Chapter 13

Evolving Systems

13.1. Introduction

We create and validate queueing network models of baseline systems, as described in Chapter 12, so that these models can be used to project the effects on performance of contemplated modifications to the workload, to the hardware, and to the operating policies and system software. In this chapter we will see how to represent such modifications by alterations to the inputs of the validated model. The accuracy and utility of the resulting performance projections depend on three factors:

- *how well the baseline model validates* — The construction and validation of baseline models was discussed in Chapter 12.
- *how accurately the modifications are forecast* — Anticipating the evolution of a system and its workload is a difficult task that is faced by organizational management. It lies beyond the scope of this book.
- *how well the anticipated modifications are represented as changes to the model inputs* — This is the subject of the present chapter.

In general, system modifications have both *primary* and *secondary* effects. For example, a CPU upgrade has the primary effect of reducing the CPU service requirement of each user (in seconds, rather than instructions), and may have one or more secondary effects, such as changing the number of times that each user is swapped, on the average. We will see that for many modifications it is relatively easy to anticipate and represent the primary effects, but harder to anticipate, and thus to quantify and represent, the secondary effects. For this reason, successful performance projection studies in which several alternatives are being considered often take the following form:

- Initially, each alternative is investigated by representing only its primary effects. This can be done quickly.
- The results may reveal that some of the alternatives are not worthy of further consideration. These alternatives are discarded.

— The remaining alternatives are investigated in more detail, with attention paid to secondary as well as primary effects.

The organization of this chapter reflects this scenario. In Sections 13.2, 13.3, and 13.4, we discuss modelling the effects of modifications to the workload, to the hardware, and to the operating policies and system software, respectively. We concentrate in these sections on representing the primary effects of modifications, but also discuss certain secondary effects that are peculiar to a particular type of modification.

In practice, two or more modifications often will occur together. For example, if an increase in transaction processing volume is anticipated (a modification to the workload), one may wish to project performance under the assumption that the CPU is upgraded (a modification to the hardware). For clarity of presentation we will discuss such changes separately. To represent the effect of multiple modifications, the corresponding model input alterations can be applied serially.

In Section 13.5 we discuss some secondary effects that are common to most types of modifications. An example is the change in the level of variable overhead (CPU and I/O overhead due to swapping, for instance) that may accompany various modifications.

Finally, in Section 13.6, we describe three related case studies in which queueing network models were used to project the effects on performance of various modifications. In each case, the accuracy of the projection was assessed after actually implementing the modification. These three case studies are similar in spirit to the two studies of an IBM computing complex that we discussed in Section 2.2, where the modelling cycle was presented. A review of Section 2.2 would be worthwhile at this point.

13.2. Changes to the Workload

The workload presented to a computer system can change in several ways. First, the intensities of workload components can change. Second, the character of workload components (e.g., the service demands) can change. Third, the number of workload components can change. The following three subsections describe how the effects of each of these changes can be represented by adjustments to the inputs of a validated model. Both in this section and in the ones to follow, we will indicate modified input parameter values as primed quantities. For example, $D'_{c,k}$ will denote the modified service demand of class c at center k.

13.2.1. Changes in Workload Intensities

The most frequently studied workload changes are changes in intensity. Naturally, the primary effect of such a change is reflected by modifying the appropriate workload intensity input parameters.

For a transaction class, a typical workload forecast would be "a 30% increase in transaction volume". This can be represented in the model by $\lambda'_c \leftarrow 1.3\ \lambda_c$.

For a terminal class, a typical workload forecast would be "a 50% increase in the number of active users". This can be represented in the model by $N'_c \leftarrow 1.5\ N_c$. (In the absence of evidence to the contrary, it is reasonable to assume that average think time does not change.)

In the case of both transaction and terminal classes, increased competition for main memory will result from an increase in workload intensity. If the baseline model included a memory constraint ("at most twenty requests simultaneously active"), then we may assume that the same constraint still applies. If no such constraint were present in the baseline model, then the analyst must decide whether or not the increased central subsystem population that results from the parameter modification is realistic in light of the amount of memory available. If not, an appropriate memory constraint should be imposed. In either case, the variable component of overhead (e.g., paging and swapping service demands) may increase. This is discussed in Section 13.5.

For a batch class, it is unusual for a workload forecast to be phrased in terms of the multiprogramming level, N_c. (More likely, such phrasing would be used to describe the addition or re-allocation of memory.) Additional complexity arises from the fact that the value of this parameter in the baseline model can be due to several factors. At one extreme, there may be a persistent backlog of batch jobs, so that N_c reflects a memory constraint. In this case, an increase in the availability of batch jobs would only result in a larger backlog. At the other extreme, if sufficient memory is available to activate most batch jobs immediately when they arrive, the value of N_c is not related to a memory constraint. In this case, an increase in the availability of batch jobs would allow N_c to increase. Typically, a workload forecast for a batch class will be phrased in terms of throughput. The analyst must adjust N_c to achieve the forecast throughput, and then consider whether or not the increased central subsystem population is realistic with respect to the available memory.

13.2.2. Changes in the Character of Workload Components

Changes to application programs may lead to changes in the resource requirements of customers. Such changes would be represented in a model by adjusting service demands. Three examples are given in the following paragraphs.

It is proposed to modify an application program to do more checking of the validity and consistency of the input data it receives. The change is projected to increase the CPU path length of a transaction by 20%. The primary effect of this modification can be represented in the model by increasing the CPU service demand of transactions by 20%.

It is proposed to introduce data compression techniques to reduce the space occupied by a file that is processed sequentially by an application. The data transferred by the application will decrease, while its CPU requirements will increase (to translate data from compressed to uncompressed format and back again). To represent this modification in the model, the data transfer component of the service demand at the appropriate disk should be decreased, while the service demand at the CPU should be increased.

It is proposed to change the structure of a file used by an application. Initially, the file had three levels of indexing with the highest level kept in memory. The number of I/Os required to access any record was three: two index blocks plus the record itself. The new organization will be based on hashing, which is expected to decrease the average number of I/Os per record access to roughly 1.5. The primary effect of this modification can be represented in the model by halving the visit count at the appropriate disk (assuming that this is the only use of the disk by the class). A secondary effect of this modification might be an increase in the seek component of the service requirement at the disk, because the hashing technique would eliminate any locality of reference that might have existed under the indexed organization.

13.2.3. Changes in the Number of Workload Components

The primary effect of removing a workload component from a system is represented easily in the model by eliminating the corresponding customer class. The result will be a decrease in the activity at various devices, and a corresponding improvement in the performance of the remaining workload components.

Similarly, the primary effect of adding a workload component is represented by adding a new class. The result will be an increase in the activity at various devices, and a potential degradation in the performance of the original workload components. Of course, the workload intensity and service demands of the new class must be determined and specified. If a similar application runs at some installation with a similar hardware and software configuration, then measured service demands can be used. For a new application that cannot be measured, estimating service demands is much harder. This problem will be treated in Chapter 14.

Both the removal and the addition of workload components have a number of effects which, although of lesser importance than changes in device congestion, still can have considerable impact on performance. When a workload component is removed, memory becomes available for allocation to the remaining components. Knowledge of the operating policies of the system is required to determine how to represent this. When a component is added, it may be necessary to obtain memory at the expense of other components. Again, system knowledge is required.

As always, secondary effects arise in the realm of variable overhead. These will be considered in Section 13.5.

Modelling changes in the number of workload components is of particular benefit in multiple mainframe installations composed of several machines of the same architecture using the same operating system. In such environments, a large part of capacity planning involves projecting the performance resulting from various ways of assigning workload components to machines. The service demands measured for a class on one system can be translated for other systems, using known speed ratios. An example of capacity planning in a multiple mainframe environment was considered in Section 2.2.

13.3. Changes to the Hardware

New hardware products based on recent technological developments are announced with great frequency. This makes capacity planning and configuration management a continuing challenge. Fortunately, queueing network models are well suited to quickly evaluating configuration modifications.

In the subsections to follow we describe how CPU upgrades, memory expansions, and I/O subsystem modifications can be represented as modifications to model input parameter values.

13.3.1. CPU Upgrades

Perhaps the most common configuration change is the upgrade of a CPU within a family of processors of the same architecture. Fortunately, this also is one of the easiest changes to evaluate using queueing network models. The relative instruction execution rates among processors within a family generally are known and publicized by vendors and user groups. Consequently, the primary parameter change is to multiply the CPU service demand by the ratio of old CPU's processing rate (r_{OLD}) to that of the new (r_{NEW}):

$$D'_{c,CPU} \leftarrow \frac{r_{OLD}}{r_{NEW}} \times D_{c,CPU} \qquad \text{for each class } c$$

A common secondary effect of a CPU upgrade is a change in variable overhead (considered in Section 13.5). Additional memory or I/O equipment often accompanies such an upgrade (later subsections suggest ways to reflect these changes).

Rather than acquiring a faster CPU, it sometimes is possible to acquire a second processor to form a tightly coupled multiprocessor system. As we discussed in Chapter 11, the primary corresponding change to model parameters would be to represent the processor complex as an FESC with service rate approximately twice as great with two or more customers present as with only one customer present. An important secondary effect is the interference between the processors in accessing memory or shared data structures. This interference causes the capacity of a dual processor to be considerably less than twice the capacity of a single processor. If appropriate measurement data is available, the service rates of the FESC can be set to reflect the degree of interference. An example in Section 13.6 treats the change from a uniprocessor to a dual processor.

13.3.2. Memory Expansions

Since additional memory can be allocated in a number of ways, representing the effect of a memory expansion requires knowledge of the operating policies of the system.

The most common way to employ additional memory is to permit an increase in the central subsystem population of various classes. For batch classes, the parameter N_c would be changed. For transaction or terminal classes, the memory constraint would be adjusted upwards. The key, of course, is to estimate the extent to which each class will be affected. To some extent, this is under the control of installation-dependent tuning parameters. A few, well chosen experiments with smaller memory sizes can help to determine the effect of operating policies. Changes in swapping and paging activities can result; these secondary effects are discussed in Section 13.5.

Additional memory also can be used to permit workload components to run more efficiently at existing central subsystem populations. In this case, the entire effect of the memory upgrade would be felt as a decrease in variable overhead (see Section 13.5).

A third use of additional memory is to make frequently accessed files permanently resident in memory. Examples include system routines or indices. If measurement data indicates frequency of use for these files, then disk service demands can be decreased by an appropriate amount to represent fixing them in memory.

As a final example, additional memory can be used to increase the size of the disk cache employed by many operating systems. Experimentation with a few different cache sizes would indicate the relationship between disk cache size and disk cache hits (and thus I/O activity).

13.3.3. I/O Subsystem Modifications

Each generation of disks can be characterized by basic quantities such as capacity, seek time, latency time, and transfer rate. From these characteristics it is possible to estimate the changes in disk service demands that will result from replacing one type of disk with another. For example, due to faster seeks and higher transfer rates, service demands are reduced by 25% to 30% when converting from IBM 3350 to IBM 3380 disks. The exact speed ratio depends on block size, seek pattern, and I/O subsystem contention.

A secondary effect to consider in this case is the fact that, because the capacity of a 3380 is nearly double that of a 3350, there is a temptation to reduce the number of drives as part of a conversion effort. The resulting change in seek patterns may cause the average seek distance to increase, making it more difficult to forecast service demands.

Recently, solid state drums have provided a new alternative in I/O subsystems. These devices have limited capacity, but provide much faster access times than conventional disks or drums (factors of 4:1 currently). In modelling the addition of a solid state drum to a system, several steps are required:

- Identify the files to be placed on the drum. (Typically, these will be small, highly active files.)

- Reduce the service demands on the disks from which these files will be removed.
- Add a new center to the model and set the service demand there to be a fraction of the service demands removed from the disks, determined by the relative speeds of the devices.

An I/O subsystem can be upgraded by increasing the numbers of channels and controllers or by changing the interconnections among existing components, as well as by adding storage devices. Changes of this sort would be expected to reduce contention in the I/O subsystem by creating alternate paths between the CPU and the disks. Consequently, the contention component of effective disk service demands would be reduced. The techniques suggested in Chapter 10 are oriented towards assessing the effect of this sort of modification.

13.4. Changes to the Operating Policies and System Software

Operating systems typically leave a great deal of flexibility to installations with respect to certain operating policies that can have a significant influence on performance: placement of files on devices, assignment of workload components to memory domains, setting of scheduling priorities, etc. The first three subsections that follow discuss the representation of modifications to such operating policies in queueing network models. The fourth subsection discusses the representation of the effect of operating system upgrades.

13.4.1. File Placement

Performance often can be improved by altering the assignment of files to devices, with the objective of balancing the load across disks and other I/O subsystem components. The parameter changes to represent such modifications in the model are straightforward. If the disks involved are identical, then the primary effect can be represented (in the case of three disks) by:

$$D'_{each\ disk} \leftarrow \frac{D_{Disk\ 1} + D_{Disk\ 2} + D_{Disk\ 3}}{3}$$

If a decrease in the contention component of the effective disk service demands is expected, then the techniques of Chapter 10 should be used, with the analyst balancing the seek, latency, and transfer components as above.

If the devices involved differ in speed, more effort is required. The service demands in the baseline model must be viewed as the product of visit counts and service times per visit. The service demand at each of k disks after balancing is given by the equations:

$$D'_{each\ disk} = V'_{Disk\ 1}\ S_{Disk\ 1} = \dots = V'_{Disk\ k}\ S_{Disk\ k}$$

and:

$$\sum_{j=1}^{k} V'_{Disk\ j} = \sum_{j=1}^{k} V_{Disk\ j}$$

Thus, we assume that balancing the load does not change the service times at the devices substantially (e.g., by changing seek patterns), and that the total number of physical I/O operations does not change. The service demand for each disk will be:

$$D'_{each\ disk} = \frac{\sum_{j=1}^{k} V_{Disk\ j}}{\sum_{j=1}^{k} (1 / S_{Disk\ j})}$$

Thus, we would seek an assignment of files to disks such that capacity constraints are not exceeded and the visit count to files assigned to each disk approximately satisfy:

$$V'_{Disk\ j} = \frac{D'_{each\ disk}}{S_{Disk\ j}}$$

The approach described above generally will succeed only in approximately balancing the I/O load. The service times at the various disks in fact will change due to altered seek patterns and other secondary effects. Also, carefully balancing the I/O load according to access patterns observed during one period of the day will not lead to a balanced load throughout the day. Consequently, in doing I/O balancing, peak load periods should be given most consideration, but implications for other periods should be considered.

When representing the addition of disks to a configuration, it is appropriate to attempt I/O load balancing at the same time. An example in Section 13.6 illustrates the evaluation of the effect of I/O load balancing through altering the placement of user files.

13.4.2. Memory Allocation

The allocation of memory is critical to performance. An operating system typically requires substantial memory for its own use, devoted to resident code and data structures, transient routines, and I/O buffers.

The remaining memory is allocated to user programs. As noted in Chapter 9, it is typical to define domains with limited capacities and to assign workload components to these domains. This approach regulates competition for memory so that thrashing does not occur.

The primary effect of altering the allocation of memory can be represented by changing the domain capacities in the model (the multiprogramming level, in the case of batch classes). The problems that arise are similar to those that arise in modelling the addition of memory, which were discussed in an earlier section. Especially in a virtual memory system, it can be difficult to determine the number of jobs that can be accommodated in a specific amount of memory. Limited benchmarking can be of assistance in determining how the rate of paging depends on the amount of main memory available for each active customer.

13.4.3. Tuning Parameters

In most operating systems, many of the scheduling and resource allocation activities are controlled by tuning parameters. Among other things, such parameters control the dispatching and initiation priorities of various workload components, and the amount of service guaranteed to customers before they are eligible to be swapped out. Queueing network models can be used to gain an understanding of the effects of changing certain tuning parameters. The major benefit of such studies is to estimate the extent to which performance might be affected by a particular parameter.

Representing the effect of changes in the relative priorities of workload components is straightforward, using the techniques described in Chapter 11. Chapter 16 includes an example of such a study.

The swapping quantum (the amount of service guaranteed a customer before becoming eligible for swapping) is another example of an important tuning parameter. The case study of Section 9.6.2 illustrates the incorporation of this parameter in a queueing network model.

13.4.4. Operating System Upgrades

Operating systems provide certain services to the programs that execute under them. The variety of services available and the efficiency with which they are delivered differs from one system to another. The operating systems for most major computer systems evolve continually. Each version (or "release") typically provides some new functions, and possibly improves the efficiency with which earlier functions are delivered.

To model the effect of an operating system upgrade, the analyst must determine the relative efficiency of various functions by relying either on statements by the vendor or on experience of early users ("beta test" sites). Given this information, modification of the model is straightforward. For example, if it is claimed that CPU path lengths for user I/O processing will be decreased by a factor of two, the analyst first must determine this overhead component of CPU service demand for the workload on the existing system, then divide it by two to represent the effect of the new release.

Operating system efficiency also is of importance when comparing various systems under consideration for the support of a new workload. In this case, it is necessary to translate a workload description in system-independent terms into service demands for each candidate system. In the case of CPU service demands, for example, the relative CPU execution rates of the various systems tell only part of the story: the efficiency of operating software can have a dramatic effect on performance. As we showed in an example in Section 2.4, simple, single-thread benchmarking experiments are appropriate and useful in quantifying software efficiencies for incorporation in queueing network models.

13.5. Secondary Effects of Changes

Previous sections have concentrated on the representation of the primary effects of system changes. In the present section we consider the representation of certain secondary effects that are common to a number of the modifications we have discussed.

To a certain extent, these issues already have been addressed in Part III of the book. In Chapter 9, we showed one approach to estimating the change in swapping activity that would accompany various system modifications. We also showed how variability in paging activity could be incorporated in a model. In Chapter 10, we developed algorithms to estimate path contention in complex I/O subsystems as a function of other system characteristics. In Chapter 11, we mentioned the representation of the CPU overhead that accompanies all other activities.

In this section, we will talk in more general terms about techniques to forecast the level of CPU and I/O overhead present in a system. Our approach will be one that was suggested in earlier chapters: to extrapolate from the results of a few measurement intervals.

13.5.1. Changes in Variable Overhead

Almost every contemplated change to a system will, as a secondary effect, change the variable overhead incurred in system operation. The most significant examples of this in many systems are changes in paging and swapping rates, which involve both CPU and I/O activity. CPU upgrades, memory expansions, increases in workload intensities, even changes in the priority structure among classes, all have the secondary effect of changing paging and swapping rates.

As we have noted in earlier chapters, when a model is used to project performance for relatively minor modifications (a 10% increase in workload intensity, a 25% increase in CPU capacity), changes in variable overhead need not be considered. The more significant the modification under consideration, the more important it is to attempt to quantify these changes. This is a difficult task; in some cases it will be necessary to employ a sensitivity analysis to indicate the range of anticipated performance.

1. Obtain measurements from several observation intervals, preferably including a range of degrees of system congestion.
2. For each interval, determine the service demand at each center.
3. For the measure of system congestion of greatest concern (e.g., workload intensity), for each center, fit a simple curve to the observed service demands as a function of the measure of concern.
4. Use the simple curve for each center to extrapolate service demand for unobserved situations.

Algorithm 13.1 – Variable Overhead in Single Class Models

An approach to characterizing variable overhead for single class models based on measurements from several observation intervals is given as Algorithm 13.1. The simplest curve to use in Algorithm 13.1 is a straight line. This suffices for representing variable overhead as long as the range of congestion being investigated is not extreme. Assume that measurements are available for two observation intervals in which the workload intensities are $I^{(1)}$ and $I^{(2)}$, respectively, and in which the observed service demands at device k are $D_k^{(1)}$ and $D_k^{(2)}$, respectively. Assuming that variable overhead increases linearly with workload intensity, an appropriate estimate for the service demand at device k for a new workload intensity I' is given by:

$$D'_k = D_k^{(1)} + (I' - I^{(1)}) \times \left[\frac{D_k^{(2)} - D_k^{(1)}}{I^{(2)} - I^{(1)}} \right]$$

Approximating the dependence of variable overhead on workload intensity by more complex curves typically yields slightly greater accuracy, particularly if workload intensity changes are large, but this gain may not justify the added complexity.

Careful treatment of variable overhead is more difficult in multiple class models. There are several issues involved:

- In the multiple class case, more observation intervals are necessary, because the workload intensity now is a vector. For example, if the workload consists of two major components, interactive and batch, we might consider four observation intervals: heavy batch and heavy interactive, heavy batch and light interactive, light batch and heavy interactive, and light batch and light interactive.
- Within each observation interval, it is difficult to attribute variable overhead to the classes accurately, because of the inadequacy of measurement tools. Techniques such as those described in Section 12.5 can be used.
- Where the single class case involved fitting a curve through some points, the analogous procedure for the multiple class case with C classes involves fitting a C-dimensional surface. Such multidimensional surface fitting, however, is too complex to be justified considering other limitations on the accuracy of this technique. In almost all cases, a sequence of one-dimensional extrapolations based on changes to one workload component at a time will suffice.

From the preceding discussion, it should be apparent that estimating changes in variable overhead is difficult, and cannot be done with high confidence. Consequently, it often is appropriate to evaluate the model under both optimistic and pessimistic assumptions in order to assess the importance of accurately estimating overhead in projecting performance. For example, when memory size is increased, paging and swapping activity typically are reduced. Because it is difficult to determine the extent of this reduction, we might evaluate the model once assuming no change in paging and swapping activity, and again assuming that all paging and swapping activity is eliminated.

13.5.2. Changes in I/O Service Times

Many modifications have the secondary effect of changing the seek, transfer, and contention components of effective disk service time. The contention component was considered in Chapter 10. Here we discuss the others.

Relocating files from one disk to another can cause the seek patterns to change on each disk. Typically, the average seek time will increase on the disk to which the file is moved and will decrease on the other. If all files do not have the same block size, then the average transfer times at both disks also will be altered.

Similar considerations arise in such system modifications as increasing the block size of a file or increasing the workload intensity of a class (which can alter the seek pattern and change the average transfer time if the class accesses some files particularly heavily).

13.6. Case Studies

In this section we describe three case studies conducted over a period of several years on an evolving UNIVAC 1100 system running the Exec 8 operating system. Initially the system was configured as an 1100/41 (a uniprocessor) with the following I/O subsystem structure:

channel 0	1 FH-1782 drum
channel 1	1 FH-1782 drum
channel 2	4 tape drives
channel 3	8 8424 disk drives
channel 4	4 8433 disk drives

In each of the three case studies, synthetic benchmarks designed to reflect actual workloads were used, and the same experimental procedure was followed:

- The benchmark was run on the existing configuration and measurements were taken with UNIVAC's SIP (the Software Instrumentation Package).
- A baseline queueing network model was developed and validated.
- The model was modified to project the effect on performance of a specific proposed change to the system.
- This change was implemented, and the benchmark was run again.
- The performance projected by the model was compared to the performance measured on the modified system.

Note that this experimental procedure follows closely the modelling cycle described in Section 2.2. It is this aspect that makes these three case studies particularly interesting in the context of the present chapter. On the one hand, the parameter adjustments used to project performance occasionally were somewhat simplistic, in that obvious secondary effects were ignored. On the other hand, retrospective attempts were made to attribute discrepancies between projections and measurements to specific secondary effects. The sequence of case studies thus is a good example of how lessons learned in one study can be used to improve the accuracy of subsequent studies. In a production environment where decisions are made after the performance projection step, there is a tendency to omit the final two steps of the procedure outlined above. These steps are important, however.

13.6.1. Moving to a Dual Processor

In the first study, a baseline model of a uniprocessor system (an 1100/41) was modified to project the performance of a dual processor system (an 1100/42). The model contained a single class of batch type and six service centers: one representing the CPU (or pair of CPUs) and five representing the five I/O channels of the system. The use of centers to represent channels rather than disks differs from the approach suggested in Chapter 10. This case study pre-dates that approach. Further, the channels had considerably higher utilizations than the disks in this system, and thus were thought to be the principal constraints on performance. Also, reliable measurements of busy times were available for the channels but not for the disks.

In this study six different benchmarks were used. After each benchmark was run on the uniprocessor system, measurement data was used to parameterize the baseline model, as follows:

- The service demand at the CPU center was set to the CPU busy time divided by the number of job completions.
- At the five centers representing the channels, the service demands were set to the corresponding channel busy times divided by the number of job completions. (Note that the seek component of disk service times was not represented in this model.)
- SIP provided an estimate of multiprogramming level that was known to be unreliable. Consequently, the value of N was adjusted until the throughput of the model exactly matched that of the system.

The technique of establishing the value of some parameter according to what yields the best results is called *calibration.* It should be avoided unless legitimate uncertainty exists concerning the value of a single parameter.

This baseline model then was modified to reflect the addition of the second CPU. This was done by replacing the CPU center with an FESC. With one customer present, the FESC service rate was the same as that of the uniprocessor. With two or more customers present, it was double this value. That is:

$$\mu(n) = \begin{cases} \dfrac{1}{D_{CPU}} & n = 1 \\ \dfrac{2}{D_{CPU}} & n > 1 \end{cases}$$

where D_{CPU} was the processor service demand in the baseline model.

benchmark	throughput			
	original	projected	actual	error
1	48.2	53.8	48.0	+ 12%
2	47.8	67.3	48.9	+ 38%
3	48.9	55.0	50.9	+ 8%
4	39.9	56.8	50.1	+ 14%
5	33.7	47.0	45.9	+ 2%
6	40.4	57.1	59.9	− 5%

Table 13.1 – Moving to a Dual Processor

Table 13.1 compares the projections of the model to the measured performance after the second processor was added for each of the six benchmarks. The error in projected throughput was 15% or less in five of the six cases, but 5% or less in only two. This cannot be viewed as successful, especially in light of the 38% discrepancy in the sixth case.

A retrospective analysis revealed that the CPU upgrade caused the average multiprogramming level to drop substantially – to one half its former value for four of the six benchmarks. This likely was the reason for the counter-intuitive fact that the addition of the second processor made essentially no difference in measured performance for the first three benchmarks. Even with the benefit of hindsight, it was difficult to understand why this drop in multiprogramming level occurred. (Conceivably it was indicative of a shortcoming in the system's job scheduler.) The assumption that the multiprogramming level would not change with the addition of the second processor played a substantial role in the optimistic throughputs projected for five of the six benchmarks.

A second factor that contributed somewhat to the optimistic projections was that interference between the two processors was not taken into account in determining the rates of the FESC. As was noted in Chapter 11, the full power of the second CPU is not realized in dual processor systems; $\mu(n)$ for n greater than one should have been set to a value less than $2/D_{CPU}$.

Finally, no change in the number of swaps per job was anticipated or represented in modifying the model parameter values. In fact, the number of swaps per job decreased, possibly due to the reduced multiprogramming level. This meant that the average number of visits made by a job to channel 1 (the location of the swapping drum) decreased, and also that the average service time per visit was reduced (because swapping operations had much higher average service times than did user I/O operations at this device). This effect was not large, and was more than offset by the other, optimistic discrepancies.

13.6.2. Altering File Placement

The second case study was an investigation of the effect of balancing the load across channels by altering the placement of user files. By the time of this study, the configuration had evolved somewhat. Specifically, the disk channels had been converted to "dual channels": two disks on the same dual channel could be active (in any phase, even data transfer) simultaneously. Thus, performance measures of the two studies are not directly comparable.

The model employed was similar to that used in the first study. Once again, there was a single class of batch type. Again, an FESC was used to model the dual processor of the UNIVAC 1100/42 configuration. The other centers in the model corresponded to channels. Each of the dual channels was modelled as an FESC that behaved similarly to the FESC used to represent the dual processor CPU.

A single benchmark was run on the system with the original assignment of user files to devices. Data from both SIP and UNIVAC IOTRACE was used to parameterize the baseline model. (IOTRACE reported the channel busy time due to accesses of each individual file.) Most of the model parameters were established in conventional ways. The centers representing the dual channels required special attention, however. One channel in each pair was the *primary* and was used whenever available. The other was the *secondary* and was used only when necessary. The service rates of each FESC were calculated as:

$$\mu(n) = \begin{cases} \dfrac{C_{prim}}{B_{prim}} & n=1 \\[2ex] \dfrac{C_{prim}}{B_{prim}} + \dfrac{C_{sec}}{B_{sec}} & n>1 \end{cases}$$

where *prim* and *sec* denote the primary and secondary channels of the pair, C_k is the measured number of operations on channel k, and B_k is the measured busy time of channel k. (Note that this calculation ignores the fact that the secondary channel is blocked if the request it is serving happens to access the same disk as the request being served by the primary channel.) Once again it was necessary to determine the multiprogramming level N by calibrating on throughput.

User files accounted for only 30% of the measured I/O accesses. The other 70% of the accesses were to system files whose placement was considered fixed in this experiment. Two alterations in the existing placement of user files were considered:

- Place all user files on the 8433 disks associated with channel 4. A careful analysis indicated that this would result in the greatest performance improvement — a "best case" scenario.
- Place all user files on devices attached to the most heavily utilized channel. It was believed that this would result in the greatest performance degradation — a "worst case" included for comparison.

The parameters of the baseline model were adjusted to represent each of these file placements, using techniques similar to those suggested earlier in this chapter. After model projections were obtained for each case, the files actually were moved, and the benchmark was run again for each case.

case	**quantity**	**original**	**projected**	**actual**	**error in projection**
best	U_{CPU}	.843	.888	.881	+0.8%
	X	79.5	86.7	84.5	+2.6%
worst	U_{CPU}	.843	.762	.640	+19.1%
	X	79.5	68.0	59.6	+14.1%

Table 13.2 — Altering File Placement

The results for both cases are shown in Table 13.2. The last column indicates the error in the projection relative to the observed value. The results show that throughput for the best placement of user files, which account for only 30% of I/O accesses, is roughly 5% greater than for the existing placement, and roughly 35% greater than for the worst placement. The accuracy of the model for the best placement is quite good, while for the worst placement it is acceptable but not good.

Retrospectively, it was observed that the major source of error was the fact that the model ignored changes in swapping behavior that accompanied the alterations in file placement. The worst case scenario caused many user files to be located on the drum containing the swap data set. Swap operations took longer, the CPU was left idle more often (because jobs were not available for service while being swapped), the scheduler activated more jobs to try to keep the CPU busy, and swapping (and the associated channel congestion) increased. The model, which assumed that swapping would be unaffected, underestimated the deterioration in performance. (The best placement caused some files to be removed from the swapping drum, leading to some reduction in swapping, but the effect was not significant.)

13.6.3. Moving Swapping Activity from Drum to Disk

A third study of the same system considered the effect of moving swapping activity from drum to disk. The disks were under-utilized relative to the drums, and their newer technology and dual channel capability made them competitive in terms of performance. By moving swapping activity to disk, the drums could be used for temporary data sets, accessed frequently during their short lifetimes.

In constructing the baseline model, additional detail in the representation of the I/O subsystem was incorporated. Centers were included to represent each disk, in addition to the FESCs representing the two dual channels. So that no component of I/O service demand would be duplicated at the disk and channel centers, the disk centers represented only seek times, while the channel centers represented latency and transfer times. Because this approach tends to yield optimistic results (in the model, one customer's seek activity at a disk can be overlapped with another customer's latency and transfer activity at the same disk), the disk centers were represented as FESCs whose service rates *decreased* when more than one customer was present.

Remembering the lessons from the first two studies, thought was given to examining both primary and secondary effects of the proposed modification. The procedure used to adjust the parameters of the baseline model to reflect the movement of swapping from drum to disk was iterative in nature:

- Assume initially that the level of swapping activity will remain unchanged after the modification.
- Knowing that the operating system tends to place temporary files on faster devices, estimate the visit counts at drums and disks that would result from moving all swapping activity to disk.
- Knowing the files placed on each device, the relative access frequencies to files, and the average transfer size for each file, adjust the service demands at the centers representing the drums, and the service rates at the FESCs representing the disks and dual channels.
- Evaluate the model.
- Use an empirically derived relationship between throughput, multiprogramming level, and swapping activity to estimate the change in the level of swapping activity resulting from the modification.
- Return to the second step, iterating until convergence is achieved.

As in the two earlier case studies, the change to the system was implemented and the benchmark was run once again. Table 13.3 displays the results. This experiment was successful in producing usefully accurate performance projections.

quantity	original	projected	actual	error in projection
CPU utilization	.609	.665	.679	− 2.1%
CPU queue length	1.08	1.32	1.39	− 5.0%
throughput	101	115	121	− 5.0%

Table 13.3 — Moving Swapping Activity from Drum to Disk

13.7. Summary

The principal value of a validated queueing network model of a baseline system is its utility as a basis for performance projection. In this chapter we have indicated, through discussion and example, how to modify the parameters of a baseline model to represent various common changes to the workload, to the hardware, and to the system software and operating policies.

A key point to keep in mind in conducting a modification analysis, especially as part of a study in which a large number of alternatives must be considered, is the need to identify those effects of the modification that are primary, and those that are secondary.

Primary effects typically are easy to anticipate and to represent. In the early stages of a study, alternatives can be compared on the basis of their primary effects alone.

Secondary effects typically are less easy to anticipate, and even once anticipated, less easy to quantify and represent. Several approaches can be adopted:

- Extreme assumptions can be evaluated; for example, the addition of memory at worst leaves swapping unaffected, and at best eliminates it.
- A more careful estimate of secondary effects can be made, based on measurements from several observation intervals.
- A sensitivity analysis can be used to assess the extent to which the projections of a model depend upon the assumptions that have been made.

We have tried to indicate the importance of the "verification phase" of the modelling cycle, described in Chapter 2. Expertise and confidence in conducting modification analyses is best acquired by learning from prior modelling experiences.

13.8. References

There have been a number of case studies in which a baseline model was constructed, performance projections were obtained, and the accuracy of these projections was checked after the system had been modified. The three studies described in Section 13.6 all were carried out at the University of Maryland, using facilities available at the computer center there. The study of moving to a dual processor was conducted by Dowdy, Agrawala, Gordon, and Tripathi [1979]. The study of altering file placement was conducted by Dowdy and Budd [1982]. The study of moving swapping activity from drum to disk was conducted by Dowdy and Breitenlohner [1981].

Several similar studies from production environments were reviewed in Chapter 2: by Lo [1980] on the effect of reallocating workloads among systems in a multiple mainframe environment, and by Lazowska [1980] and Sevcik, Graham, and Zahorjan [1980] on evaluating various candidate systems for specified applications.

There are several related papers that we have not discussed specifically. Tibbs and Kelly use quadratic fits obtained by non-linear regression to forecast the change in overhead in doing performance projections for a UNIVAC 1100 [Tibbs & Kelly 1982]. Bard [1978] describes a performance projection tool for systems running IBM's VM/370 operating system. Buzen presents a queueing network model of systems running IBM's MVS operating system, which includes the effects of shared memory domains and performance periods [Buzen 1978]. Models of DECsystem-10 systems running TOPS-10 have been described by Saxton and Lamont [1978] and by Sanguinetti and Billington [1980]. Dowdy, Stephens, and Perez-Davila [Dowdy et al. 1982] have done a study of performance projection in a UNIX environment, in which the treatment of memory management was the principal issue.

In the realm of workload forecasting, Artis proposes a way of estimating what the workload of a system would be if sufficient capacity were provided to handle it [Artis 1981]. Cooper [1980] describes an approach to capacity planning in an organization, which integrates business planning forecasts with the use of models of computer system performance.

[Artis 1981]
H. Pat Artis. Estimating Latent Demand for Random Arrival Batch Workloads. *Computer Performance 2*,1 (March 1981), 26-29.

[Bard 1978]
Y. Bard. The VM/370 Performance Predictor. *Computing Surveys 10*,3 (September 1978), 333-342.

[Buzen 1978]
Jeffrey P. Buzen. A Queueing Network Model of MVS. *Computing Surveys 10*,3 (September 1978), 319-331.

[Cooper 1980]
J.C. Cooper. A Capacity Planning Methodology. *IBM Systems Journal 19*,1 (1980), 28-45.

[Dowdy & Breitenlohner 1981]
Lawrence W. Dowdy and Hans J. Breitenlohner. A Model of Univac 1100/42 Swapping. *Proc. ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems* (1981), 36-47. Copyright © 1981 by the Association for Computing Machinery.

[Dowdy & Budd 1982]
Lawrence W. Dowdy and Rosemary M. Budd. File Placement Using Predictive Queueing Models. In R.L. Disney and T.J. Ott (eds.), *Applied Probability - Computer Science: The Interface, Vol. II.* Birkhauser, 1982, 459-476.

[Dowdy et al. 1979]
Lawrence W. Dowdy, Ashok K. Agrawala, Karen D. Gordon, and Satish K. Tripathi. Computer Performance Prediction via Analytical Modeling — An Experiment. *Proc. ACM SIGMETRICS Conference on Simulation, Measurement and Modeling of Computer Systems* (1979), 13-18. Copyright © 1979 by the Association for Computing Machinery.

[Dowdy et al. 1982]
Lawrence W. Dowdy, Lindsey E. Stephens, and Alfredo Perez-Davila. Performance Prediction in a UNIX Environment. *Proc. 18th CPEUG Meeting* (1982), 205-211.

[Lazowska 1980]
Edward D. Lazowska. The Use of Analytic Modelling in System Selection. *Proc. CMG XI International Conference* (1980), 63-69.

[Lo 1980]
T.L. Lo. Computer Capacity Planning Using Queueing Network Models. *Proc. IFIP W.G.7.3 International Symposium on Computer Performance Modelling, Measurement, and Evaluation* (1980), 145-152.

[Sanguinetti & Billington 1980]
John Sanguinetti and Richard Billington. A Multi-Class Queueing Network Model Of An Interactive System. *Proc. CMG XI International Conference* (1980), 50-55.

[Saxton & Lamont 1978]
Harold E. Saxton and Gary B. Lamont. Validation of a DEC-10 Closed Queueing Network Model. *Proc. CMG IX International Conference* (1978), 143-151.

[Sevcik et al. 1980]
K.C. Sevcik, G.S. Graham, and J. Zahorjan. Configuration and Capacity Planning in a Distributed Processing System. *Proc. 16th CPEUG Meeting* (1980), 165-171.

[Tibbs & Kelly 1982]
Richard W. Tibbs and John C. Kelly. The Application of Analytic and Simulation Models to Size a Large Computer System. *Proc. 18th CPEUG Meeting* (1982), 231-257.

13.9. Exercises

1. Expand on Exercise 1 of Chapter 12. For each of the case studies, indicate how the model you proposed could be modified to represent the primary effects of the system change being investigated. In addition, consider what secondary effects should be represented.
2. A group of files are stored on a disk and a drum with service times of 30 and 10 milliseconds per access, respectively. Currently, the service demands at the disk and drum are 6 and 3 seconds, respectively. Consider each of the following scenarios for changing the system:
   a. Knowing the relative access counts for the files, indicate how you would relocate files in order to balance the demand on the two devices.
   b. If the disk were replaced by a second drum, and the demand were balanced across the two devices, what would the service demand at each be?
   c. If all files on the disk were moved to a solid state drum with a service time of 2 milliseconds per access, what would be the resulting service demand?
3. Consider a system with a single batch class in which each customer has a CPU service demand of 30 seconds and does 1000 I/O operations involving a total of four files: 400 accesses to file W, 300 to file X, 200 to file Y, and 100 to file Z. The files can be placed on three I/O devices with service times per access of 10 milliseconds at device

1, 30 milliseconds at device 2, and 50 milliseconds at device 3. Using a single class queueing network solution package such as the one provided in Chapter 18, determine how to assign the files to the storage devices to maximize throughput, for each of the following situations:

a. Multiprogramming level is 1.

b. Multiprogramming level is 4.

c. Multiprogramming level is 12.

(Assume that each device has sufficient capacity to accommodate whatever files you choose to assign there.)

4. Three observation intervals yield the following information:

quantity	measurement interval		
	1	2	3
jobs completed	600	800	500
CPU busy time	14400	20800	11500
multiprogramming level	4	6	3

In projecting performance for a multiprogramming level of 10, what service demand should be used to reflect a simple linear model of variable overhead?

5. Suppose that you had two single class models, one open and one closed. Suppose that these models were "equivalent" in the sense that they had identical service centers, identical service demands, and identical throughputs and utilizations.

   a. Would you expect these models to have identical queue lengths and residence times? Why or why not?

   b. If you were to modify the open model by doubling the arrival rate and the closed model by doubling the population, how would you expect the changes in performance measures to differ between the two models?

   c. Doubling the arrival rate of an open model and doubling the population of a closed model correspond to two very different "scenarios" about the future of a system. State the system change that is addressed by each of these modifications.