CHAPTER

1 Performance, Efficiency, and Tuning

Introduction

So, how are your SAS applications performing on OS/390?* Are SAS batch jobs completing on time and consuming a minimum amount of system resources? Are SAS interactive applications receiving good response time? Are the computer usage charges for your group's SAS applications high or low? Have recent changes to SAS code or to data affected the performance of your applications?

If you don't know the answers to these questions, don't worry; you are hardly alone. Traditionally, SAS applications programmers work at supplying management with relevant business information. They automate manual processes, reduce and summarize data, and perform statistical analyses. They create reports and charts and graphs characterizing metrics of interest to the organization. They focus on how they can use the SAS language to process the data and on what the data reveals. Issues of performance and efficiency, while interesting, are not their main concern.

However, the success of SAS within organizations is changing the way SAS programmers look at performance. While SAS is an excellent tool for data exploration and ad hoc reports, many organizations are using it as a full-blown applications development language. This means that scores of batch applications are being developed in the SAS language and implemented into production. Interactive SAS applications are being written for groups of end users who navigate large SAS data sets as if they were databases. The portability of SAS among different computer platforms has made organizations embrace it as an enterprise-wide tool. As the volume of SAS applications grows within an organization, the cost of those applications to the organization gains higher visibility.

More and more often, SAS programmers are being asked to improve the efficiency of their programs. The batch windows in which their SAS applications run are shrinking. Their end users are demanding quicker turnaround time for reports. Users of online applications need and expect

* In the past fifteen years, the mainframe operating system that is popularly called "MVS" has evolved from MVS/ESA, to OS/390, and finally to z/OS. The material in this book is relevant to SAS running on all of the aforementioned operating systems. As of this writing, many sites are just beginning to migrate from OS/390 to z/OS. To keep references to the operating system simple, the term OS/390 will be used exclusively throughout the book.

faster response time. Computer costs are climbing for SAS applications, and end users are becoming very cost conscious. These, and other, factors are putting pressure on SAS programmers to look at the performance and efficiency of their applications and to tune them. But, what exactly *is* "performance" or "efficiency" or an "application"? And, how do you *tune* an application?

This chapter is divided into two sections. The first section, "Performance and Efficiency Overview," discusses the basic concepts of SAS program performance and efficiency. It also examines the advantages of efficient SAS applications to programmers, end users, and the organization. The second section, "The Tuning Cycle," outlines a specific methodology for tuning SAS applications in a controlled, step-wise manner. It begins by providing an overview of the basic tuning cycle and a discussion of setting application performance standards. Then it details how to implement the tuning cycle to tune real-world SAS applications at the task, program, and application levels. This section continues with a discussion on how to create an application performance "snapshot." The final pages of the chapter are devoted to three sample tuning worksheets.

Performance and Efficiency Overview

What Is a SAS Application?

The word *application*, as used in this book, is defined as one or more related SAS language programs that perform a specific function for an organization on a cyclic basis. An application can be as simple as a single SAS program executed in batch mode that reads data from a tape and creates a report. It can be as complex as a dozen batch jobs executing scores of SAS programs to process the payroll for an organization. Similarly, an application can be a group of SAS/AF programs used by hundreds of sales personnel to determine product availability. Or, it can be a library of SAS/IntrNet programs that provides corporate information to Web users, worldwide. Whether simple or complex, an application is executed on a predictable basis: continuously, hourly, daily, weekly, monthly, quarterly, or yearly.

There is a distinction between SAS applications and SAS ad hoc programs. Ad hoc programs are created as one-time efforts to reduce, analyze, and present data. Requests for their creation and execution are generally unpredictable. Ad hoc programs differ from SAS applications, which run in a production mode on a predictable, cyclic basis. Though the focus of this book is on SAS applications, the performance of SAS ad hoc programs is also considered important. The efficiency techniques discussed in the following chapters apply to all SAS programs run in the OS/390 environment.

Understanding Efficiency

The first step in discussing the efficiency of SAS applications is to define the word *efficient*. An efficient SAS program is one that uses the minimum amount of computer resources possible to complete its task. Therefore, an *efficient SAS application* is one in which all of the component programs are efficient. *Efficiency* is the *degree* to which programs are efficient. Efficiency is tied directly to the consumption of OS/390 computer resources. (See Chapter 2 for a discussion of important OS/390 resources.) If a SAS application

cannot be modified so that computer overhead is further reduced, then the application is efficient.

Remember that it will always take some amount of computer resources to execute a SAS program. There is a point at which the computer overhead expended for a given task cannot be reduced if the task is to be completed. An efficient program runs at this point. In an efficient SAS program, the ratio of overhead to output cannot be reduced any further without affecting the output.

A program is not necessarily inefficient because it consumes a lot of computer resources. It is not the volume of expended computer overhead that determines efficiency. Rather, efficiency is determined by whether the overhead of a program can be reduced and the same output still be achieved. If the answer is “no” for a specific program, then it is an efficient program.

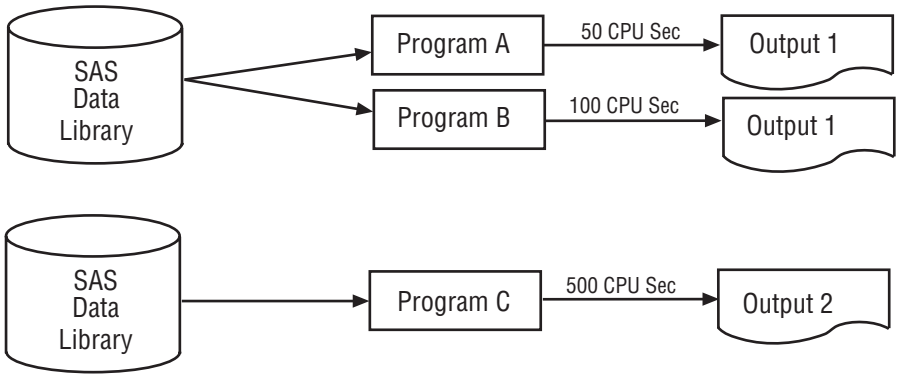

Figure 1.1: Examples of program efficiency.

Figure 1.1 illustrates program efficiency. In the first example, programs A and B read the same data set from the same SAS data library and create identical reports. Yet program A consumes 50 CPU seconds, while program B consumes 100 CPU seconds. Program A is more efficient than program B because its ratio of overhead to output is lower for the exact same task.

In the second example, program C reads data from a different SAS data library and creates a different report. Program C consumes 500 CPU seconds. This is significantly higher than either program A or B. Is program C efficient? The answer is “yes” if program C cannot be tuned any further and still produce the same report. The answer is “no” if some aspect of program C can be modified so that it produces the same report with less computer overhead.

Understanding Performance

Performance is directly related to efficiency. *Performance* is defined as a judgment of the relative overall efficiency of a SAS program or application. This judgment is made by measuring the resource consumption of a program and comparing it to other measurements. The other measurements could be from previous runs of the same program or from similar programs that perform like functions.

The performance of an individual SAS program is either good, bad, or unknown. Programs that have been tuned to their optimal efficiency are said to have *good performance*. Programs that waste computer resources and are inefficient have *bad performance*. A program's performance is *unknown* if it has not been measured or if there is nothing against which to compare a particular measurement.

The idea of what constitutes good performance can vary among organizations. This variance is due to different organizations having different mainframe computer resources that are in short supply. For instance, one organization may consider applications that use a lot of CPU time to be bad performers, while another might consider applications that consume a lot of EXCPs to be bad performers. (CPU time, EXCP count, and other OS/390 performance metrics used to gauge the performance of SAS applications are thoroughly discussed in Chapter 2.) Both outlooks are correct if the applications use unnecessary amounts of these resources to perform their tasks. Therefore, performance is generally tied to the perspective and needs of an organization.

An application's performance should not be judged by the absolute volume of computer resources that it uses. It is not the volume of resources consumed by an application that makes it either a good or a bad performer. Rather, it is the amount of *unnecessarily expended* resources that is the criterion for judging application performance. If a particular application consumes a large amount of critical system resources, but is executing as efficiently as is possible, it is a good performer.

Advantages of Efficient Applications

The preceding discussions of program efficiency are not designed to help you achieve some sort of esoteric, textbook realization of program purity. In this age of corporate downsizing, data center outsourcing, increased fiscal tightness, and business re-engineering there are many real-world advantages to reducing the overhead of SAS applications. Some advantages are for you, the programmer, some are for the end user, and some are for the organization.

Advantages for the Programmer

Whether you are a beginning, intermediate, or expert programmer, you can appreciate the flexibility of the SAS programming language. As your level of SAS experience grows, you are consistently challenged to find new ways to process and present data. This inevitably brings you into contact with the many procedures, functions, options, and other elements of the SAS language. As you encounter elements that are new to you, you move from discovery to mastery of them. Mastering program efficiency will add great value to your SAS programming skill set.

Mastering the ability to write efficient SAS applications makes you a more professional programmer. It enhances your skills by enabling you to deliver a streamlined product to your end users. Not only can you deliver the required output to your users, but you can do it with the minimum computer resources needed. This makes you a more valuable programmer within your group and within your organization.

Writing efficient SAS applications makes you a more competitive and sought-after programmer. Let's take the simple sports analogy of professional runners running a one-mile race. The runners all know they will complete the race; that fact is not in question. It is the speed and efficiency of their individual efforts that set them apart. The same is true for SAS applications programming. Most programmers can write the SAS code that produces the required output. However, a programmer who can produce the output quickly and efficiently has a competitive advantage over programmers who cannot.

Advantages for the End User

Whether you are a staff member of an organization or a contracting consultant, it is important that you create efficient applications for your end user. The word *end user* is used in this book to describe the recipient of the *end product* of the SAS applications you are creating and running. In that sense, the end user could vary from a nontechnical end user who executes a SAS/IntrNet application, to an information systems executive who receives a graph. No matter who the end users are, there are some definite advantages to making their applications as efficient as possible.

One advantage of efficient SAS batch applications is that they generally have reduced run time. Reducing batch run time allows end users to get their reports, graphs, or charts in a more timely manner. It limits the likelihood that batch jobs will be interrupted by external factors such as contention for system resources or interruption by higher priority tasks. Many organizations have shrinking or congested batch windows, so the sooner a job can be initiated and completed, the better. End users benefit by receiving quicker batch turnaround and getting their output faster.

Online response time can also be reduced by efficient SAS applications. Exposure to personal computer applications has created a generation of more demanding online users. They expect sub-second response time from their SAS applications running on the mainframe computers. Creating efficient online SAS applications helps end users get to the data they need as quickly as possible so they can concentrate on the data and not on the delivery system. Efficient SAS online applications allow end users to do their jobs more effectively.

In addition, efficient applications can reduce end-user computer processing charges. Today, many organizations use computer chargeback systems to manage and control information system expenditures. A computer chargeback system is an in-house accounting software package that charges end-user groups for the amount of computer resources their applications consume. The money that pays for the computer charges typically comes from the end-user group's operating budget. This motivates end users to review the costs of running their applications and to try to keep the costs low. Efficient SAS applications do not waste computer overhead, thus ensuring that computer charges are kept as low as possible.

Well-tuned SAS applications give end-user groups credibility in an Electronic Data Processing (EDP) audit. Some organizations have their data processing functions audited on a cyclic basis. This audit may come from an EDP audit group within the organization or from a regulatory agency. The

goal of an EDP audit is to determine if an end-user group is processing data in a manner that conforms to organizational or governmental standards. End-user groups that build efficiency into their SAS applications will be able to show EDP auditors that they have given their applications added value. This should help their overall standing when the auditors rank the groups within the organization.

Finally, efficient applications give a group competitive advantage within an organization. Most organizations are divided into groups based on the business functions they perform. (For example, a manufacturing organization might be composed of a personnel group, a payroll group, a sales group, a manufacturing group, and a distribution group.) The groups within an organization usually compete with each other for annual budgetary dollars. Groups that run efficient applications can reduce their data processing expenditures. They can either spend the money they save on non-data processing goods and services or run more applications for their usual data processing allotment. This makes the group more competitive within the enterprise. And, it shows the organization that the group's management and staff are fulfilling the group's mission in an exemplary fashion.

Advantages for the Organization

Efficient SAS applications offer many advantages to the entire organization. Whether the organization is an insurance company, a government agency, a bank, or a pharmaceutical company, it is still important to keep overhead expenditures low. The net effect of all SAS programmers' striving to keep their programs as efficient as possible is that the overall processing overhead of the organization is reduced. This has important consequences for the entire organization by helping to minimize the amount of money it has to spend on data processing services.

Efficient applications help the organization to better manage valuable computer resources. The amount of processing power in a mainframe computer is finite. An organization must get all of its data processing work done within the limits of the amount of resources available on its computers. Efficient SAS applications complete their work with the minimum possible expenditure of computer overhead. This means they are not wasting valuable processing power that can be used by other applications. The organization can schedule other applications to use the resources freed by well-tuned SAS applications.

Figure 1.2 illustrates the overall capacity of an organization's computer system. **Figure 1.2A** shows a system that is 100% utilized and has work waiting. The unnecessary overhead of inefficient applications has inflated resource usage and delayed other work from being processed. **Figure 1.2B** shows the same system after the unnecessary overhead has been eliminated through application tuning. Now, all of the organization's work gets completed.

Efficient SAS applications reduce competition for system resources. Competition among applications for processor time, data sets, work space, tape drives, etc., is reduced when the applications are efficient. The more quickly an application can complete its task, the sooner it is out of the

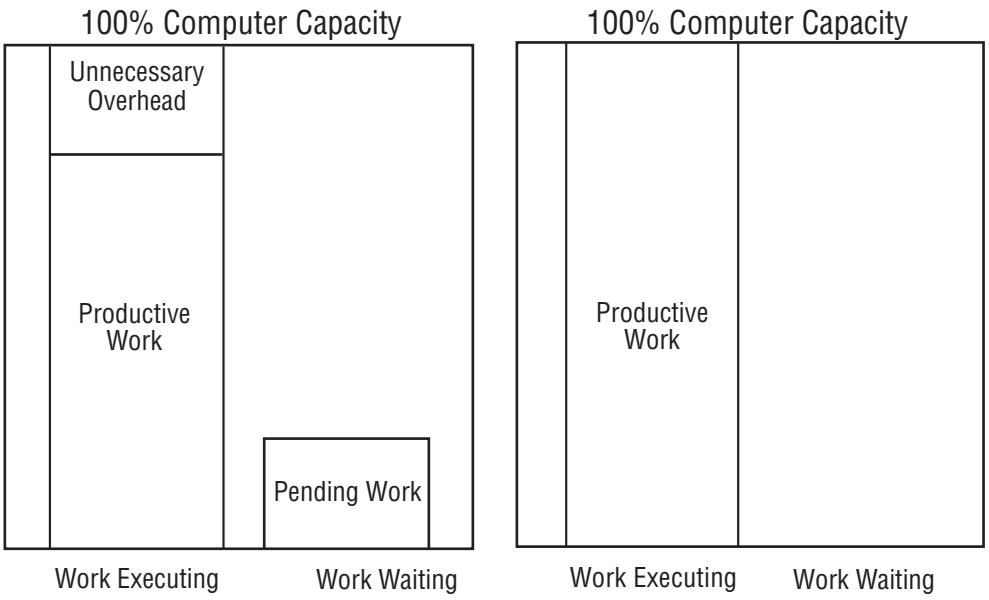

Figure 1.2: Total capacity of the computer system.

competition. Tasks with unnecessary overhead stay in the system longer than they really need to. Other applications wait for them to complete their usage of system resources. If these errant applications were more efficient, they would complete their work sooner and free resources for other applications to use. The net effect of reducing contention for resources is that more applications can be run on an organization's computers in a given time frame. This is especially helpful in the heavily utilized prime-time, 9:00 am to 5:00 pm shift.

Figure 1.3 illustrates the competition for system resources among applications. **Figure 1.3A** shows that for a given time interval only three applications can execute. Each of the three applications is inflated by unnecessary overhead, forcing a fourth application, application D, to wait. **Figure 1.3B** shows the same time interval with applications that have been tuned. Since each application uses fewer system resources, competition is reduced and all four complete their tasks in the given time interval.

Efficient SAS applications reduce an organization's need for a capital expenditure on a larger mainframe computer. Today, the cost of a new mainframe computer can run from several hundred thousand dollars to tens of millions of dollars. So, it is in an organization's best interest to delay a computer upgrade for as long as possible. Inefficient applications inflate an organization's computer overhead and hasten the day an upgrade is needed. An organization that can efficiently run all of its applications can put off the day when a capital outlay for the new processor is needed.

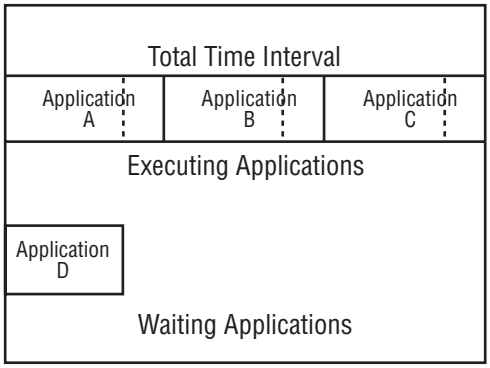

Figure 1.3A

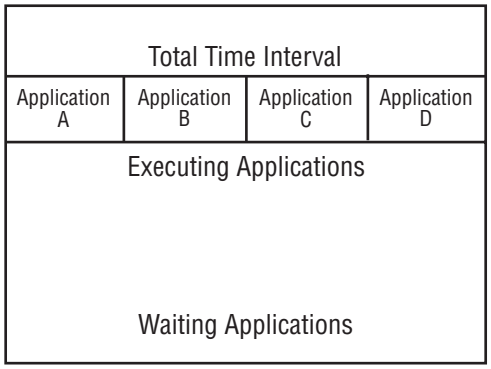

Figure 1.3B

Figure 1.3: Competition for system resources among applications.

The Tuning Cycle

The Need to Tune SAS Applications

It would be wonderful if every SAS application were executing at its peak level of efficiency. But many factors found in the typical work place make this unlikely. Deadlines sometimes force applications to be written quickly and hurried into production. Programmers with differing skill levels, experiences, and programming styles create programs with varying levels of efficiency. Data are often stored on different media and frequently occur in a wide variety of formats and block sizes. Performance options and strategies introduced in the latest releases of SAS may not be known to, or understood by, all staff programmers.

SAS programmers can do a lot to create efficient applications. They can use the good programming practices detailed in the SAS documentation. They can keep their knowledge of the SAS language current by attending local and national SAS Users Group meetings, and by reading papers published in the annual SAS Users Group International (SUGI) proceedings. They can follow and take part in discussions of SAS software issues posted to the SAS-L listserver by SAS users throughout the world. They can subscribe to the electronic newsletter *Your SAS Technology Report* at **`www.sas.com`**, and they can search the **`support.sas.com`** Web site for articles about tuning performance. These practices help SAS programmers to write applications that generally perform well. But, to ensure that their applications are running as efficiently as possible, programmers must actively tune them.

Tuning Cycle Overview

The best place to begin a discussion of the tuning cycle is with a definition of the word tuning. *Tuning*, as referred to in this book, is the act of modifying SAS programs, data, or system options to reduce processor overhead. Tuning is a conscious effort a programmer undertakes to create a more efficient program or application. As discussed previously, an application with component programs that have been tuned to their greatest efficiency is a good performer.

The tuning cycle begins with the assumption that a particular SAS application is not performing at its optimal level. This basic hypothesis is tested and evaluated in a series of specific events carefully orchestrated by the programmer. The completion of these events in a structured, step-wise manner results in an application tuned to its greatest efficiency.

The tuning cycle is composed of three steps: measurement, evaluation, and modification. Each step is executed in sequence to advance the performance goals of the programmer. Using this cycle, the programmer has a structured, scientific framework for evaluating and reducing the overhead of an application. **Figure 1.4** illustrates the flow of the tuning cycle. Each of the three major steps is examined in the following sections.

Measurement

Measurement is the basic element of the tuning cycle upon which everything else is based. There is an old saying that "what you do not measure you cannot control." This is essentially true for tuning the performance of SAS applications. Without measurement, you cannot know the resource overhead of an application. Without measurement, you cannot judge the performance ranking of one application relative to another. Without measurement, you cannot judge whether changes made to an application have affected its performance either favorably or adversely. Measurement provides the frame of reference for observing and evaluating the past, present, and future performance of SAS applications.

There are several specific metrics in the OS/390 environment that are measured and recorded for the tuning cycle. Chapter 2 discusses the metrics in detail, so they will be briefly mentioned here. CPU time, EXCP count, and memory utilization must all be measured for the SAS application being tuned. This means direct action must be taken to activate the measurements and to record the values of the metrics. (Chapter 3 discusses how to activate the metrics and where to find their values.)

The measurement taken at the very beginning of a tuning cycle is called the *baseline measurement*, or simply the *baseline*. This is the first measurement of an application, against which future measurements will be compared. It is important to accurately record a baseline for all of the programs that are likely to be modified during the tuning cycle. Failure to do so will give you nothing to compare the results of future modifications to.

The measurement taken during subsequent iterations of a specific tuning cycle must also be accurately recorded. The volume and type of data run through an application on subsequent tuning cycle iterations should be very close to those that were run for the baseline. If an application is in the development phase of its lifecycle, it is an easy matter to run the same test data through the application for each measurement. Test data offers the tightest control over measurement by guaranteeing that each iteration will have the same data characteristics. This ensures the measurement is not affected by variances in data, but rather by variances in the application's programs.

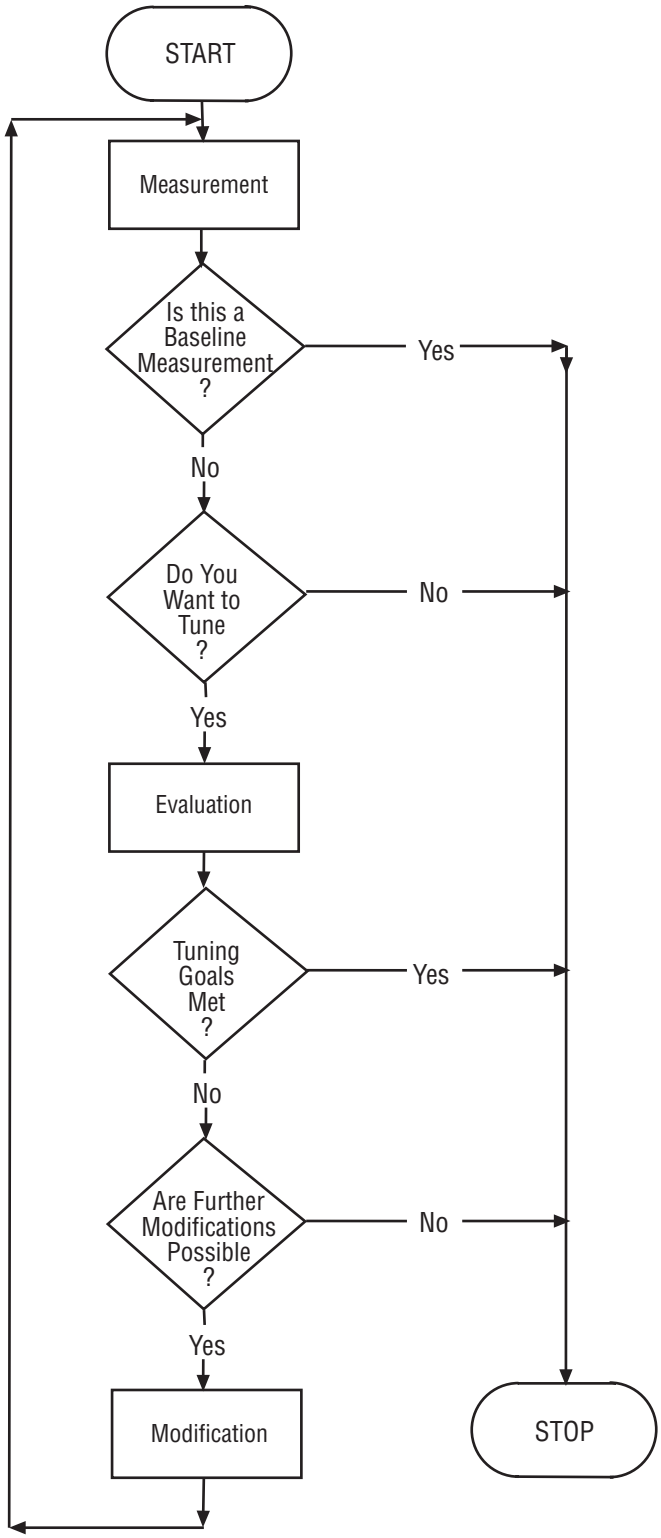

Figure 1.4: The tuning cycle.

Evaluation

Evaluation is the analytical step of the tuning cycle. In this step, the current measurement is compared against previous measurements or against expected values. The individual statistics of the two measurements are carefully compared and the differences analyzed. If you determine that your tuning goals have been met, then the tuning cycle is completed. If not, you may continue on to modification, the next step of the tuning cycle.

Evaluation is applied to both baseline and subsequent (secondary) measurements. When a baseline is being evaluated, the measurement is analyzed to see if the values appear to be reasonable. Reasonable means that values fall within an expected range and that they do not violate your organization's performance standards. If the measurement statistics are reasonable, then there is no need to continue with the tuning cycle. In this case, you should file the measurement away for future reference and continue on to other tasks.

When subsequent measurements are evaluated, the latest measurement statistics are compared against the baseline statistics or against some other premodification measurement. If a modification has brought about a reduction in system overhead, the change is deemed successful. When this happens, you must decide if there are other tuning opportunities available or if it is time to stop the tuning cycle. If system overhead increases or remains static, the modification has not been effective. When this occurs, you should discard the errant modification and evaluate other tuning possibilities.

Evaluation is the decision-making middle step of the tuning cycle. This step is the point at which the decision to continue or to stop the tuning cycle is made. If tuning goals have been met or if modifications are not reducing computer overhead, you may decide to terminate the tuning cycle. If significant resource reductions are being realized or if further tuning possibilities are available, you may decide to continue on to the next step: modification.

Modification

Modification is the action step of the tuning cycle. In this step, some aspect of the SAS application's program(s) or data is deliberately changed. The goal of the modification step is to effect a gain in the application's performance the next time it is run. Exactly *what* should be modified in this step is the main subject of this book and is covered in great detail in other chapters.

Modifications to programs or data should be made one at a time. This ensures that the full impact of a single modification can be accurately measured. If two or more changes are made and computer overhead is reduced, there is no way to tell which change was actually responsible. If a single change is made, its exact effect can be measured and evaluated. Making one modification at a time ensures there is no ambiguity about whether a specific change makes an application more efficient.

This book provides an arsenal of possible modifications that can be implemented to reduce processor overhead. It is up to you to decide which changes are candidates for the application you are tuning. You may decide to

override a SAS system option, change the block size of a data set, add an index, or make some other change. Your decision on modification candidates will be based on the characteristics of the application's data and program(s), the type of processor overhead you hope to reduce, and your own knowledge of tuning.

The modification that you think will make the greatest difference should be made first. Then rerun the application and measure and evaluate its performance. If the modification is successful, you may move on to the next change and repeat the tuning cycle. If not, you should remove the modification, implement the next candidate modification, and continue with the tuning cycle.

Tuning Cycle Example

This section presents a case study of a tuning cycle. In the case study, the simple SAS program being tuned reads a single SAS data set and creates a report.

Event One

Measurement: CPU time: 60 Secs, EXCP count: 15,000.

Evaluation: The measurement is a baseline, and the programmer decides to try to reduce CPU time and EXCP count.

Modification: An index is created for the SAS data set accessed by the program and the application is rerun.

Event Two

Measurement: CPU time: 45 Secs, EXCP count: 1,000.

Evaluation: These are good results, but the programmer would like to see a greater improvement.

Modification: The programmer uses data set compression to compress the SAS data set.

Event Three

Measurement: CPU time: 75 Secs, EXCP count: 850.

Evaluation: Although significant, the drop in EXCP count is not enough to justify the rise in CPU time.

Modification: Compression is de-implemented for the SAS data set.

Event Four

Measurement: CPU time: 45 Secs, EXCP count: 1,000

Evaluation: The programmer is satisfied with the results, so the tuning cycle is completed.

Modification: No modifications are made.

A tuning cycle is completed when no further changes will make an appreciable difference in the performance of an application. In the example, the programmer realized great efficiency with the first modification, and questionable efficiency with the second. The second change was deemed unsuccessful because CPU time is considered very valuable in the example programmer's organization. The modest reduction of EXCPs did not justify the rise in CPU time, so the second modification was deleted. No further changes were proposed, so the tuning cycle was complete.

Common sense has to be used when evaluating whether or not to run a tuning cycle. If the computer resources used by an application are already low, then it may be unnecessary to tune it. Similarly, ad hoc reports and programs that are one-time runs against modest amounts of data usually do not warrant tuning. Indeed, the time-criticality of many ad hoc reports would make it impractical to put the programs through the tuning cycle.

On the other hand, applications that use large amounts of the system resources that are in short supply in your organization make good tuning targets. You should compile a list of your applications and the critical system resources that they consume over a specified period of time, such as a week or a month. Arrange the list in descending order of the most critical system resource. (For instance, if CPU time is the most important resource, the applications would be listed in order of descending CPU time.) Now you have a prioritized list of applications that you can tune. If you are successful in your tuning efforts, then you will have freed valuable system resources for other applications to use.

Performance Goals

To effectively implement the tuning cycle, an organization needs some specific performance goals or standards. Without performance goals or standards, programmers do not have a basis from which to judge whether or not to tune an application. Or, as Lewis Carroll so aptly stated, "If you do not know where you are going, any road will lead you there."

Performance goals and standards can take on many different forms. They can be as simple as "reduce CPU time as much as possible." Conversely, they can be as complicated as a multipage publication of SAS language programming standards. Performance standards may be set by the organization, by the group, or even by the user. There may be different performance standards in effect for different groups within the same organization.

Here is an example of some possible performance standards:

- CPU time is very valuable and should be reduced as much as possible.
- WHERE clauses shall be used instead of subsetting IF statements.
- SAS data set BLKSIZE shall be set to 1/2 or 1/3 track for all SAS data sets.
- Indexes will be built and used on all large SAS data sets.

- Extracts of external files shall be stored as SAS data sets and not as flat files.

Check with your management to determine what the performance standards are for your organization. Commit your organization's performance standards to paper in a simple one-sentence-per-point list, as illustrated above. Have the list in a handy place so that you can consult it whenever you are programming. This will enable you to write and tune applications that conform to your organization's specific performance standards.

If you work in an organization whose performance standards are either nonexistent or unclear, you should create your own. Generally, this is not too hard to do. Standard methods of processing data with SAS should be examined, embraced as standards, and written down. The standards should be based on the efficiency techniques stated in SAS documentation, published in user group papers, and stated in this book. Performance standards should also reflect the performance constraints of your organization's mainframe computer(s). This effort will help to ensure that your applications are written as efficiently as possible.

When to Tune Applications

Tuning New Applications

The best time to tune an application is when it is first written. Tuning at this time usually has the minimum impact upon the end-user. During the unit testing phase of development, you should run a representative amount of data through the application and create a baseline. Then you should run the application through the tuning cycle and tune it to its greatest efficiency.

Review the SAS programs comprising a new application before it is implemented into production. During the review, SAS code that does not meet the performance coding standards of the organization should be modified. If the programmer used good programming techniques, the correct SAS system options, and effective OS/390 parameters, the application should be very efficient. A baseline measurement of the performance metrics of the application should be made for future reference.

Tuning SAS programs when they are new is a good way to build performance into applications. This allows you to move on to satisfying your users' needs for other information without having to worry about the performance of old applications. The best way to avoid performance problems is to start off with performance built into the applications.

Tuning Existing Applications

A mature organization with an embedded SAS culture is likely to have a large number of SAS applications in production. These applications may have been written by various programmers with different levels of experience. Some applications may have originally been written in early versions of SAS. They may have been converted directly to the current release without being rewritten when the organization changed releases of SAS. Some applications may have been created quickly and hurried into production with the emphasis on getting the information to the client as soon as possible.

Although time always seems to be short, it is worth the effort to re-examine existing applications and tune them. Once performance standards are in place, you have a guideline to use to tune existing applications. You can be proactive and examine your group's applications for tuning possibilities. Then you can establish benchmarks and engage in the tuning cycle. When you have completed tuning, you can report performance gains to your management as savings for the group and for the organization.

Keep in mind that some tuning possibilities do not reveal themselves when a program is first written. Things change. The quantities and types of data change, data access patterns change, sort keys change, and the number of fields in observations sometimes changes. Some tuning strategies work better with large volumes of data. Thus, a well-tuned application may need to be re-examined for tuning possibilities after it has been in production for a while.

You should examine existing applications for tuning possibilities on a cyclic basis. The more volatile the application, the more often its performance should be checked. Statistics should be kept on the main OS/390 performance metrics of all applications on a monthly, quarterly, biannual, or yearly basis. This will leave an audit trail of how the application is performing.

A good time to audit the performance of existing applications is when a new release of SAS is installed. New releases usually enrich SAS with greater functionality and with performance upgrades. Programs that were written in earlier releases and that are not audited and rewritten to take advantage of such changes and enhancements do not reap the performance benefits that may be available. You should obtain the latest *What's New in SAS…* publication and the most recent edition of the *SAS Companion for z/OS* to determine which enhancements you can exploit.

Implementing the Tuning Cycle

It is very important that you approach the tuning cycle in a methodic, structured manner. To do this, you need to make several decisions before you begin to tune. Those decisions will help you to set clear-cut goals for your tuning effort, keep your tuning cycle focused, and ensure that the results are unambiguous. The three major issues you have to decide upon before enacting the tuning cycle are

- the scope of the tuning cycle
- what to use for the baseline
- what program modifications to make.

Each of these three issues is discussed in turn below.

Determine the Scope of the Tuning Cycle

When you tune SAS applications, the first thing you need to do is to determine the scope of the tuning cycle. That is, will your tuning efforts be directed toward a specific SAS task, an entire SAS program, or a complete SAS application? Answering this question helps you to focus in on the level

of measurement and the efficiency tools that you need. It allows you to center on a specific facet of one of your SAS applications and concentrate your tuning efforts. The resources you bring to bear upon tuning at each of these levels can be as different as the levels themselves.

Task-level tuning involves implementing the tuning cycle for a specific SAS task within a SAS program. The goal of this effort is to tune the particular SAS task to its greatest efficiency. WHERE clauses, DROP and KEEP statements, and the Stored Program Facility are some of the performance tools that may be used at the task level. When you are tuning an individual task, it is important that SAS processing statistics be turned on at the task level. This facilitates the measurement of the tuning changes made to the particular task. The SAS processing statistics, written to the SAS log, can help you to determine the success of your tuning decisions.

Program-level tuning involves implementing the tuning cycle for all of the SAS tasks within a SAS program. The purpose of program-level tuning is to reduce the computer overhead of the *entire* program. In its simplest implementation, this may involve the tuning of two or more existing SAS tasks. In its most complex form, it may involve the addition and deletion of SAS tasks, the restructuring of SAS data sets, or, possibly, a complete rewrite of the entire program. When you are tuning at the program level, processing statistics in either the SAS log or the OS/390 job log may be used to measure performance. If the SAS log is used, processing statistics such as CPU time and EXCP count for all of the SAS tasks within the program must be combined. The aggregate of the individual SAS tasks' processing statistics provides the true measure of the program's overall efficiency.

Application-level tuning entails implementing the tuning cycle for two or more of the programs that comprise the application. The objective of application-level tuning is to reduce the overall computer overhead of the *complete* application. This may involve tuning individual tasks in specific programs, writing new programs, rewriting old programs, or restructuring SAS data sets. Tuning at this more global level requires that SAS or OS/390 processing statistics be recorded at the individual program level. The aggregate of the program-level measurements yields the application's processing statistics totals. These totals can be used to measure the success of your efforts to tune the entire application.

Determine What to Use for the Baseline

Once you have decided upon the scope of the tuning cycle, you must determine if baseline measurements currently exist. Baseline measurements are the processing statistics recorded at the beginning of a tuning cycle, before any program changes are made. The baseline acts as a "before" measurement, against which future, "after" measurements are compared. The baseline metrics you need are normally recorded in the SAS log or the OS/390 job log. (Refer to Chapters 2 and 3 for important SAS metrics and where to find them.) It is vital to have an accurate baseline so that you can determine the merits of your tuning endeavors.

Baseline measurements must be available for the level at which you intend to tune. For instance, if you are tuning a SAS task, you need a baseline for that task. If you are tuning a program, you need a baseline either for the entire

program, or for all of the tasks that compose the program. For an application, you need a baseline for either the application, all of the application's programs, or all of the application's tasks.

If no baseline exists, then you must act to have one created before you begin to tune. Enable the SAS processing statistics for the next run of the SAS task, program, or application that you intend to tune. Modifying SAS code for efficiency's sake without a baseline measurement cannot seriously be called tuning. Without a baseline, you cannot prove that any real change has taken place, either positively or negatively in your SAS task, program, or application. Consequently, no tuning can begin until you have secured a valid baseline measurement.

Perhaps you have already enabled the SAS processing statistics in all of your SAS applications. Doing this ensures that the information you need for tuning is always available. This eliminates the necessity of having to overtly enable the SAS processing statistics and wait for the next run of a particular program. With the SAS processing statistics enabled, you can simply review the SAS log of the latest program execution to establish your baseline.

Remember that the baseline should be current and should contain the same volume of input data as will be run through the modified program. This is necessary to eliminate false tuning results that can occur because of disparate data volumes. Obviously, larger volumes of data require more processing overhead. A baseline taken for a program processing millions of observations is not adequate if subsequent runs of the program process a few thousand observations. The ideal situation is to make the baseline and subsequent program runs using the same data. This can be controlled in a testing or development environment, but is often difficult in a production system.

Determine What Program Modifications to Make

The last thing you need to decide on, before beginning the tuning cycle, is exactly what modifications you intend to make to your SAS program. List the modifications on paper. Order them in descending order of which ones you expect will effect the biggest performance gains. This is the order in which you should actually make the modifications. By listing them in this order you may shorten your tuning cycle process. If the first or second modification allows your application to reach its tuning goals, then there is no essential requirement for continuing to tune. So, listing your modifications in order of importance may actually save you time during the tuning cycle.

Each modification on your list should be made separately, and evaluated, before the next change is made. In this way, the outcome of the change can be fully evaluated on its own merit. This cannot be stressed strongly enough! If several performance-oriented modifications are made at the same time to the same task, the results of each individual change are obscured. You cannot determine exactly which one effected the change in program efficiency. Perhaps you can conjecture that one change "must have" had a more profound impact than another. But that is just an educated guess. Without serializing the modifications in an incremental fashion, you can never really verify exactly which change made the difference in performance.

Run the Tuning Cycle

Once you have prepared for the tuning cycle by making the decisions above, it is fairly easy to enact it. Simply follow the three basic steps of the tuning cycle as outlined in the earlier sections of this chapter. The process is straightforward, and you will probably find it fairly simple after you have done it a couple of times.

While tuning, it is very important that you keep track of all of your modifications and the results of having made them. To help you to do this, some worksheets have been included at the end of this chapter. There are separate worksheets for tuning SAS tasks, SAS programs, and SAS applications. Each worksheet can be used to record modifications and the measurements made before and after modifications. You may decide to save them for future reference.

Report Performance Gains

One of the best aspects of tuning your SAS applications is reporting the performance gains to management. To accomplish this, you need to contrast the baseline measurement with the measurement taken after you have enacted the program efficiency changes. The difference between the two metrics, at the SAS task, program, or application level, is the amount of computer resources you have saved your organization.

In reporting performance gains, you should characterize them in the best terms that management understands. If management is cognizant of terms such as EXCP count and CPU time, you can use them in your report. If they only understand data processing expenses, and your organization has a computer chargeback system, use dollars and cents to report your savings. For example, consider how the resource savings in the tuning cycle example given earlier in this chapter could be presented:

- Savings of 15 CPU seconds and 14,000 EXCPs

or

- CPU time reduction of 25% and an EXCP count reduction of 93%

or

- CPU time cost reduction of $7.50 and an EXCP count cost reduction of $14.00. (Based on a CPU time charge of $.50 per CPU second and an EXCP count charge of $1.00 per 1,000 EXCPs.)

If the performance savings from the modifications seem paltry, try characterizing them over time. Perhaps when they are considered as weekly, monthly, or yearly savings, they will appear more substantial. Portraying them in such a way is not being deceptive. They *actually* are real processor savings that your organization will enjoy. Management should be aware that you have made changes that will save computer resources. By characterizing them over time, you allow management to see their true long-term value. An example of portraying the above savings for a weekly job is

- The program tuning has resulted in a monthly savings of $86. This amount includes a $30 reduction in CPU time charges and a $56 reduction in EXCP count charges.

Figure 1.5 illustrates a memo written to describe the savings of having tuned the SAS program in the tuning cycle example. In this instance, the program ran once a week in batch job RELICS01. This environment had a computer chargeback system that levied $.50 per CPU second and $1.00 per 1,000 EXCPs. The memo provides enough information for the department head to understand the significance of the tuning effort.

```
To:         Dr. Arnold Layne
From:       Michael A. Raithel
Date:       April 27, 2005
Subject:    Results of Tuning the Benefits System

This memo was written to notify you of the performance
savings that were realized by tuning the job RELICS01 of the
Benefits System. By implementing and using an index to
access the main SAS data set, program CPU Time was
reduced by 25% and the EXCP Count was reduced by 93%.
This translates to a reduction in CPU charges of $7.50 and
EXCP charges of $14.00, per run. The total savings per run
is $21.50. Since the job is run weekly, we can expect to
save $93.17 per month, and $1,118.00 per year.

We are always looking for ways to improve the efficiency of
the Benefits System. We will keep you informed of future
tuning efforts.
```

Figure 1.5: Example of a memo to management characterizing the results of tuning a SAS program.

Since your tuning endeavors, when successful, have value to your organization, you should not hesitate to publicize them. Be sure to write a memo or an e-mail message that distinguishes the computer resource savings you have achieved. Both your direct management and your end users should be aware that you are doing the most to make their applications as efficient as possible. This gives them the full picture of your accomplishments and enhances your reputation as a top SAS programmer.

Summary

Given the importance to your organization of efficient applications, it is imperative that you create SAS programs that have good performance. To accomplish this requires a certain shift in your focus. Looking at program performance directs your attention away from the traditional end product of a SAS program. Program efficiency is not concerned with a particular report, graph, or chart. Rather, it is concerned with the SAS program processes that *produce* the report, graph, or chart. The introspective reviewing of SAS

program processes is central to creating efficient programs. By focusing on how you process data and arrive at your output, you can create SAS programs that perform at their optimum level.

The tuning cycle is a methodology that you can utilize to tune the performance of your SAS applications in a controlled, step-wise manner. The tuning cycle is composed of three basic steps:

- measurement
- evaluation
- modification.

When you implement the steps of the tuning cycle and make incremental changes to your SAS programs, you can determine which tuning decisions really do reduce computer overhead. You can implement the tuning cycle at the SAS task, program, or application level. By doing so, you will guarantee that your applications are running as efficiently as possible on your mainframe computer.

SAS Task Tuning Worksheet

Application Name: ______________________________

Program Name: ______________________________

Task Identification: ______________________________

Programmer Name: ____________________

Date: ____________________

Modification Number: _______			
	Baseline Measurement	**Modification Measurement**	**Difference Between Baseline & Modification**
CPU Time:			
EXCP Count:			
Task Memory:			
Total Memory:			
What Was Modified:			

Modification Number: _______			
	Baseline Measurement	**Modification Measurement**	**Difference Between Baseline & Modification**
CPU Time:			
EXCP Count:			
Task Memory:			
Total Memory:			
What Was Modified:			

Modification Number: _______			
	Baseline Measurement	**Modification Measurement**	**Difference Between Baseline & Modification**
CPU Time:			
EXCP Count:			
Task Memory:			
Total Memory:			
What Was Modified:			

Page ___ of ___

SAS Program Tuning Worksheet

Application Name: ________________________

Program Name: ________________________

Programmer Name: ________________

Date: ________________

Modification Number: _______			
	Baseline Measurement	**Modification Measurement**	**Difference Between Baseline & Modification**
CPU Time:			
EXCP Count:			
What Was Modified:			

Modification Number: _______			
	Baseline Measurement	**Modification Measurement**	**Difference Between Baseline & Modification**
CPU Time:			
EXCP Count:			
What Was Modified:			

Modification Number: _______			
	Baseline Measurement	**Modification Measurement**	**Difference Between Baseline & Modification**
CPU Time:			
EXCP Count:			
What Was Modified:			

Page ___ of ___

SAS Application Tuning Worksheet

Application Name: ______________________________

Programmer Name: ____________________

Date: ____________________

Modification Number: _______			
	Baseline Measurement	**Modification Measurement**	**Difference Between Baseline & Modification**
CPU Time:			
EXCP Count:			
What Was Modified:			

Modification Number: _______			
	Baseline Measurement	**Modification Measurement**	**Difference Between Baseline & Modification**
CPU Time:			
EXCP Count:			
What Was Modified:			

Modification Number: _______			
	Baseline Measurement	**Modification Measurement**	**Difference Between Baseline & Modification**
CPU Time:			
EXCP Count:			
What Was Modified:			

Page ___ of ___