communication network, where the

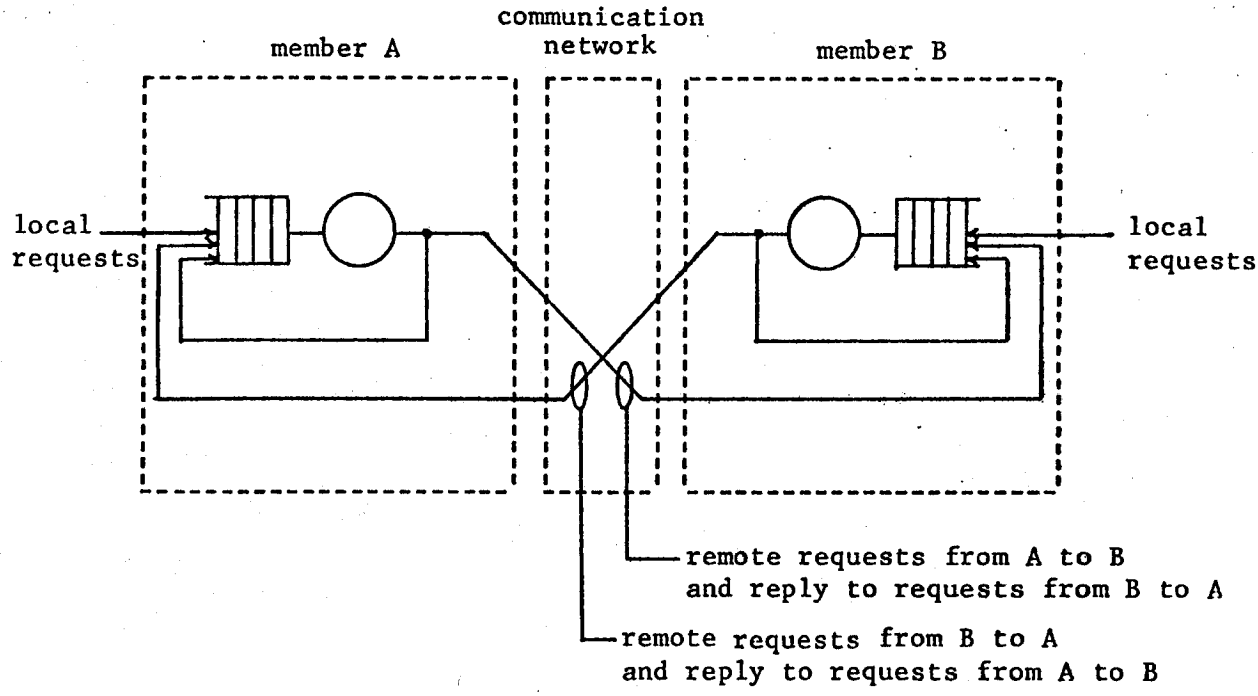

Figure 1: A queueing model for a two-member distributed system.

remote requests are represented by independent Poisson processes [2]. How realistic it is to use such an approximation can be decided only through an empirical validation. In my opinion, to get a reasonably good understanding of the behavior of the type of model shown in Figure 1, it is necessary to use simulation.

Simulation

The technique of simulation can be viewed as a combination of modeling and measurement. It again requires a model of the system and a model of the system's workload, but these models can be more complex and more detailed than if the performance is to be determined mathematically. A simulator simulates the behavior of the system under the specified workload and collects the data required for performance evaluation. Depending on the type of system simulated, a simulator may be a very large and complex program. This can be expected in simulation of distributed systems, in particular when the system has many members, and each member may behave differently. As with any large piece of software, a simulator should be designed in a structured way.

One approach to structuring a complex system is to use abstractions. An abstraction is a construct that represents objects of a particular type (e.g., queue) and supports a well-defined set of operations on those objects (e.g., enqueue, dequeue, test if queue is empty). These operations are the only operations that a user (a higher level abstraction) of such an object can apply to that object.

The representation of the object is completely hidden to the user.* For example, the system model is the highest level abstraction. The model is represented by lower level abstractions -- models of individual components that can be representated by yet lower level components until finally at some level the representation has to be made up of the basic data structures of the computer system on which the simulator is implemented. Since the same type of component may be needed in several places in this hierarchy, it makes sense to provide an abstract type, a construct that serves as a prototype for creating specific objects of the given type.

To study different configurations, it should be possible (and easy) to assemble and simulate different system models from the models of the components. Again, this is particularly important in configuring distributed systems, where it is necessary to experiment with a varying number of members and different communication networks [13].

Computer system simulators are in general implemented as discrete event simulators. A discrete event simulator maintains its own clock (simulation clock) that is simply advanced to the time of the next event that needs to be simulated. That is, the clock is incremented by a variable amount corresponding to the time that in the real system would elapse between

---

* Abstractions can be classified as data abstractions, function abstractions and control abstractions. Most programming tasks seem to involve operations on (possibly complex) data structures, yet data abstractions are not supported by conventional programming languages. New programming languages are currently being developed that support data abstractions (CLU [17], Alphard [31]). It is interesting that while this approach has not found strong proponents in the field of simulation, it was a simulation language, Simula [7], that first used data abstractions. However, Simula does not enforce the constraint that access to objects may occur only through a specified set of operations.

subsequent events, where an event is a change in the state of the selected system model. In a model of a distributed system, it may be necessary to simulate both local events, that is, events that affect one member only (e.g., new local request) and global events that need to be recognized on the system level (e.g., initiation of a remote request that results in a request for the communication network, and the appropriate remote member). As it is in the real system, the actual ordering of two local events belonging to different members is irrelevant. However, the simulator has to keep track of all events to be sure that the global events are scheduled properly. While it may be possible to simulate a distributed system on a multiprocessor system (e.g., each member simulated by a different processor), the control of simulation, in contrast to the operations of the real system, needs to be centralized; all processors must always see the same simulated time.

One of the most important problems in both analytical modeling and simulation is proving the validity of the model, that is, proving that the model is an accurate representation of the evaluated system. A performance model is usually considered representative of the system in question if the two yield the same performance. Validity of a model that is supposed to represent a concrete existing system can be tested against measurements obtained from the real system. Simulation is often used to validate analytical models, since the simulated model can be made more realistic than the model used to derive the mathematical solution. Simulation, however, has another problem: the simulation algorithm or its implementation may be incorrect. Thus, in the case of simulation, it is necessary to verify that the simulator correctly simulates the specified model. The other question of course is if the specified model (correctly simulated) is a sufficiently good

representation of the studied system. The verification of the correctness of the simulator can be achieved by carefully designed tests, the correct outcome of which can be determined outside of the simulator [13].

Measurement

The most fundamental technique used in performance evaluation is measurement. Measurement can be used to assess the effect of different configurations or resource allocation strategies on the performance of the system, but primarily measurement is essential to understanding how a particular system is used, and how users' requests translate to requests for the internal system resources.

A distributed system has many different active points that together effect the performance of the entire system. Thus to understand the system's performance, it is necessary to monitor all of those parallel activities. Actually this problem arises even in systems with a single central processor; as a well known example, movement of data to and from a disk device normally proceeds in parallel with central processor activities. To understand how the operations of the central processor and the disk are related (under particular workload), the central processor and the disk controller should be monitored in parallel. Usually, a hardware monitor is needed for such combined measurement [27].

In a distributed system, there may need to be a monitor for each member, especially if the distribution is also geographical. This local monitor should be able to monitor

i) tasks generated and processed locally

ii) requests sent to other members

iii) requests received from other members

To understand the behavior of the distributed system as a whole, it may be necessary to obtain a trace (history) of relevant events observable by individual members such that these individual histories can be later merged and analyzed. Successful merge of histories of individual members requires that events be timestamped such that the timestamps reflect the true ordering among global events, and ordering of global events to local events of members affected by the global event. This means that the clocks used to timestamp measurement data should be synchronized.* Of course, the timestamps are needed not just to correctly order the observed events, but to measure the time duration of various tasks and waiting periods. The measurement data can be accumulated at individual members and the entire file then shipped to a measurement center where required analyses are performed. It is assumed here that the measurement center is a part of the system, that is, accessible through the same communication network as the regular members. An alternative strategy is to report each interesting event to the measurement center by sending it a message [18]. It is also possible to have several measurement centers that collect different types of events (for different types of analyses). Of course, to use such a strategy, the event messages must not interfere unduly with the "useful" traffic in the communication network, that

---

* Depending on how accurate the individual clocks are and to what degree they have to be synchronized, the synchronization may be a difficult problem by itself [14].

is, it is feasible only for some relatively low-frequency events.

If the communication network is of a broadcast type, that is, all messages from all member anywhere in the network, much of the measurement concerned with global events can be done by a single "spy" monitor that monitors all traffic passing through the communication network [28]. Such a spy monitor would have to be fast enough to absorb information at the rate equal to the transmission rate of the communication network. In addition, a substantial amount of storage may be needed to store all information until it can be properly analyzed. Of course, to understand what is happening in individual members, it is still necessary to have a local monitor for local events.

A system can be measured under its real workload, but often it is necessary to use artificial workloads [27]. Alternate choices in system configuration and resource allocation policies must be evaluated for the same workload, or the differences in the workload must be factored out prior to assessing the effect of the system changes on performance. The latter is often a difficult if not an impossible problem. The purpose of artificial workload is to provide a controllable reproducible environment for performance optimization studies. Artificial workload is also used for comparative evaluation of different systems. For an interactive system, the workload must represent the users as well as their programs. Such workload is generated by a special generator that simulates the actions and random delays (think and type time) of the users.*

---

* To understand the system performance, it is necessary to understand the characteristics of the workload. If the workload is generated artificially, such understanding may be thought of as being implicit. However, the

In a distributed system, each member should be exercised in a way similar to its actual use, but at the same time in a way that produces a reproducible workload in a global sense. Thus the actual implementation of the workload generator requires a special driver for each member, or at least several members, depending on the purpose of the measurement experiment. The design of such a distributed driver is an interesting project by itself. It is of course possible to turn on each of the local drivers individually, wait for some time to be sure that all of them have initialized properly and are indeed generating requests, and start measuring. A better alternative, but more difficult to implement, is to start the measurements automatically after the individual drivers have confirmed that the initiation is complete.

Extensive measurement facilities were designed for some computer networks, but the measurements have been concerned with the performance of the communication network rather than the performance of the entire system under some distributed application. The ARPANET designers realized the need for measurement early in the design of the network. Measurement capabilities were built into the IMPs and also into selected hosts [12]. The measurement tools include artificial message generation, a trace mechanism that keeps track of messages as they pass through a sequence of IMPs, accumulation of various statistics and snapshots of queue lengths, and finally a package for control, collection and analysis of data sent from individual IMPs to the Network

---

workload used to drive a system during a measurement experiment may be specified in terms of user commands; for performance analysis the workload should be specified in terms of demands for the internal system resources. Thus even when artificial workload is used in an experiment, it may be necessary to measure the actual characteristics of the workload on the level of interest.

Measurement Center at UCLA. A comprehensive computer network monitoring system is being implemented at the University of Waterloo [22]. The basic elements of this system are modular hybrid monitors, each being able to monitor a computer system and a set of communication links to terminals and other computers. These hybrid monitors are remotely controlled from the network measurement center. Each monitor has access to a single standard clock. An artificial workload generator (network traffic generator) has also been implemented that can simulate several interactive users. Measurement tools are also being designed for DECNET, specifically, a distributed message generator, where several members participate in the workload generation [26]. Finally, the Laboratory for Computer Science at M.I.T. is building a local (broadcast) network that will serve as a base for a distributed system. Measurement facilities utilizing the particular structure of the network are currently being planned [28].

Conclusion

This paper is by no means a complete survey of performance issues in distributed systems. Nor is it a complete survey of existing projects in this area. A great amount of work has already been done; the references in this paper cover only a relatively small fraction of it. Yet distributed systems are a fairly new field and many of the associated performance problems probably have not been discovered yet. It is first necessary to get more experience with design and use of distributed systems. The same has to be said about the performance evaluation techniques and tools. This paper made some suggestions concerning the characteristics and implementation of various tools, but only experience with working distributed systems will show what is really needed.

References

[1] Akoka, J., Chen, P., "Optimization of Distributed Database Systems and Computer Networks," M.I.T. Sloan School of Management, WP916-77, March 1977.

[2] Babic, G.A., Liu, M.T., Pardo, R., "A Performance Study of the Distributed Loop Computer Network (DLCN)," Proc. of Computer Networking Symposium, NBS, Gaithersburg, Maryland, December 1977, pp. 66-75.

[3] Casey, R.G., "Allocation of Copies of a File in an Information Network," Proc. AFIPS SJCC 1972, pp. 617-625.

[4] Chandy, K.M., "Models of Distributed Systems," Proc. of Third International Conference on Very Large Data Bases, October 1977, pp. 105-120.

[5] Chu, W.W., "Optimal File Allocation in a Multicomputer Information Center," IEEE Trans. on Computers, Vol. C-18, No. 10, October 1969, 885-889.

[6] D'Oliveira, C.R., "An Analysis of Computer Decentralization," M.I.T. Laboratory for Computer Science TM-90, October 1977.

[7] Dahl, O.J., Myhrhaug, B., and Nygaard, K., "The SIMULA 67 Common Base Language," Publication S-22, Norwegian Computing Center, Oslo, 1970.

[8] Erikson, W.J., "The Value of CPU Utilization as a Criterion for Computer System Usage," Proc. Second SIGMETRICS Symposium on Measurement and Evaluation, September 1974, pp. 180-187.

[9] Farber, D.J., Larson, K.C., "The Structure of a Distributed Computer System," Proc. Symposium on Computer- Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, April 1972, pp. 21-27.

[10] Gray, J.N., "Notes on Data Base Operating Systems," Advanced Course on Operating Systems, Technical University, Munich, Germany, 1977.

[11] Howard, P.C., "Performance Considerations for Distributed Data Processing Systems," Proc. SIGMETRICS/CMG VIII Conference, November 1977, pp. 237-246.

[12] Kleinrock, L., Queuing Systems, Vol. 2: Computer Applications, John Wiley & Sons, 1976.

[13] Krizan, B.C., "A Minicomputer Network Simulation System," M.I.T., Department of Electrical Engineering and Computer Science, M.S. thesis, September 1977.

[14] Lamport, L., "Time, Clocks, and the Ordering of Events in a Distributed System," Massachusetts Computer Associates Technical Report CA-7603-2911, March 1976.

[15] Lampson, B., Sturgis, H., "Crash Recovery in a Distributed Data Storage System," Xerox Palo Alto Research Center, 1976 (to appear in Comm. of ACM).

[16] Levin, K.D., Morgan, H.L., "Optimizing Distributed Databases -- A Framework for Research," Proc. AFIPS NCC, 1975.

[17] Liskov, B., Snyder, A., "Abstraction Mechanisms in CLU," Comm. of ACM, Vol. 20, No. 8, August 1977, pp. 564-576.

[18] McDaniel, G., "METRIC: A Kernel Instrumentation System for Distributed Environments," Proc. of Sixth ACM Symposium on Operating Systems Principles," November 1977, pp. 93-99.

[19] Metcalfe, R.M., Boggs, D.R., "Ethernet: Distributed Packet Switching for Local Computer Networks," Comm. of ACM, Vol. 19, No. 7, July 1976, pp. 395-404.

[20] Millstein, R.E., "The National Software Works: A Distributed Processing System," Proc. of ACM Conference, October 1977, pp. 44-52.

[21] Montgomery, W.A., "Measurement of Sharing in Multics," Proc. of Sixth ACM Symposium on Operating Systems Principles, November 1977, pp. 85-90.

[22] Morgan, D.E., Banks, W., Goodspeed, D.P., Kolanko, R., "A Computer Network Monitoring System," IEEE Trans. on Software Engineering, Vol. SE-1, No. 3, September 1975, pp. 299-311.

[23] Reed, D.P., "Protocols for the LCS Network," M.I.T. Laboratory for Computer Science, Computer Systems Research Division, Local Network Note No. 3, November 1976.

[24] Roberts, L.G., Wessler, B.D., "Computer Network Development to Achieve Resource Sharing," Proc. AFIPS SJCC, 1970.

[25] Rockart, J.F., Bullen, C.V., Leventer, J.S., "Centralization vs. Decentralization of Information Systems: A Preliminary Model for Decision Making," M.I.T., Sloan School of Management, working paper (draft), 1977.

[26] Strazdas, R.J., "A Network Traffic Generator for DECNET," M.I.T., Department of Electrical Engineering and Computer Science, M.S. thesis (in preparation), 1978.

[27] Svobodova, L., Computer Performance Measurement and Evaluation Methods: Analysis and Applications, American Elsevier, 1976.

[28] Svobodova, L., "Comparative Study of the Ethernet and the Ringnet: Measurement Techniques and Tools," M.I.T. Laboratory for Computer Science, Computer Systems Research Division, Local Network Note No. 15, October 1977.

[29] Thomas, R.H., "A Resource Sharing Executive for the ARPANET," Proc. AFIPS NCC, 1973, pp. 155-163.

[30] Wecker, S., "The Design of DECNET - A General Purpose Network Base," Proc. IEEE ELECTRO/76, Boston, Massachusetts, May 1976.

[31] Wulf, W.A., London, R., Shaw, M., "An Introduction to the Construction and Verification of Alphard Programs," IEEE Trans. on Software Engineering, Vol. SE-2, No. 4, December 1976, pp. 253-263.