EDP ANALYZER

JULY, 1980
VOL. 18, NO. 7

QUANTITATIVE METHODS FOR CAPACITY PLANNING

The planning and management of computer capacity never has been exactly easy. For one thing, it has always been difficult to determine just how much capacity a given computer configuration really has, even if the workload is just 'plain old batch'. But the problem is getting even more complicated. There is a growing variety of workloads—batch, remote batch, transaction processing, time-sharing, database queries, and so on. And then some of the new computer architectures make it difficult to know when the computer is actually doing useful work and when it is going through wasteful motions. In this report, we discuss two quantitative methods for getting computer capacity planning under better control.

Liberty Mutual Insurance Company, with headquarters in Boston, Mass., is one of the largest combined casualty-life insurance companies in the U.S. Annual income exceeds $2.8 billion and the company has some 20,000 employees. Liberty Mutual is the largest writer of worker's compensation insurance in the U.S., with about 10% of the market.

This summer, Liberty Mutual is in the process of consolidating its two large data centers into a new one. One of the current centers is located at the company's headquarters, and the other is about 20 miles away. The new consolidated center will be in New Hampshire, some 60 miles north of Boston. The Boston center has been using two IBM 370/158-APs and the other center has been using a 370/168-MP.

The company has been planning for this consolidation move for about two years. And it was the Boston center that offered the most challenge. The main question was: how would the center be able to serve its customers during the period until the new center was ready, in the face of some rather severe constraints?

The problem has been that the Boston data center has reached its maximum of physical space. Very little capacity could be added during the past two years without a major equipment change—and this the company was unwilling to make. The daytime shift at the Boston center has been used mostly for time-sharing (TSO) and da-

ISSN 0012-7523. Multiple copy prices listed on last page.

tabase (IMS) applications. The second and third shifts have been used mostly for batch work. But the time-sharing applications have been growing at the rate of about 15% a year (in resources consumed). And two major database applications had been started; by the time the new center would be available, one of these would increase its use of computer resources by 400% and the other by 2,500%. And there just was not any more floor space for adding much in the way of equipment. Existing tape units could be replaced by higher speed ones, and some additional main memory could be added—but that was about all.

So, two years ago, the Boston data center management was faced with the following questions: (1) How much could the capacity of the computers there be increased, to handle the increased workload? (2) Would this increased capacity be able to handle the larger workload, without serious deterioration of service levels and response times? (3) Could we verify that the new center would have the capacity to handle the total workload for a year after the consolidation took place?

To help answer these questions, Liberty Mutual turned to two sources. One was the IBM SNAP/SHOT service, available at an IBM data center in North Carolina. The other was BGS Systems, Inc., of Waltham, Mass. and their BEST/1 product. SNAP/SHOT is a simulation-type program that operates in a batch mode, for investigating how jobs will consume computer resouces. BEST/1 is an interactive system that uses a queueing theory model of a computer system for studying capacity problems.

SNAP/SHOT is a proprietary service of IBM that is available at IBM's Raleigh Systems Center, Raleigh, North Carolina. It is only available to IBM customers by way of their IBM sales representatives, we understand. It is not a commercial product because of the problem of keeping it up-to-date, as all new hardware and software releases are made. It covers central processors and their various operating systems, paging devices, data communications networks including SNA, SDLC, CICS, etc.

Liberty Mutual had used BGS several years previously, to investigate how many more applications could be added to an installed DEC PDP-11 geographically-dispersed, eight-processor system. BGS used BEST/1 in this investigation. At the end of the study, they told Liberty Mutual not only how much capacity the PDP-11 system had and how much more capacity could be added, but also they pointed out where the file access times of the PDP-11 looked wrong. In fact, something *was* wrong, and Liberty Mutual picked up some more capacity when they corrected the DEC file access routines. So, with the data center consolidation in the offing, Liberty Mutual decided to call on BGS again, on a consulting basis.

Since both SNAP/SHOT and BEST/1 required essentially the same input data, not a lot of extra work would be involved in using the two services. And by using the two services, Liberty Mutual would get a good cross-check. The capacity questions were just too important to allow for any doubts on the accuracy of the results.

For the data about time-sharing usage, Liberty Mutual used IBM's Resource Measurement Facility (RMF) data and TSO accounting data. For the batch workloads, RMF data was used. And for the database applications, RMF, IMS log tapes, and data communications monitor data were used. Since volume data for the two new database applications was not available, it had to be estimated by consultation with user departments and with system analysts.

During a four-day visit to the IBM center at Raleigh, the Liberty Mutual people were able to make about 60 SNAP/SHOT runs. Because of the computer time involved in these runs, the relatively stable batch workload was not considered, just the time-sharing and database workloads.

After BGS had developed the BEST/1 models for the configurations and workload, Liberty Mutual and BGS people were able to make between 50 and 60 analysis runs in an afternoon. They studied the impact of the batch, time-sharing, and database applications, by shift. Further, the time-sharing workload was divided into short, medium, and long jobs, based on the amount of resources consumed. The computer resources that were to be made available for each type of workload were defined, also.

The main points about the results of the studies were as follows. The two services gave comparable results. Where there were differences, a little thought could always find the reason. (In one case, for instance, a peak load was defined as 33% above the average load for the IBM study and 50% above average for the BGS study.) The similarity of results gave Liberty Mutual a feeling of confidence in the figures.

Also, sensitivity analyses were performed, using BEST/1. The company found, for instance, that if record retrieval times were 25-30% greater than estimated, system performance would be greatly affected.

But, of course, the main questions concerned expanding the capacity of the 158s and the ability to handle the growing workload. The analyses showed that the 158s could be enhanced by converting to the MVS/SE operating system, adding two megabytes of memory each, and converting to the 6250 bits-per-inch tape units—and this was as much as could be done within the physical space constraints. With this added capacity, the data center could support all workload growth except for one of the new database applications; only about six-tenths of that workload could be handled. Liberty Mutual would have to schedule conversion to that system so as to stay within the the capacity constraints. Also, a moratorium was declared on more terminals for system development.

The study also confirmed that the proposed configuration for the new data center would handle the initial workload when the data center opened. In addition, it would handle all of the anticipated, additional workloads for the first year—much of which had been delayed because of the capacity constraints mentioned earlier.

The analyses pointed out some performance figures that would have to be tracked very carefully during the period before the consolidation took place, to make sure that nothing was getting worse than expected. The company decided to acquire BEST/1 from BGS, so that they could continue to make analyses. And, we gather, things have worked out just about as BEST/1 predicted.

American Express

American Express Company is the third largest diversified-financial company in the U.S., according to *Fortune* magazine. With its headquarters in New York City, the company employs almost 38,000 people and has assets of over $14 billion. It is best known, of course, for its credit card and travelers cheque services, but the company also provides money order, travel agency, and other services.

We visited the travelers cheque division in New York City, to learn about their new use of a capacity planning method—the software physics approach to capacity planning, developed by the Institute for Software Engineering (ISE), of Palo Alto, California. (We discussed the software physics method in our March 1978 issue.)

In the spring of 1979, data processing management in the division thought that their IBM 370/168-III was operating close to its capacity. For one thing, the growth in volume of travelers cheque transactions had continued its steady pace. Also, the system development staff was asking for more and more test time, as new systems were being developed and old ones enhanced. In addition, the data processing activities of the money order and travel di-

visions had just been transferred to the travelers cheque data center. All in all, the computer workload had just about doubled within the previous year.

Before ordering another computer, however, data processing management decided to use software physics to find out just how close to maximum capacity they were operating the 168. American Express had been a member of ISE for several years but had not yet tried to use software physics. This was a good chance to find out just how useful it was, management felt.

The first step (with some help from ISE) was to define the workload in terms of units of software work; we will discuss these units later in this report. Computing the units of software work for data transfers to and from disk and tape, for each application, was easy; it is equal to the number of characters of data transferred. But computing the units of software work performed by the CPU for each application took a little more effort.

To compute this CPU work, the travelers cheque division people used the CPU power figures provided by ISE for the 168-III. 'Power' is defined as the amount of work performed by the CPU per second—and we will have quite a bit more to say about this later. Multiplying the power figure by the amount of CPU execution time for each application gave the CPU work content of that application.

Another important measure is calculated from the power figures, and that is the 'practical maximum' power of the configuration. While the full theoretical power of the configuration is of interest, it is almost impossible to actually use that amount of power; service levels and response times become intolerable as it is approached. Instead, a lower 'practical' power level should be considered as a target—and ISE defines this level as that power which is sufficient to insure 95% of the jobs being completed within the specified service levels and response times.

To get the raw data from which the software work was computed, the division people used measurements obtained with IBM's RMF, SMF and IMS log tape data. These are common sources of performance measurement data for users of large IBM mainframes.

The next step was to plot the workload against the practical maximum capacity, by hour of the day, for selected average and peak days in the six months ahead. The travelers cheque business is quite seasonal, with 60% of the volume occurring in the five summer months.

What these charts showed was what data processing management suspected. During peak hours, the workload exceeded the practical maximum—meaning that service levels and response times were affected. Also, there was no obvious waste of the power; no simple tuning of the configuration would provide the needed extra capacity. Further, the condition was going to get worse with each passing month.

The upshot was that management was convinced that more capacity was, in fact, needed. The option that was chosen was to order a second computer, an IBM 3032. Initially, the production work was kept on the 168, and all development work was transferred to the 3032. Eventually, as production work outgrows the 168, some of it will be transferred to the 3032.

This use of software physics turned out to be relatively easy, we were told. So much so that division data processing management would like to make its use much more routine. And perhaps, as American Express' other data centers learn about this experience, they might decide to use this approach to capacity planning.

Capacity management: Who needs it?

Who needs to practice capacity management? The short answer is, of course: essentially all computer users. But a more realistic answer is: some users have a more

pressing need for capacity management than do others.

The less-pressing problem case. We have talked with a number of users of large computers who do not seem to have pressing problems of capacity management. The characteristics of these organizations which might explain the situation are the following. The bulk of their workloads is both very important to the organizations (central to their purpose, in fact) and the volume is very hard to predict. So capacity management is mostly a case of obtaining enough computer capacity to provide a satisfactory level of service for the urgent batch work, and a satisfactory response time for the important on-line work.

In the situations of this type that we have observed, the urgent batch work and important on-line work are generally performed during the first shift. Routine production work is handled on second and third shifts.

Also, within quite broad limits, it is much more of a 'sin' for data center management to have too little capacity as opposed to too much. We have encountered some cases where the computer budget has reached such a magnitude that top management begins asking, "Do you *really* need that new capacity you are requesting?" *Then* capacity management begins to get important.

In general, though, we have observed that data center management just extrapolates the past usage for critical work, in order to determine how much capacity will be needed in the future. The capacity needed for handling the critical work during the first shift then turns out to be more than enough to handle the routine work during second and third shifts.

The problem case. The more common situation, however, as far as we can determine, is where the routine production work is the major part of the total workload. It is often overlaid with reasonably critical work, such as time-sharing use, database queries, transaction processing, and so on. Data center management must provide enough capacity to give acceptable service for this important work. But at the same time, top management imposes a fairly tight budget on total computing costs. So a real effort is made to obtain sufficient—*but not too much*—capacity.

This is the situation where capacity management becomes more of a challenge. And this is where the use of quantitative methods can help.

Why capacity management? One might ask: Why the concern about not over-buying capacity? How can top management know if there is more than enough computing capacity or not? Don't they have to take data center management's word on this?

That used to be the situation. But management audits are becoming more commonplace. And a sharp computer professional, performing such an audit, can point out that money is being wasted in excessive computer capacity. Top management need no longer take data center management's word on how much computer capacity is needed.

In fact, we gather that top management in many organizations, for a number of years, has been suspicious of the demands for bigger computers. They have been concerned that some of these demands might really amount to 'empire building'. E. L. Pritchard, in Reference 1, shows a *New Yorker* magazine cartoon that expresses top management's concern. In the cartoon, a king is talking to a meeting of his ministers, and says, "Gentlemen, the fact that all my horses and all my men couldn't put Humpty together again simply proves to me that I must have *more* horses and *more* men."

But in most organizations, we believe, the intent of data center management is to do as good a job as possible and to provide users with acceptable levels of service, not build empires. They would prefer to err on the side of somewhat too much capacity,

because they know that too little capacity will soon turn into very loud customer complaints. And it can be very difficult to determine just how much capacity will actually be needed, because of hard-to-predict elements of the workload. These elements include possible mergers, consolidations, new uses of the computer not foreseen a year or so ago, and so on. Further, they know that capacity management is not a one-time affair; they will continue to face the problem, year-in and year-out. They want an orderly path to capacity growth over the years.

So who needs computer capacity management? All computer users—but particularly those whose budget constraints are forcing them to provide sufficient capacity to insure good customer service but without wasting money on much unused capacity. It is such organizations that can benefit from the use of quantitative methods for capacity planning, we believe.

What is capacity management?

The Institute for Software Engineering defines capacity planning (as quoted in Reference 2) as: "...that set of functions concerned with determining and maintaining the proper balance between the workload and the equipment configuration at a minimum cost consistent with throughput, response time, and reliability objectives." P. C. Howard, in Reference 2, goes on to say that "The basic elements comprising the field (of capacity planning) include: instrumentation and measurement, analysis and reporting, performance evaluation and control, planning and forecasting, *EDP* cost accounting, and management review and control."

The terms 'capacity management' and 'capacity planning' seem to be used almost interchangeably. We see capacity management as perhaps broader than the above definitions imply. We think that capacity management must also consider: (1) the organization's total computing environment, such as multiple data centers (perhaps including some in other countries), the advent of distributed systems, and the way that the computing resources can best be managed in such an environment, (2) the long range plans of the organization, for offering new products in the market place, or for entering new markets, or for possible mergers or acquisitions, (3) the long range plans for data processing, and (4) the most likely schedule and impact of new computer technology, such as new hardware announcements.

We talked with the people at Transamerica Information Services, in Los Angeles. They use, and have used, software physics—in part, for capacity planning. But the point they emphasized to us was that such quantitative methods can be helpful *only for a portion* of the total capacity management problem. That portion is important and the methods can be quite helpful. But, beyond that, there are other aspects of computer capacity management that *must* be considered.

In this report, then, we will concentrate on that part of capacity management where quantitative methods can help. The two methods we will discuss are: *operational analysis* and *software physics.*

Operational analysis

G. S. Graham, in Reference 3, provides a good summary of the historical background of operational analysis. In order to be able to predict the behavior of complex systems, models of those systems are needed. Two major types of models are the *analytic* and the *simulation* models. We are concerned here with analytic models.

Some pioneering work on both of these types of models was done in the 1950s and early 1960s at the Management Sciences Research Project, at the University of California, Los Angeles, under the sponsorship of the Office of Naval Research. Most of this research was aimed at finding models for production scheduling in complex job shop environments. Some of the key ana-

lytic model work was done by James Jackson, using queueing theory models.

Jackson's results were used to predict the behavior of multi-programmed time-sharing systems in the early 1970s. The results proved to predict performance quite accurately. And this accuracy suprised some knowledgeable people. As Denning and Buzen describe (in Reference 3), the traditional queueing theory models are based on a series of assumptions that cannot be proved. For example, these assumptions say that the system under study behaves as a stationary stochastic process, that the job steps follow a Markov chain, and that job service times are exponentially distributed. The authors point out that there is no way, given a set of measurements on a system, to prove that these assumptions are, in fact, true. Moreover, there are many reasons for believing that they are *not* true.

In short, there are a number of complex concepts that the analyst should understand in order to properly use traditional queueing theory. Further, since the basic assumptions cannot be proved, how can the analyst be sure that the results will be valid? Just because past results have been accurate is no guarantee that future results will be.

Buzen began work in the early 1970s on *operational analysis*—that is, a method of analyzing queueing network models in such a way that all hypotheses are operationally testable. He was soon joined in this work by Denning, and since then the two have collaberated in developing the method. We found their paper to be very readable, with lots of examples to illustrate the principles. Moreover, the method seems very 'common sense-ical'. We will give a brief overview.

Determine parameters. First, the analyst determines the *workload parameters*—measurable quantities that characterize the workload. These include the number of jobs of each type that will use the resources in the system. Also needed are the average number of requests for service for each device, for each job type. Some of these parameters may be readily counted, or found from source code. Or some may be determined by hardware or software monitors.

Then the analyst must determine *device parameters*. For instance, the key performance characteristics of disk drives are seek and rotation times, plus data transfer rates. And some of the parameters are combinations of workload and device parameters—such as average disk service time, which is based on the disk drive parameters plus the average amount of data transferred per request, which is job related.

Measure variables. The next step is to measure some specified variables that characterize the workload. One is the length of the observation period. Another is the number of job arrivals during the period. Still another is the number of job completions during the period. Another variable is the total time the system is busy. These quantities—arrivals, completions, total time, and total busy time—are measured for each device in the system. For on-line systems, other variables include the number of terminals attached, the average operator 'think time' at the terminal before making the next entry, and the average response time.

It is evident that these variables are easy-to-measure ones. There is no great mystery about them.

Derive other variables. With this data at hand, some other variables are calculated. One is the arrival rate—which equals the number of job arrivals divided by the number of seconds in the observation period. A second variable is output rate, which is the number of completions divided by the number of seconds. Next is utilization, which is busy time divided by total time, and then average service time per job (busy time divided by number of completions). And the 'visit ratio' must be determined, which is the average number of

times that each type of job requests service from a device, such as disk storage.

These variables, too, seem quite straight-forward.

Invariance assumption. We should single out one assumption that applies to the *use* of opertional analysis. It need not always be true; where it is not true, the study results would be affected.

This assumption is the *invariance* of parameters. The user of operational analysis typically assumes that the workload and device parameters will not change under different levels of workload. This assumption of invariance is not always true, say the authors. For instance, increasing the number of on-line terminals may unexpectedly reduce the amount of multi-programming available for batch work. So the analyst should determine which parameters might change and explicitly state any such changes.

The analysis. At this point, the analysis can begin. The authors have derived a number of operational analysis equations—and many of them do not seem too complicated.

In their examples, the authors compute a number of expected performance results for systems. Many of the examples include a mixture of both batch and interactive workloads that use (mostly) the same resources. The authors calculate such things as overall completion rates, interactive response times, identifying the bottleneck devices and the volumes at which they will saturate, and the effect on the completion rate and response time of substituting a faster CPU for a slower one.

The examples illustrate the *principles* of operational analysis. In the practical application of the method, computer modelling would be desirable. One of the authors (Buzen) has developed BEST/1, which we will discuss shortly.

To repeat: the big advantage of operational analysis over analysis by traditional queueing network theory is that, unlike the latter, the analyst can conceive simple experiments to measure and verify all of the hypotheses upon which the model depends. With operational analysis, the predicted results agree to within 10% of actual, where verification can be performed, say the authors. So the effect of deviation from the method's assumptions is not too serious, they say.

The advantage of operational analysis over *simulation* models is—the former are *much* faster to process. With simulation models, it usually takes a lot of computer time before the analyst can get a fairly good idea of how the system under study performs.

In a separate paper in Reference 3, Buzen describes the application of operational analysis to the study of a computer system that uses IBM's Multiple Virtual Storage (MVS) operating system. With this system, the memory is divided into domains, and each workload type is assigned to a domain. Management may want, say, 30% of available first shift resources to be used for batch processing, and 50% for time-sharing, and the final 20% for on-line transaction processing. Further, the time-sharing work can be divided into three classes—short, medium, and long, each with its own domain. Jobs can then shift from the higher priority domain into the next lower priority domain as service time accumulates.

These papers illustrate in quite readable language, we think, how operational analysis can be used for studying the behavior of computer systems with many levels of multi-programming.

BEST/1

BEST/1 is a proprietary software system developed by BGS Systems, Inc., of Waltham, Massachusetts. It is based upon the principles of operational analysis and is used for evaluating and predicting computer system performance. For more information on it, see Reference 5.

In use, the analyst first defines the existing workload, as we discussed above. This includes daily normal and peak loads. Estimates of future growth in volume are obtained, for both normal and peak loads. And the analyst must check the flow of work carefully, to make sure that there are no 'hidden' characteristics that are overlooked.

The next step is to define the hardware and software parameters. For this step, standard measuring tools are normally used, such as accounting packages and hardware and software monitors. The CPU time is broken into several components—the CPU time per database access, the time required to assemble logical records from physical database segments (where that is relevant), and so on.

Next, a network queueing model is defined in BEST/1, to represent the system being studied.

The data is then input to the BEST/1 model, which determines such things as output completion rates, response times, bottleneck devices, and such. Each device has its own queue; a job destined for that device 'waits' in the queue until its turn comes. But this 'waiting' is not in the same sense as a simulation model; rather, it is calculated by equations, as a function of the other work that the system is working on. Total service time for a job is the sum of the service times and the wait times.

The model, the variables, and the parameters are first tested by estimating the service times for the current workload and then comparing these figures to actual times. Once the model has been thus tested and found accurate, it is used to evaluate performance under changed conditions—greater volume, changes in hardware, new applications added, and so on. Further, sensitivity analyses generally are made. Parameters and variables can be changed and new performance figures computed, to see when performance begins to deteriorate.

And, as we indicated earlier in this report, it is quite feasible to make 50 or more such performance 'runs' in the course of a one-day use of BEST/1.

Let us now consider the use of software physics for capacity planning.

Software physics

The following concepts of software physics were developed by Kenneth Kolence, President of the Institute for Software Engineering, in Palo Alto, California. Many of the points will be found in Kolence's papers in References 1 and 4.

Workload represents the *demand* for computing capacity, while the computing resources provide the *supply* of capacity. What is needed, says Kolence, is a common unit of measure that can apply to both the demand and the supply.

In the past, the unit of measure has usually been the *job*. But this is a very ambiguous measure, he says. Jobs vary greatly in size. At one extreme is a very long batch job; at the other is the processing of a single transaction in an on-line transaction processing system. The total workload can be divided into general classes in order to avoid such great disparity, but even within a class, jobs can and do vary in size.

In fact, says Kolence, the unit of measure should not only avoid the shortcomings of the 'jobs' unit, it should also be independent of the exact characteristics of the equipment on which the workload is processed. The amount of work in a given workload should be invarient, regardless of the computer on which it is done.

The unit of work chosen by Kolence is *software work units*. In brief, a unit of software work is done when one byte (of data or program) is transferred—between the CPU and main memory, between the CPU and disk storage, between tape and the CPU, etc.

Means are needed for computing the total work in a given workload. The workload can be divided into parts, for easier later analysis: by *type* of work

(batch, time-sharing, test, database queries, etc.), by *time of day* (first, second, and third shifts—or by hours, if needbe), or by a combination of *both.* So, within any such division, the total work must be determined.

Determining the amount of work done in transferring data and programs between the various storage devices and the CPU may be easy. It is equal to the number of such transfers multiplied by the average number of bytes in each transfer. Such data can be determined from the use of standard accounting and performance monitoring packages.

Determining the amount of work done by the CPU on the workload is not as easy. For this, one needs to know the 'power rating' of the CPU being used. Initially, power ratings must be determined by the use of hardware and software performance measuring devices. But once the power rating of a given CPU has been determined, it can be used by the users of that CPU. (We will mention below some of the complications of these power measurements with some of today's CPUs.)

Given the power rating (amount of work the CPU can do per second), the total amount of CPU work is computed by multiplying the power rating times the CPU execution time (in seconds) to perform the workload.

Total work for an application is then computed. It is the sum of the work performed on that application by the various devices—CPU, disks, tapes, printer, and other devices.

Then, says Kolence, you can do two things that will help the later analysis. One is to divide the total work within an application by the number of natural units in the workload—that is, the number of transactions processed, or the number of paychecks printed, etc. These are numbers that users quickly relate to. He calls these 'natural forecast units' because users usually can estimate how many transactions or paychecks the system will have to handle at different points in the future.

The other thing to do, which is closely related to the natural forecast units, is to determine the overall average work per 'job', where the job might be, say, all of the payroll runs. In fact, a 'standard' job can be determined for each type of workload, for comparison among types of workload and between different installations. But note: in this case, the standard jobs have been determined in terms of units of software work; they are not the ambiguous jobs so often used in capacity planning.

The supply side. The goal of capacity planning is to match the capacity (the available power) against the workload. It must be recognized, says Kolence, that the power that is used is generally less than the available power of a configuration.

Further, most of the discussion of power is confined to the CPU, the disks, and the tapes. The total configuration power, of course, must include the printers, card readers, data communications, etc. But the power added by these other units is small compared with that of the main three—CPU, disks, and tapes.

The *relative* power of the configuration is determined by the total amount of work done divided by the *elapsed* time. Since elapsed time is greater than the execution time of the units, the relative power is only an approximation of what is wanted. The *absolute* power is determined by the amount of work done divided by the actual execution time of the CPU (for CPU power), and by the execution time for *any* disk (for disk power) or *any* tape (for tape power).

But, as Kolence discusses in his paper in Reference 1, for some CPUs the power computations are not straight-forward. For instance, in IBM 370 models 158 and above, the processors have been designed to reduce average instruction execution time by fetching multiple instructions and operands at one time and storing them in

fast memory. If those extra instructions and operands are actually used, fine; work has been done by bringing them into fast memory. But if the program branches so that they are not used, then no real work has been done by bringing in the unused ones.

The CPU execution time can also be affected by the design of the software—the instruction mix used and the sequence in which the instructions are used. The software may have been designed in a way that stretches the overall execution times, thus degrading performance.

So determining the CPU power for larger 370 models, for instance, has required a good amount of study—by staff members at Kolence's Institute for Software Engineering and by some corporate members of the Institute. The Institute has developed a 'computer power calibration instrument', a self-contained measurement and reporting unit for making power measurements more precisely than is generally done with hardware monitors. Also, instead of a single instruction mix, three basic mixes are used (representing decimal arithmetic, engineering calculations, and long moves and compares).

The result is (a) a *set* of power ratings for each CPU, based on the different instruction mixes, and (b) a representive single power value for each CPU. The set of values is useful for system and program design, for acceptance testing of field upgrades, and for benchmark construction and analysis. The single value is useful for comparing configurations and for determining CPU work.

We have concentrated on the CPU part of power determination. But disk and tape power also can involve a few complications. For instance, with multiple IBM 3330 disk drives operating in the non-RPS mode, there can be multiple seeks and some multiple searches occurring simultaneously. Because of this overlap, more work is performed during the execution time than if just one drive were busy. Somewhat like the case of the CPU, the power can be affected by the job mix.

An interesting point brought out by Kolence is that adding disk drives to a disk control unit does little to increase the *power*; it is mostly the storage capacity that is increased. For instance, with eight drives connected to one control unit, the power is only 2.1 times the power of a single drive. So control units and channels can act as 'power bottlenecks'.

Using software physics. As the discussion implies, with software physics one determines the workload, by type, in terms of natural forecast units and/or in terms of standard jobs. Then one determines the power of the configuration to perform work, taking into account some of the subtle points just discussed. And then one plots available power and workload, for the hours of the day and the days of the month being considered.

The workload, expressed in units of software work, is considered to be independent of the configuration on which it is processed. The subtle impacts of the software design on execution times are taken into account in the power ratings of the devices. So it is quite feasible, says Kolence, to test out the same workload on a variety of configurations.

For more information on the Institute for Software Engineering and on software physics, see Reference 6.

Comparing the methods

We will try to give a few comparisons of these two methods, based on some points made by others plus our own thoughts from studying the methods and from talking to users. But we will start off by describing some aspects that apply to both methods.

For one thing, all quantitative methods for capacity planning require the collection of data about the workload and about the speed at which the devices perform work. Most users so far, we gather, rely almost completely on data gathered by ac-

counting and performance monitoring software packages. Examples of such packages include IBM's SMF, RMF and IMS log tape analyzer. And mainly, it has been the accounting package (such as SMF) that has been used. These packages, while helpful, fall short of providing the type of data really needed for good quantitative capacity planning.

Secondly, both of these methods really need trained people for their effective use. These people should have a background in capacity planning, and should receive additional training in the method(s) to be used. The two methods discussed in this issue have been developed in a way that does not require the users (the analysts) to know advanced mathematics or queueing theory. For using these methods, one to three people, say, might be selected from the staff of people experienced in systems programming, the operating system, and data communications. Then these people must be trained in the new method(s), and should communicate with other users of the method(s). One must expect, too, that some of these people will not work out and will have to be replaced.

We hope that our discussion has given some idea of the staff capabilities that are needed for this work. If you are interested, however, you should check into these requirements in much more depth.

Thirdly, potential users seem not quite sure of their ability to handle rather sophisticated methods such as these, we gather. One of the main concerns in the minds of potential users seems to be: do we have the wherewithal to collect the necessary data and prepare the input for these methods? The tools for collecting the data (such as SMF, RMF, etc.) can seem imposing in themselves.

Along this same line, we heard a talk on BEST/1 given at a SHARE conference, and the questions and comments gave us the impression that the attendees were not sure of their ability to use the method. Further, P. C. Howard, in Reference 2, reports on the Computer Capacity Management Conference (of which Reference 1 is the proceedings). He says that about 50% of the attendees were members of the Institute for Software Engineering (which sponsored the conference) but only a few organizations indicated they were actually using software physics. Many of the attendees were really there, he said, to evaluate software physics as a possible methodology for their shops.

Comparing the methods. A main point to make, we think, is that these two methods—operational analysis and software physics—are much more complementary than they are competitive. In fact, we understand that Kolence and Buzen have discussed the idea of expressing the operational analysis variables, parameters, and equations in software work terms, rather than in the units now used (such as 'jobs').

As an example of how the two methods might help each other, Kolence, in his writings, has pointed out some aspects of operational analysis that he thinks are improved by this use of software physics. One of these is what we just mentioned—replacing 'jobs' with 'software work' as the unit of measure of work. Software work units make the analyses more consistent between installations and over time, he feels.

Another aspect of operational analysis that Kolence thinks is improved by using software physics is in separating the properties of the workload from the properties of the resources. For instance, operational analysis uses the concept of 'average service time per job' for the various devices—disk, tape, etc. But this concept includes device parameters (such as disk seek, search, and rotation times) and job parameters (such as average number of characters transferred). It will be easier to make different analyses, sometimes varying the workload and sometimes varying the resources, if the two are kept separate.

Another main point is that operational analysis is dynamic in nature, and consid-

ers the flow of work through a queueing network of devices. Software physics looks at the power of the devices to perform work; any dynamics are introduced by defining the devices and the workload that exist (or will exist) at any specified point in time.

The people at ISE tell us that they *have* re-described operational analysis into software physics terms, as described in Reference 4b. They feel that the use of operational analysis has been strengthened thereby.

The people at BGS agree with this only in part. While they see software physics as possibly strengthening some aspects of operational analysis, they feel that the method they now have is sufficiently useful.

For all practical purposes at the present time, a user organization probably would want to select one or the other of these methods. And until such time as the two might be combined, it seems to us that either of the two methods can offer substantial benefits for capacity planning.

REFERENCES

1. *International Conference on Computer Capacity Management, 1979; Conference Record,* available from the Institute for Software Engineering (address below).
2. *EDP Performance Review,* June 1979; published by Applied Computer Research (3003 West Northern Avenue, Phoenix, Arizona 85021). This issue is a report on the above conference. Many issues of this well-written publication deal with capacity planning; for instance, the September 1979 issue discusses capacity planning at American Airlines. Subscription price is $48 per year; single copy, $5.
3. *ACM Computing Surveys,* September 1978, Special issue on Queueing Network Models; published by ACM (1133 Avenue of the Americas, New York, N.Y. 10036); price $8 prepaid.
4. Papers available from the Institute for Software Engineering (address below):
   a) Kolence, K. W., "The capacity management and software physics primer"
   b) Traister, L. M., "Queueing network analysis in software physics"
5. For more information on BEST/1, contact BGS Systems, Inc., P.O. Box 128, Lincoln, Massachusetts 01773.
6. For more information about the Institute and about software physics, contact the Institute for Software Engineering, P. O. Box 637, Palo Alto, Calif. 94302.

SUBJECTS COVERED BY EDP ANALYZER IN PRIOR YEARS

1977 (Volume 15)

Number

1. The Arrival of Common Systems
2. Word Processing: Part 1
3. Word Processing: Part 2
4. Computer Message Systems
5. Computer Services for Small Sites
6. The Importance of EDP Audit and Control
7. Getting the Requirements Right
8. Managing Staff Retention and Turnover
9. Making Use of Remote Computing Services
10. The Impact of Corporate EFT
11. Using Some New Programming Techniques
12. Progress in Project Management

1978 (Volume 16)

Number

1. Installing a Data Dictionary
2. Progress in Software Engineering: Part 1
3. Progress in Software Engineering: Part 2
4. The Debate on Trans-border Data Flows
5. Planning for DBMS Conversions
6. "Personal" Computers in Business
7. Planning to Use Public Packet Networks
8. The Challenges of Distributed Systems
9. The Automated Office: Part 1
10. The Automated Office: Part 2
11. Get Ready for Major Changes
12. Data Encryption: Is It for You?

1979 (Volume 17)

Number

1. The Analysis of User Needs
2. The Production of Better Software
3. Program Design Techniques
4. How to Prepare for the Coming Changes
5. Computer Support for Managers
6. What Information Do Managers Need?
7. The Security of Managers' Information
8. Tools for Building an EIS
9. How to Use Advanced Technology
10. Programming Work-Stations
11. Stand-alone Programming Work-Stations
12. Progress Toward System Integrity

1980 (Volume 18)

Number

1. Managing the Computer Workload
2. How Companies are Preparing for Change
3. Introducing Advanced Technology
4. Risk Assessment for Distributed Systems
5. An Update on Corporate EFT
6. In Your Future: Local Computer Networks
7. Quantitative Methods for Capacity Planning

(List of subjects prior to 1977 sent upon request)

PRICE SCHEDULE (all prices in U.S. dollars)

	U.S., Canada, Mexico (surface delivery)	Other countries (via air mail)
Subscriptions (see notes 1,2,4,5)		
1 year	$48	$60
2 years	88	112
3 years	120	156
Back issues (see notes 1,2,3,5,)		
First copy	$6	$7
Additional copies	5	6
Binders, each (see notes 2,5,6)	$6.25	$9.75
(in California	6.63, including tax)	

NOTES

1. Reduced prices are in effect for multiple copy subscriptions and for larger quantities of a back issue. Write for details.
2. Subscription agency orders are limited to single copy subscriptions for one-, two-, and three-years only.
3. Because of the continuing demand for back issues, all previous reports are available. All back issues, at above prices, are sent air mail.
4. Optional air mail delivery is available for Canada and Mexico.
5. We strongly recommend AIR MAIL delivery to "other countries" of the world, and have included the added cost in these prices.
6. The attractive binders, for holding 12 issues of EDP ANALYZER, require no punching or special equipment.

Send your order and check to:
EDP ANALYZER
Subscription Office
925 Anza Avenue
Vista, California 92083
Phone: (714) 724-3233

Send editorial correspondence to:
EDP ANALYZER
Editorial Office
925 Anza Avenue
Vista, California 92083
Phone: (714) 724-5900

Name_____

Company _____

Address _____

City, State, ZIP Code_____