Since each Apollo manned space flight makes new demands on the computer configuration and operating system, data processing efficiency is tested before each flight by simulation described in this paper.

Discussed is the dynamic gathering of operating system performance data during real-time simulation, achieved by incorporating appropriate routines in the Apollo control program. The data thus collected is used as input to improved system models.

The effect of the statistics gathering routine on systems performance can be measured.

Statistics gathering and simulation for the Apollo real-time operating system

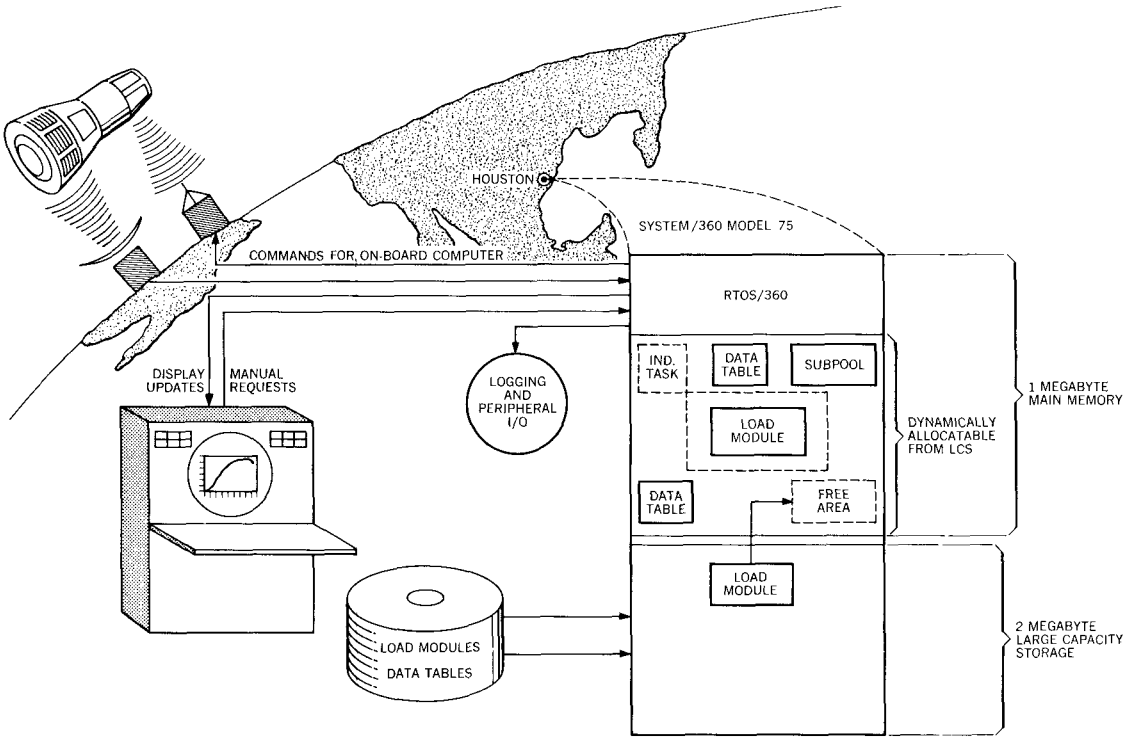
by W. I. Stanley and H. F. Hertel

In monitoring Apollo manned space flights, National Aeronautics and Space Administration flight controllers must rely on a complex real-time computing system. Trajectory and telemetry data, providing both positional and environmental parameters of the space vehicle and the astronauts, must be processed immediately upon arrival at the Real Time Computer Complex (RTCC) via a worldwide telecommunication network. Computers, peripheral equipment, and programming support, as shown in Figure 1, must be efficient enough to avoid a backlog.

In addition to their primary purpose of supporting manned space flights, RTCC computers support real-time space-flight simulations. These simulations generate trajectory and telemetry data to test the mission support system and to train flight controllers in an environment that closely approximates an actual mission.

Each Apollo mission presents the RTCC with a unique set of processing requirements. For example, real-time data sources may change in number, arrival rate, or message size. These and other such factors cause changes in the performance of real-time computing systems. So that changes do not cause the systems to perform below acceptable limits, performance of current systems is measured and that of future systems is modeled. Performance measuring as well as design modeling and simulation are the main subjects of this paper. These techniques are a formal method of

Figure 1 Real Time Computer Complex programming system



assuring that RTCC performance requirements are met in advance of computer system delivery. This paper introduces processing requirements and the operating system briefly. Later, performance measuring and modeling techniques are discussed as they are used in studying the design of future systems.

processing requirements

To date, the most important Apollo mission areas requiring modeling are the launch and orbit phases. The launch phase consists of two periods: powered flight and a brief period of computation, called "hold." Powered flight processing consists primarily of two asynchronous processing cycles (trajectory and telemetry) competing for all system resources. During hold, go/no-go and abort computations operate as background to the higher-priority trajectory and telemetry cycles. It is imperative that the response of the hold cycle be within certain allowable limits. At the same time, the responses of the two faster cycles, trajectory and telemetry, must not be seriously impacted. Because of the load of cyclic processing, the critical system resource during launch is the central processing unit (CPU). Launch-phase modeling is primarily concerned with the evaluation of the ability of the computer to maintain all cyclic responses.

In the orbit phase, telemetry and trajectory processing are done only when the spacecraft is in range of an adequately equipped

station. Even then, the processing magnitude of orbit phase is much smaller than that of the launch phase. However, other tasks are normally performed during orbit: mission planning (rendezvous computations and orbital changes), updating trajectory predictions, and calculating time-to-fire. In general, these tasks do not have severe response requirements and operate as background to orbit cyclic processing. Thus, they do not necessarily impose a heavy load on the system CPU. However, the program and data sizes associated with these tasks are quite large. Because a number of these tasks can be in process at any one time, such tasks can make serious storage demands. Therefore, just as launch-phase modeling is centered on the evaluation of CPU capabilities, orbit-phase modeling primarily studies main storage and large-capacity storage availability.

Support for the Apollo space flights consists of five SYSTEM/360 Model 75 computers and a modification of the SYSTEM/360 Operating System (OS/360). The decision to use OS/360 as a real-time operating system to support the stringent response requirements (real-time data messages arrive at rates of 50 messages per second) was not without problems. First, the performance of OS/360 had been essentially untried in an application of this magnitude. Second, RTCC controls the computer time sharing and peripheral I/O devices among many semi-independent jobs that operate asynchronously, but share or reference some of the same data. Third, the first system had to be operational by the last quarter of 1966, which left little leeway for misjudgment of performance or miscalculation in design.

operating system

The following introduces the RTCC real-time operating system (RTOS/360) and briefly describes how statistical and simulation methods are used to analyze its design and performance.

RTOS/360 is basically OS/360 (multiprogramming with a variable number of tasks) with modifications to the nucleus to increase efficiency in the RTOS environment and to provide additional real-time capabilities. Part of the deviations from OS/360 result from the need to effectively use a one-megabyte SYSTEM/360 Model 75 with an additional two-megabyte Large Capacity Storage (LCS). Other modifications are: supervisor services that interface with specialized methods of data management, a modified definition for task management, and new receiving and routing logic for incoming telecommunication data. The following items suggest the scope of RTCC changes to existing OS/360 routines that significantly improve system performance when executing an RTCC real-time job step. Program fetch was modified to use LCS as a residence for load modules and real-time data tables. The capabilities of loading LCS from a disk and loading main storage from LCS were added. The main storage supervisor (MSS) was modified to use LCS as an extension of main storage. The main storage supervisor was modified to include storage allocation algorithms tailored to the real-time environment. Temporary storage for system control tables is obtained from fixed-size pools instead of from MSS.

Some design changes were necessary to allow several independent units of work to share computing resources. Thus, the concept of "independent task" was introduced. An independent task is a sub-job step that can share programs and data with other tasks and can be referenced by a symbolic task name, which allows a task to have a queue of work that it performs serially. An independent task has a dormant state that allows the task to remain known to the system although it has no current work. The dormant state permits cyclic processing since independent task resources are saved until the next data frame or processing cycle when the task becomes active again. Independent tasks may be created and destroyed without affecting the operation of other independent tasks.

Data tables were added to the data management services offered by OS/360. Data tables are referenced symbolically with a system name in much the same way as a load module. Data table services do not require OPEN or CLOSE macroinstructions to be executed. Data table macroinstructions are not access mechanisms to I/O devices; rather, they reference the named data or storage, which may be allocated in LCS or in main storage.

Control program services were added to control the RTOS/360 logic that receives and routes real-time input data arriving over the telecommunication lines. This logic provides RTOS/360 with an efficient method of identifying input data by using parameters supplied by the application programs. Data is buffered and routed to a work queue of the proper independent task. These and other control program services modify OS/360 to form RTOS/360, which supports the RTCC real-time application systems created for the Apollo project.

In deciding which design alterations and which implementation changes yield the best performance, two evaluation techniques are used. A Statistics Gathering System (SGS) obtains timing and frequency statistics on RTOS/360 control program services as well as all application programs. The General Purpose Simulation System (GPSS)[1,2] predicts expected performance (given the SGS statistics) of both real-time job steps and job-shop operations.

In an initial simulation, actual performance statistics are generally not available.[3] However, initial simulations can and should be performed during the systems design stage to evaluate the machine and basic program design together. Performance statistics for models used in the initial simulation are based on performance statistics of similar systems. As the programming component of the computer system is developed, actual performance statistics are gathered and added as refinements to the model. Thus, as the computer system is being developed, the model provides a more accurate evaluation of the programming design.

Measuring system performance

To measure the performance of a real-time system and monitor its execution, a comprehensive Statistics Gathering System (SGS)

was developed. SGS, a program and not a hardware device attached to the computer, provides an accurate means of measuring performance on RTOS/360 by collecting:

- Timing information on control program services and application programs
- Percentage figures showing how definable system functions use the CPU resource
- Elapsed-time figures showing task response time in a multiprogramming environment

An interface with RTOS/360 enables SGS to record time, logic flow, and frequency statistics. Although SGS degrades performance, it does not push the processing load to the point of failure. SGS relies on the fact that a real-time system normally operates with enough idle time or surplus capacity to allow the system to handle peak load processing surges. (While statistics are being collected, some surplus capacity is used up; thus, the time to process a peak load is lengthened.) In the normal job-shop environment, where SGS monitors the FORTRAN compiler, assembler, or execution job step, the monitored job step takes longer to complete. In both the real-time and job-shop environments, SGS gives the percentage of computer capacity used by itself in collecting and reporting statistics. This feature allows the analyst to remove the effect of SGS from statistics on the actual system.

statistical categories

The SGS design for RTOS/360 is patterned after an earlier version used with the Gemini 7094 executive control program. Experience shows that dividing the obtainable statistics into several independently selected categories reduces the impact on normal performance of the system. Six general categories of statistics have been defined for SGS:

RTOS/360 *statistics* provide execution times and frequencies for control program functions, i.e., the control program services and other control program routines.

Load module statistics provide execution times and frequencies for each load module and show the number, type, and CPU utilization for control program services used by each load module.

Gross CPU *utilization statistics* provide the percent utilization of the CPU by RTOS/360 and by the application system, the percentage of capacity spent waiting for I/O operations, and the percentage of capacity spent idle, i.e., time when no work is in process and none is queued waiting to be processed.

Independent-task statistics provide frequencies with which named tasks are executed, average response times for the tasks, and computer capacity used by them to perform assigned work.

I/O *device statistics* provide frequencies with which I/O devices are used.

Logic traces provide the logic flow and CPU execution time of the logic required to satisfy an application program request for a control program service.

SGS accounts for all CPU time. Symbolic clocks are kept on each program segment identified to SGS. Time not spent executing programs is tallied either as time spent waiting for I/O or as idle time. A notable characteristic of SGS timing statistics is that the measured execution time of the instruction logic of each system function is independent of other system functions. Elapsed times for I/O operations and execution of other functions are accrued separately. The statistics obtained are more nearly independent of the environment in which they are obtained. Therefore, timing statistics may be used in a model of a real-time system that simulates a different operating environment from the one in which the statistics were obtained.

Statistics gathering is divided into three phases: initialization, collection, and reporting. A user may select one or more of the six categories of statistics by entering control parameters to start and stop the selected categories according to time or the initiation and termination of a job step. These control parameters, which are entered via the job stream or an on-line typewriter, are passed to the SGS independent task to start the initialization phase. All SGS logic is implemented as transient load modules to minimize the impact of size on storage allocation for an operational system.

initialization

The first phase initializes the RTOS/360 nucleus enabling control to pass to SGS collection routines whenever the CPU begins executing instructions for a new or different system function. The definition of system function used by SGS is the change of processing purpose implied by CPU interruptions. Since each purpose requires its own implementation, the implementation logic and purpose define a system function. The interface with RTOS/360 is via the new program status words (PSW), which pass control to SGS upon each interruption of the CPU.

There are five types of interruptions that cause a change in the PSW:

- I/O *interruption* — indicates the need to service an I/O channel
- *Program interruption* — indicates a program error or exceptional result during execution
- SVC *interruption* — indicates a request for a control program service
- *External interruption* — signals the computer to service an external device
- *Machine check interruption* — indicates an error in computer hardware

Each type of interruption has an assigned new PSW and an old PSW in main storage. Upon interruption, the old PSW is saved and the new PSW is loaded with the new contents of the instruction counter to give the starting address of an RTOS/360 interruption

handler. When SGS is in use, the new PSW sets the instruction counter to the start of the SGS program that handles the particular type of interruption. After recording the statistics required, the SGS program passes control to the RTOS/360 interruption handler with all machine conditions set as though the interruption had proceeded there directly.

An interruption is not associated with every change in system function, and, therefore, interruptions alone cannot define the complete SGS interface. For instance, a load PSW (LPSW) instruction, which returns control of the CPU to an application program after completing a control program service, does not interrupt the CPU. Nevertheless, there is a definite change in functional purpose of the instructions being executed. The method of keeping SGS informed in cases like this is to force an interruption. Combined with the nucleus by the linkage editor is a small table of symbolic references to such special instructions as all LPSW's, entry to program fetch, and entry to the dispatcher. At SGS initialization time, the operation code of these instructions is replaced with an illegal operation code. Upon each execution thereafter, a program interruption occurs at these identifiable addresses.

In the case of certain control program subroutines, SGS not only must recognize when the program is entered, but also must recognize when the program is exited. It is possible to take advantage of the fact that general register 14 (GR 14) is used as a standard return register. For example, assume program A wants to call subroutine B using the SYSTEM/360 assembly language instructions

```
L     15, = V(B)
BALR 14, 15
```

At address B, SGS gains control by means of an illegal operation code at the subroutine entry point. SGS then saves the contents of base register 14 and replaces the contents with an SGS address. Also, statistics gathering is stopped for A and started for B. Subroutine B is executed and then returns control via the address in base register 14 (BR 14). SGS gains control via the address in base register 14, then stops statistics on B, restarts statistics on A, restores base register 14, and branches to the intended return point in program A.

During statistics initialization, SGS routines that obtain the selected categories of statistics are loaded into main storage. Until collection is terminated, these routines are entered each time there is a change in system function.

collection

The second phase in SGS consists of routines that record execution time, frequency counts, and logic flow. Statistics are collected in main-storage buffers that are linked to the proper job step, independent task, and load module. Note that most categories of SGS statistics do not report every statistical event, rather statistics are averaged over a period of time and the average is reported for each system function.

Storage for statistics is obtained, as necessary, from the main storage supervisor. The size of SGS in main storage depends on the number and kind of statistical categories selected, and on the number of system functions in the selected categories for the application system being monitored.

The statistics for each system function are identified by a symbol. Normally, these symbols are also used externally to SGS, such as: macroinstruction names for statistics on RTOS/360 control program services, load module names for application program statistics, and independent task names for statistics reported for each independent task and any of its dependent tasks.

Multiprogramming is a basic ingredient in the SGS method of maintaining statistics. Each time RTOS/360 performs a task switch, SGS interrupts the collection of statistics for the current task and starts collecting statistics for the new task. Each time RTOS/360 resumes a previous task, SGS resumes collecting statistics for that task. Any partially completed system function, which is temporarily interrupted because of multiprogramming considerations, is recognized by SGS, and a partial set of statistics is saved until the function is resumed at a later time.

To record accurate timing statistics, a clock with a ten-microsecond resolution is used. Each time a program or routine is started, stopped, or interrupted, a logical clock with the appropriate symbolic identification is updated accordingly. A typical sequence of SGS is as follows:

1. Time is recorded, and the current function stops.
2. Contents of all necessary registers are saved.
3. Statistics for the interrupted routine are updated.
4. Overlaid instructions are simulated.
5. If the interrupt is a supervisor call (SVC), a symbolic clock is started, according to a control code in the SVC instruction.
6. If the interruption is caused by illegal instructions resulting from the augmented SGS interface, a symbolic clock is started, according to a control code in an SGS table.
7. Control returns to the normal RTOS/360 logic flow.

reporting

The third phase of SGS is that of reporting, which may be called periodically to allow a requestor to record trends or changes in system performance. Programs associated with generating SGS reports operate as transient load modules. The reporting phase runs as an independent task whose priority is adjusted to allow it to compete favorably for system resources. Since the reporting phase cannot stop the system to report the statistics collected, the reporting and collecting phases alternate. When a complete report is generated (normally in less than one second), the time and frequency counts are reset to zero, and the collection phase begins again. Formatted reports are usually written on tape for off-line printing.

Samples of the first statistics produced by the SYSTEM/360 version of SGS are shown in Table 1 and are intended to provide

Table 1 Execution times for logical functions

RTOS/360 *logical function*	*Average time (microseconds)*	*Number of executions timed*	*Description of function*
LINK*	1693	3	Passes control temporarily to a load module
XCTL*	1760	1	Passes control to a load module
LOAD*	2026	6	Requests a module be loaded and retained in main storage
EXIT*	836	45	Passes control from current load module
Dispatcher	498	123	Passes control to tasks according to priority
GETMAIN*	1186	58	Allocates main storage for use by requesting program
REGMAIN (get mode)	2427	35	Allocates main storage in supervisor subpools
FREEMAIN*	1133	3	Returns program-allocated main storage to a free pool
REGMAIN (free mode)	1253	67	Frees main storage from supervisor subpools
WAIT*	712	30	Waits for a specified number of events to occur
POST*	804	10	Sets a *complete* flag when an event has occurred
POST (branch entry point)	142	38	Sets a *complete* flag for I/O supervisor
BLDL*	700	2	Builds a special directory in main storage
Program fetch	1760	2	Reads a load module into main storage
Program fetch after I/O interruption	604	18	Reads a load module into main storage after I/O interruption
EXCP*	1872	38	Requests transmission to/from an I/O device
Input/output first-level interruption handler	347	49	Retains register contents of interrupted program
Interruption supervisor	1053	49	Performs I/O device control
External first-level interruption handler	650	4	Retains register contents of interrupted program and determines cause of interruption
Time routing	232	5	Creates a work request based on time
Routing	160	5	Creates a work request based on time or data
RTATTACH*	2600	1	Enters a work request in task queue
OPEN*	3280	1	Prepares system for data transfer
CLOSE*	2760	1	Restores system after data transfer
DELETE*	1408	5	Makes a load module eligible for purging
STIMER*	1176	5	Provides time control services
RTIME (243)*	460	2	Acquires Greenwich Mean Time
RTIME (250)*	960	1	Sets system to Greenwich Mean Time
DTROUTE*	1760	1	Creates a control block for routing
RUATTACH	840	2	Enters a work request in task queue (used by routing)
DTWRITE*	1190	12	Writes a data table from user load module
DTREAD*	1140	2	Reads a data table for user load module
DTLOAD*	1410	12	Requests a data table to be allocated to main storage
DTDELETE*	1040	1	Makes a data table eligible for purging
RTWRITE*	660	4	Transmits telecommunications data
Logging	1120	4	Records telecommunications data on a log tape

* RTOS/360 macroinstructions.
Note: Average times for the RTOS/360 functions may vary depending upon the application system from which they are obtained and the level of SGS development.

the reader with an insight into the level of detail that SGS yields. These statistics were obtained while running test cases on a SYSTEM/360 Model 50. The average time reported for each RTOS/360 system function is the time to execute the instructions for that particular function only. The average times are the basic statistics with which system performance is analyzed at RTCC.

It should be understood that the average times given in Table 1 reflect only the time required to execute the basic routine. Providing the whole control program service may, in fact, require the execution of additional system functions. For example, the total time for the complex control service LINK, given in Table 1, can be calculated by including all additional logical functions, such as REGMAIN, to the basic LINK execution time.

System functions marked with an asterisk in Table 1 are derived from RTOS/360 control program macroinstructions and are thus directly available to the application programmer. Functions without the asterisk are used only by the RTOS/360 control program and are unavailable for direct use by the application programmer.

Simulating system performance

The Real Time Computer Complex is not a project that is blessed with a firm definition of mission requirements. Results of each mission impose requirements for future missions and, thus, levy new demands for real-time support. It is essential to the orderly development of RTCC real-time systems to anticipate problems in computer system configuration or system program design that could impair the success of future missions. To analyze future system performance, RTCC uses models written in the language of the General Purpose Simulation System (GPSS/360).[4]

The primary responsibility of the RTCC modeling effort is system assurance. Models of particular missions are primarily developed to assure the RTCC project that both the machine configuration and the system programs to be used will perform satisfactorily. Thus, the most frequent output of modeling is a prediction of systems performance for a particular mission. This prediction can be expressed in such ways as CPU load, cyclic response, and channel utilization.

GPSS models designed for the RTCC are composed of four major components:

operating system model

1. The SYSTEM/360 computers and many of the peripheral I/O devices are modeled. This component defines the CPU, main storage, and I/O devices in terms of parameters that allow speed and size characteristics to be changed in order to model other computer configurations.
2. The RTOS/360 nucleus component simulates significant RTOS/360 and OS/360 control-program services. These services are modeled as subroutines so that new designs for operating-system logic may be tested.

3. Application systems are modeled in a manner analogous to programming the real-time system. Models of application-system programs are combined with the computer and the RTOS/360 logic components to simulate a total real-time system.
4. The world-wide telecommunication network model represents message size, arrival rate, and transmission-line speed of messages arriving at RTCC during an Apollo mission.

Of these four components, the SYSTEM/360 and RTOS/360 nucleus models are relatively constant. Therefore, these models are subroutines in the GPSS/360 modeled operating system (GMOS). By using GMOS models of control program service routines, the analyst need only characterize the application programs and the telecommunication data. GMOS includes current timing statistics and logic flow of the RTOS control program services. Therefore, the analyst who is studying a new application has a significant portion of his system accurately modeled.

GMOS provides users with an easy interface to hardware models and models of control program service routines. A control program service is requested by the following format wherein TRANSFER and ASSIGN are GPSS macroinstructions:

```
TRANSFER    SBR, SVC, 12
ASSIGN      4, (service)
ASSIGN      4, (argument 1)
ASSIGN      4, (argument 2)
ASSIGN      4, (argument 3)
ASSIGN      4, (argument 4)
```

More specifically, a request for a control program service (such as EXCP) that simulates a reference to an I/O device (such as TAPE) is written as follows:

```
TRANSFER    SBR, SVC, 12
ASSIGN      4, EXCP         Service being simulated
ASSIGN      4, TAPE         I/O device
ASSIGN      4, 40           Number of bytes being transmitted
ASSIGN      4, ECB          Event control block
```

When this sequence of instructions is executed by an application model, a TRANSFER is made to the GMOS logic that simulates an SVC interruption. CPU time is simulated for the instructions executed in the first level interruption handler and the execute channel program (EXCP) logic. The RTOS/360 model then simulates a START I/O instruction, which initiates a model of a tape device. While simulating the time to transmit 40 bytes of data to or from tape, the CPU model passes control back to the application model (after the CPU time for exit logic). When the data-transfer time has elapsed, the CPU model is interrupted, and control passes to a model of the I/O interruption handler and to a model of the I/O supervisor. After the proper amount of CPU time is simulated and the event control block (ECB) is posted, GMOS logic proceeds to a model of the dispatcher and then, again, to the application model. A simple interface for the application modeler results in a com-

plex web of logic in much the same way that a simple control program service initiates similar logic in RTOS/360 or OS/360.

application models

One of the most important parts of computer systems analysis is a good definition of the application program logic and processing requirements. Often, obtaining this information is not a trivial task.

The modeler is concerned with expressing the system logic and processing requirements in terms of GMOS standard interfaces. At RTCC, the modeler defines input characteristics (frequency, message size, routing procedure), obtained largely from NASA mission requirements. He then models the input in a series of predefined calls to GMOS input routines. The modeled input serves as a driver (i.e., initiates all processing) for the mission model just as the real input drives the system being modeled.

Simulation involves the concept of mission logic, which refers to the logic within the many load modules that make up an Apollo mission plus a general description of all load modules and data tables used. The analyst gathers this information from mission programmers and then expresses the logic or refers to modeled data tables in a series of calls to GMOS routines. Each of these calls corresponds to a similar call in RTOS/360 and, in general, carries with it a similar set of arguments. The only exception to this generality is a special call representing the expenditure of enabled processing time within a load module. The load-module and data-table descriptions are expressed as parts of a GPSS/360 FUNCTION. As an example, the following is a typical description of a load module on a GPSS function follower card:

```
LMID    2000    626    1    BEG    8000
```

Here, LMID is the predefined identification of a 2000-word-long load module resident in Large Capacity Storage (LCS). That the load module is serially reuseable is indicated by the GMOS code "1." The number 626 points to a location that identifies LCS. The load module has an entry point at BEG (defined in the model of the load module) and executes for 8000 time units. Sometimes, if the analyst is working in advance of actual implementation, this information is largely guesswork. If the load module is already in operation, however, the gathering of these characteristics is simple: SGS measures execution time, and all other items are available from the programmer.

Occasionally, in preparing model-based predictions of system performance, it is found that the defined system is not capable of meeting mission requirements. In these cases, a model becomes a valuable tool for use in defining and evaluating possible solutions to the problem.

A model of the GT-6 orbit phase (the first planned-rendezvous mission in the Gemini series) disclosed that the system would be unable to service program queues as fast as they were generated. The model was used to define two solutions to this problem: a more efficient program-priority arrangement, and an increase in the size of main storage. Also, the RTCC conversion from an IBM 7094-based

system to SYSTEM/360 was largely brought about by model predictions that the advanced Apollo requirements would far exceed the 7094 capacity. Both of these efforts utilized a GPSS model of the executive control program,[5] the 7094 equivalent of RTOS/360. GMOS is expected to be used in much the same manner to evaluate SYSTEM/360 performance. It is now being used in the prediction of early Apollo 500 series mission performance and Apollo storage requirements.

The very existence of a model has many times proved valuable in testing specific hardware or system program design alternatives (often on short notice). For example, an existing model of the Apollo simulation system for mission test and training (SCATS) was useful in evaluating several alternative main-storage purge algorithms. For such a study, the model is modified to run with each algorithm, and the most efficient algorithm is identified by a comparison of results. The same procedure can be followed to evaluate design alternatives in such areas as: system bulk storage devices, main storage allocation algorithms, data tables versus subpools, program linkages, and routing algorithms.

model statistics

In creating a model such as GMOS to simulate multiprogramming systems as complex as those at RTCC, it is necessary to model logic and to represent timing statistics with a reasonably high degree of accuracy. The logic modeled in GMOS represents all the unique services provided by RTOS/360 in the normal execution of a real-time job step. Decisions to use these services are based on the same parameters that would influence processing for an actual system. For example, if the LINK macroinstruction is executed, the logic modeled for the LINK routine executes a GETMAIN for an SVRB (supervisor request block) similarly to the way that the actual RTOS/360 does. If the requested load module is not in main storage, the GMOS model enters logic to simulate the program fetch. If program fetch simulates a GETMAIN for a transient area in which to place the load module and if the model of the main storage supervisor cannot find an area large enough, the purge routine removes unused load modules from a simulated main storage. Performance of each significant system function is simulated according to the same parameters and reacts to the same conditions that the RTOS/360 would if it were operating in the defined computer system.

SGS provides the timing statistics necessary to accurately assess the computer capacity required to execute each system function. Each function (GETMAIN, LINK, program fetch, etc.) uses CPU time according to the average execution-time statistics obtained by timing the actual system program with SGS. The statistics provided by GMOS show the analyst:

- Elapsed time to perform an independent task
- Percentage utilization of I/O devices and the number of accesses
- Number of purges necessary

- Number of times load modules and data tables are allocated to main storage
- Number and kind of control program services requested
- Percentage of CPU time used for: RTOS/360, application program, waiting for I/O, and idle time
- Detailed CPU utilization statistics of RTOS/360 functions

With knowledge of how system capacity is being used, the analyst can spot performance problems, i.e., services for which too great a price is being paid for the work being accomplished. The performance of new logic design can be tested by modeling the new ideas, replacing the model of existing logic, and rerunning the total system model.

model analysis

Once a model is working successfully, a great amount of information is available for analysis. For models of the Apollo launch phase, the analyst is interested in CPU utilization and its breakdown into such component parts as mission time and times for I/O supervisor, task management, and storage management. A simple insertion by the modeler of GPSS/360 TABULATE macroinstruction blocks at strategic points in the model produces detailed measurements of response time for all cyclic work. Such response measurements are indispensable to evaluating the successful completion of all cyclic processing.

For models of the orbit phase, the analyst might be more interested in the number of times that each load module was fetched versus the number of times each load module was executed. (Both items are available in a GMOS table.) He might also be interested in the number of purges that were necessary in a given time (derivable from block counts) and the degree of main storage fragmentation at periodic intervals. The latter is available by forcing periodic output of a GMOS-maintained storage map.

In either case, the analyst might be interested in knowing the traffic volume on various real-time lines and system I/O channels, both available in standard GMOS output. Also produced by GMOS are tables showing the frequency of use of each of the standard RTOS/360 control program services.

Thus, the analyst automatically receives a great amount of information through the use of GMOS. The great flexibility of GPSS/360 allows the gathering of many additional statistics with only small modifications to GMOS or the application system model. One of the benefits of such an abundance of information has been the occasional discovery of an important fact from statistics that was not originally considered important.

three studies

Three studies are presented here as examples of the variety of problems analyzed at the RTCC by means of GPSS/360 models.

The first study was made to determine the significance of main storage to an Apollo real-time application system. While the size of the simulated main storage was varied, the response of a significant calculation cycle was noted. Although this study was limited

to one Apollo real-time application, the results are expected to be typical of similar applications. LCS was used as a bulk storage device from which programs and data were loaded into main storage. The amount of main storage thus required is known to be sensitive to the ability of RTOS/360 to refresh storage dynamically in real time.

In view of this, the general approach taken in this study was the following. First, consider improvements to the purge algorithm in RTOS/360 and select an algorithm independent of the specific application. Then consider the application system performance at several storage-required-to-storage-available ratios (S_R/S_A) when such an algorithm is used. With this, one can use the application-system size estimates to determine the expected S_R/S_A ratio and the resulting performance. Curves relating measures of system performance to the S_R/S_A ratio are shown in Figures 2 and 3. The following describes how these curves were obtained and how they are used. A base case was established using a storage size that was larger than the total size of the application program plus RTOS/360. GMOS was used in the simulation with a modified purge algorithm.

The purge algorithm maintains "use counts" to decide which modules to purge. During a single execution of the purge program, all unused load modules and data tables having a use count equal to or below a threshold are purged. All modules having a use count higher than the threshold are retained, but their use counts are reduced to zero. Thus, use counts reflect only uses since the last purge. Data tables that have been updated must be written on an external storage device before being purged. In this study, thresholds of 1 and 2 were considered. In all cases, the threshold of 1 yielded the better performance, and those results are used in this paper.

The S_R/S_A ratio is a measure of the relative size of an application system to a given main storage size. Storage required (S_R) is taken as the total size of load modules, data tables, and subpools. Storage available (S_A) is taken as the size of main storage, minus the total size of RTOS/360 plus system subroutines. Starting from a given S_R/S_A ratio, an increase in the ratio reflects either a growth in application size requirements or a decrease in main-storage available; a decrease in the ratio reflects either a lessening of application size requirements or an increase in available main storage.

Figure 2 shows response times to one-second telemetry-cycle processing, which operates concurrently with half-second trajectory processing. The S_R/S_A ratio is plotted along the horizontal axis. The 1.0 value along this axis represents the base case for which no buffering is required (all programs and data just fit in main storage). The left boundary of the shaded area indicates performance based on initial size estimates. The right boundary shows the performance if this size estimate were increased by 20 percent to account for errors in size estimations.

Note the steep slope in Figure 2 between the first two points on the curve. At the first point, the telemetry cycle is completed

Figure 2 Cyclic response versus storage ratio

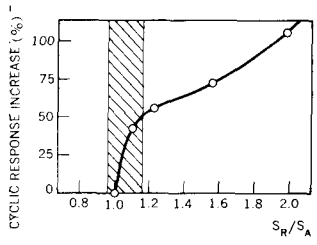
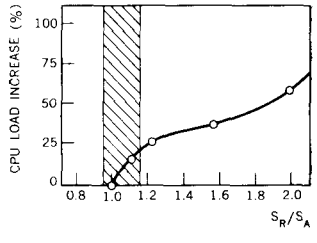


Figure 3 CPU load versus storage ratio



before a trajectory cycle begins. Following the first point, a slight increase in the time required to process the telemetry cycle causes overlap with the next trajectory cycle. Since trajectory has greater priority, the telemetry response is lengthened by an amount roughly equivalent to the length of the trajectory cycle. The relation between telemetry response and the increase in CPU load can be seen by comparing Figure 3 to Figure 2. Figure 3 shows the relationship of CPU load to the S_R/S_A ratio.

The second study, using a model of the FORTRAN-H compiler, was aimed at optimizing throughput by changing I/O device configurations. At RTCC, performance of job-shop runs, SYSTEM/360 assemblies, FORTRAN compilations, etc., are as important to the development of a real-time system as the execution of that system is to the support of an Apollo mission. Job-shop efficiency increases job-shop throughput and decreases turn-around time for debug runs.

In the second study, a compilation was executed as an application model with GMOS to determine how various I/O devices for SYSIN, SYSOUT, SYSPUNCH and SYSRES would change the time to compile a sample source deck of 436 statements using the FORTRAN-H compiler model. Since RTCC has LCS associated with each SYSTEM/360 Model 75, LCS was considered as an I/O device for certain system residence modules from SVCLIB and LINKLIB.

Table 2 shows the parameters that were varied to produce the results; constant parameters are not shown. Production times for an object deck and listing are included in the table. One can see that I/O devices prevent good utilization of the Model 75 CPU. (The IBM 2311 disk was the output device for object code to the linkage editor job step.) Comparing runs 1 and 4, one notices about a 6-to-1 improvement over a standard I/O configuration (run 1)

Table 2 Sample FORTRAN-H compilation results

	Model runs			
	1	2	3	4
Parameters				
SYSIN	cards	tape	tape	tape
SYSOUT	printer	tape	tape	tape
SYSPUNCH	punch	tape	tape	tape
SYSRES	disk (2311)	disk (2311)	drum (2301)	LCS
Results				
CPU utilization (%)	8	23	36	44
I/O waiting (%)	92	77	64	56
Elapsed compilation time (sec)	177	62	39	31

when tapes and LCS are used (run 4). Comparing runs 2 and 4, about a 2-to-1 improvement is observed using LCS in place of the disk, under the assumption that input and output go to or from tape in an off-line process. A sequential job scheduler was assumed in this study.

For the third study, a preliminary design analysis using GPSS/360 model results and SGS timing statistics was made to determine what design changes might improve RTOS/360 performance. Continual emphasis was placed on producing an efficient real-time operating system at RTCC. Because of the frequency with which control program services are used and because of the critical nature of the real-time processing, it is necessary to provide a reasonable margin of safety, so that peak processing loads do not degrade the response required to process in real time. The study determined which RTOS/360 services were heavy users of computer capacity, with the objective of planning ways to reduce the computer capacity required for these services. Frequency counts for use of control program services were obtained from the GPSS/360 model of the Apollo launch system. Timing statistics were obtained with SGS.

As a result of this study, several design changes were recommended, resulting in CPU capacity savings ranging from fractions of a percent to twenty percent. One of the most significant changes proposed was the elimination of references to the main storage supervisor (MSS) for all control tables and temporary storage required by the RTOS/360 control program. It was calculated that if all references to GETMAIN, FREEMAIN, and REGMAIN were replaced by references to preallocated fixed-size buffers, storage still could be provided to control program services as required, with up to twenty percent decrease in CPU utilization. It was also noted that requirements for supervisor request blocks (SVRB) and program request blocks (PRB) constituted over half of the demands on MSS by RTOS/360. Based upon these results, the cost of additional tailoring to the RTOS/360 environment seemed to be justified.

Concluding remarks

Experience and techniques used at RTCC to analyze computer system performance have evolved through long exposure to the problem of assuring workable system designs or problem solutions. The original development of the techniques discussed in this paper began in mid-1963 for use with Gemini systems and the 7094-II. When problems in computer size or speed arose, new computers were simulated. When I/O devices caused delays in processing, different I/O devices were simulated to improve performance. When programming design seemed inefficient (either control or application programs), new designs were modeled. Results of these studies were presented for management decision.

Present versions of SGS and GPSS/360 multiprogramming models are improvements over earlier versions, but essentially the same techniques are being successfully used to study the RTOS/360 pro-

gramming systems for SYSTEM/360. Several attributes of these techniques at the RTCC are:

- Measurements of system performance are acquired by using SGS
- Current statistics are used in predictions of future system performance via GMOS
- Application systems are modeled and tested with relative ease by using GMOS

This method of analysis has proved to be accurate and effective. Techniques discussed in this paper have been used to evaluate application systems that run under two major multiprogramming control programs: the RTCC executive control program for the 7094-II and RTOS/360 for SYSTEM/360. Programming applications analyzed include: job-shop throughput performance, configuration studies, and real-time programming design. It seems reasonable to conclude that other programming applications might also benefit from these modeling and measuring techniques.

ACKNOWLEDGMENTS

Implementation and use of the programs discussed in this paper were achieved by a group effort, which the authors are pleased to acknowledge. Special credit goes to T. A. Humphrey, who originated the computer systems analysis for the RTCC and who has managed this work since the early Gemini missions.

CITED REFERENCES AND FOOTNOTES

1. G. Gordon, "A general purpose systems simulator," *IBM Systems Journal* **1**, 18–32 (September 1962).
2. R. Efron and G. Gordon, "A general purpose digital simulator and examples of its application, Part I, Description of the simulator," *IBM Systems Journal* **3**, 1 22–34 (1964).
   C. R. Velasco, "Part II, Simulation of a telephone intercept system," *ibid.*, 35–40.
   A. M. Blum, "Part III, Digital simulation of urban traffic," *ibid.*, 41–50.
   D. F. Boyd, H. S. Krasnow, and A. C. R. Petit, "Part IV, Simulation of an integrated steel mill," *ibid.*, 51–56.
3. An exception to the availability of initial performance statistics occurs when the operating system is a measurable, working product and only the application programs are under development.
4. The reader should be aware that the use of GPSS models, as described in this paper, is not the only alternative to good computer systems analysis. There are other simulation languages such as SIMSCRIPT and Computer System Simulator, which can be used. Work is also being done with analytical models such as queuing theory and Markov models, which have application in computer systems analysis. See: H. M. Markowitz, et al., SIMSCRIPT —*A Simulation Programming Language*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, (1963), and A. L. Scherr, *An Analysis of Time-Shared Computer Systems*, The M. I. T. Press, Cambridge, Massachusetts (1967).
5. The executive control program was implemented by IBM Houston Operations to support Gemini missions and the early Apollo mission in the IBM 7094-II computers. See: J. H. Mueller, "Aspects of the Gemini real-time operating system," *IBM Systems Journal* **6**, 3, 150–162 (1967).