Discussed is an extended facility of job management for OS/VS1. This facility provides spooling and scheduling in a virtual storage system. Its three major components are peripheral services, central services, and queue management.

The job entry subsystem of OS/VS1

by J. H. Baily, J. A. Howard, and T. J. Szczygielski

With the advent of System/370, the existing System/360 operating systems were extended to take advantage of virtual storage capabilities. OS/MFT (multiprogramming with a fixed number of tasks) was extended to become OS/VS1. In addition, the introduction of HASP[1] (Houston Access Support Processor) had demonstrated that the performance and functional capabilities of job management could be significantly improved. Thus, to extend the operating system for the virtual environment and to increase performance and functional capabilities, it was necessary to modify the job management component of OS/MFT.

Historically, job management was responsible for directing and controlling the flow of jobs through an operating system by providing the functions of job processing and command processing. The Job Entry Subsystem 1 (JES1) of OS/VS1 is an extension to job management that provides a centralized facility for spooling[2] and scheduling input and output streams. JES1 operates in a virtual storage environment and incorporates some of the performance and functions of HASP.

This paper describes JES1, contrasting OS/MFT job management with OS/VS1 job management and highlighting the comparative advantages of JES1. The paper outlines the input required by the subsystem and discusses the general functions of the subcomponents of JES1 including, when appropriate, how the functions were accomplished.

Overview

When the objectives of JES1 were being defined, the following areas of OS/MFT job management (Figure 1) were identified as critical for throughput in the new system:

- Multiple copies of many modules of job management were used.
- Readers and writers used problem program partitions, and the reader shared a partition with the initiator.
- The job queue data set contained so many different types of data that queue contention was a problem.
- The system-input/system-output process had a heavy overhead in the allocation and unallocation of direct-access storage space. Also, the availability of direct-access space for a data set was dependent on the degree of segmentation of the total direct-access space.
- Interpretation of the job control language (JCL) and creation of scheduler[3] control tables at reader time and heavy job queue access by both the reader and writer caused unit record devices to be used inefficiently.

Figure 2 illustrates the subcomponents of JES1 and the data sets or devices that they access. JES1 is divided into three parts—queue management, which is the access method to the job queue data set and scheduler data sets,[4] central services, which is the access method to spool, and peripheral services, which serves the I/O devices.

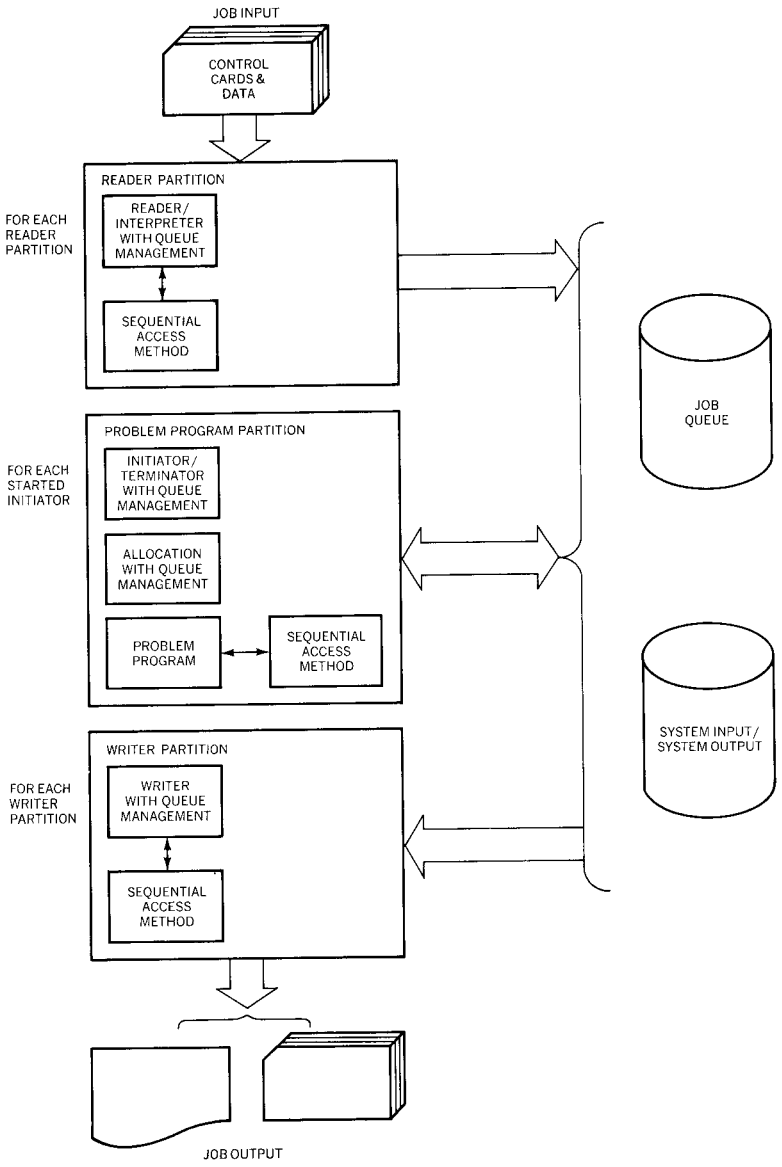
JES1 is a single load module that is resident in virtual storage. The code is reentrant so that multiple copies do not exist and multiple loads do not occur. This is especially important in a virtual environment because multiple copies and multiple loads would increase paging.

Also, with JES1 as a resident reentrant module, the readers and writers need not use up a partition. In addition to freeing problem program partitions, the between-job overhead that existed as a result of a reader/interpreter[5] sharing a partition with an initiator is eliminated. The elimination is significant because sharing caused the two tasks to go through a suspend-restore process for each job that was read or scheduled.

For each job read from the input device, the reader assigns space on the job queue for tables containing queuing and accounting information only. Scheduler control tables are not created from the job's JCL until the job is selected for execution. Then they are written to the scheduler data set, which is a resource of the initiator selecting the job. Also, the system messages related to the job are spooled with the rest of the output of the job. This separation of control tables and system messages from the job queue effectively reduces job queue contention.[6]

The entire process for system-input/system-output data sets was improved. The allocation and unallocation of direct-access space for the data sets uses an algorithm that utilizes a bit map

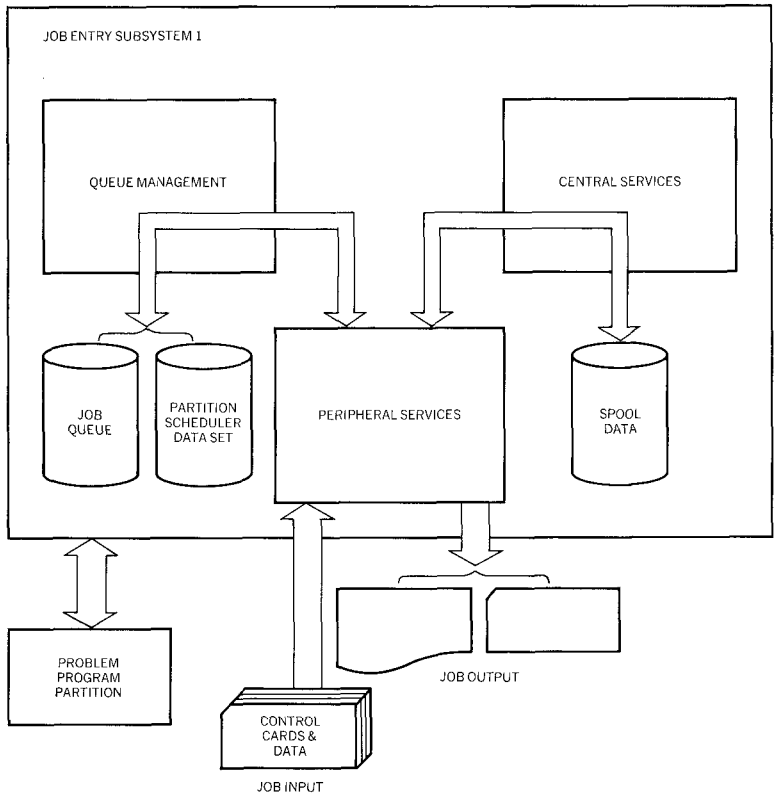
Figure 1 OS/MFT job management



in main storage. No physical I/O operations are necessary to locate or return space. The algorithm balances I/O activity among the spool devices and consistently makes the same amount of spool space available. It is not affected by the segmentation problem of OS/MFT. The writing of data is more efficient because the system spool data set is preformatted.

The last critical area is the efficient use of unit record I/O devices. Both the reader and writer use channel command word

Figure 2 JES1 job flow



chaining. On the reader side, however, an even more important change was the reader/interpreter split. Removing the interpretation of JCL and the creation and writing of scheduler tables from input stream processing frees the reader to read and spool the input stream as fast as it can drive the input device.

Thus the areas within job management that are critical for throughput were addressed in the design of JES1. In addition, other areas of job management required modifications. The main change was to make the interface to JES1 an integral part of the system and to maintain the full restart capabilities[7] that existed in OS/MFT.

External impacts

We now look at some specifics of JES1. The primary consideration will be the ways in which JES1 affects the problem programmer, the system operator, and the system programmer. External compatibility with OS/MFT was the rule when JES1 was imple-

mented. The intent was to place the bulk of the responsibility for change on the system programmer and to impact the system operator and problem programmer only to the extent required for new functions.

problem programmer

The *OS/VS1 Planning and Use Guide*[8] lists the known problems that a problem programmer may encounter. For example, no execute channel program (EXCP) level logic is permitted for system input or output. While there are differences between OS/MFT and OS/VS1, they are so minor that JES1 has caused little or no impact on the problem programmer. With the inclusion of the JES1 spooling facilities for system input and output data sets, the problem programmer may want to reconsider his system input or output processing for old applications and, certainly, for new applications. For example, JES1 blocks and deblocks the user's data, and as a result, the problem programmer gains little or nothing from his own blocking algorithm.

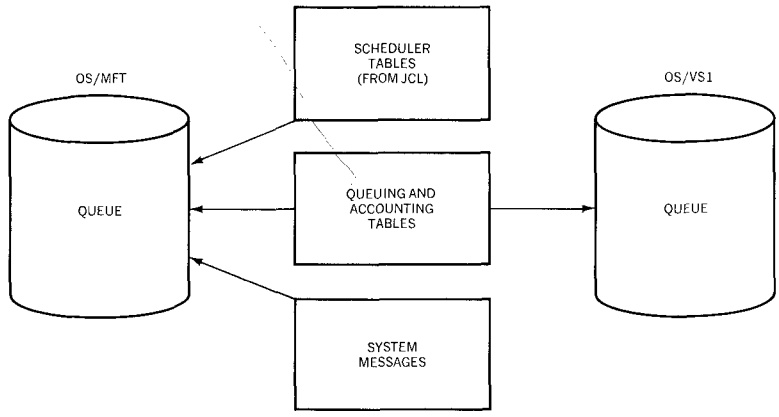
system operator

The changes for the system operator are minimal and generally required for new functions. The operator, unlike the problem programmer, becomes aware of JES1 the first time that the operating system is initialized, or loaded into main storage. Three main changes in JES1 affect the system operator. First, the operator may format all spool volumes or invoke the spool change processor. The change processor allows the spool volume list to be reset at system initialization time. The second external change related to the operator is the "hot" reader capability that leaves unit record readers open until a stop command is entered. This feature reduces the amount of operator action required to process multiple job streams. Last, a new writer command permits operator control of system output data sets being printed or punched. The operator can forward space or back space a given number of logical pages in the data set being printed or punched. He can override the control characters to force single, double, or triple spacing of the data set, suspend the job being processed and enqueue it to the system output hold queue, and make up to 255 copies of the data set or job currently being printed or punched.

system programmer

The system programmer is impacted by JES1 in three main areas: the configuration definition, system task procedures, and JES1 system data set definitions. The system programmer defines the JES1 configuration with the JES system generation[10] macroinstruction. The result of the macroinstruction is the job entry subsystem communication table (JESCT) and the JES parameters member of the system parameter library. The JES parameters member permits changing the configuration without repeating the system generation process. The automated system initialization feature of OS/VS1 (Release 2) permits different JES parameter members to be created and used for different system initiali-

Figure 3 OS/MFT and OS/VS1 queue comparison



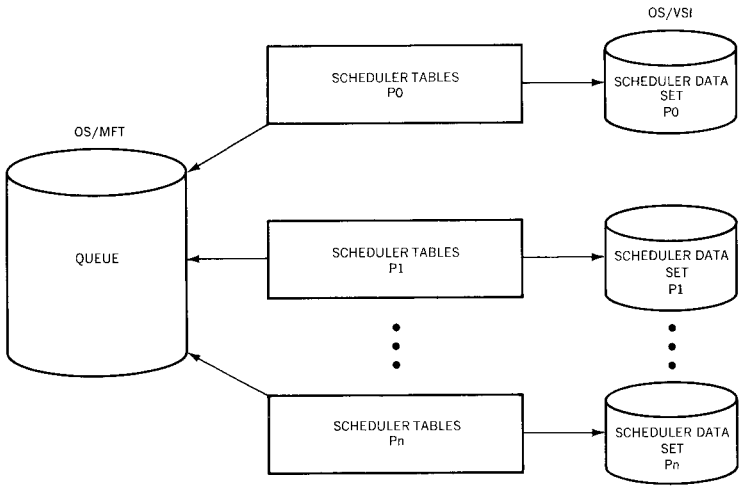
zation requirements. JES1 configuration and automated system initialization are intended to give the system programmer flexibility in tailoring his operating system.

The system programmer can use the parameters in the JES system generation macroinstruction to specify for central services the spool allocation unit, the number and size of the buffers into which central services will block data being transferred to and from spool, the maximum number of output records to be allowed for a job, and the volumes that contain spool data sets. He can also specify the percentage of spool capacity which, if exceeded, causes central services to inform the operator and suggest corrective action. For peripheral services, he can specify the maximum number of readers and writers that may be active concurrently, the blocking factor for unit record I/O activity, nonunit record block sizes, and the amount of storage required for user-supplied writer routines.

The system programmer is affected by several changes to the reader and writer cataloged procedures. Through the reader procedure, he can control whether procedures invoked by jobs in the input stream are written to the spool data set or read directly from the system procedure library at the time the job is selected for execution. In addition, the parameter field in the writer procedure has been extended to provide for a variable number of job separator pages, to optionally translate unprintable characters to blanks, to specify the number of lines per page of printed output, and to specify the checkpoint interval for system output data sets.

The third major area in which JES1 impacts the system programmer is estimating the required size of the system data sets. Sec-

Figure 4 OS/MFT queue compared to OS/VS1 data set

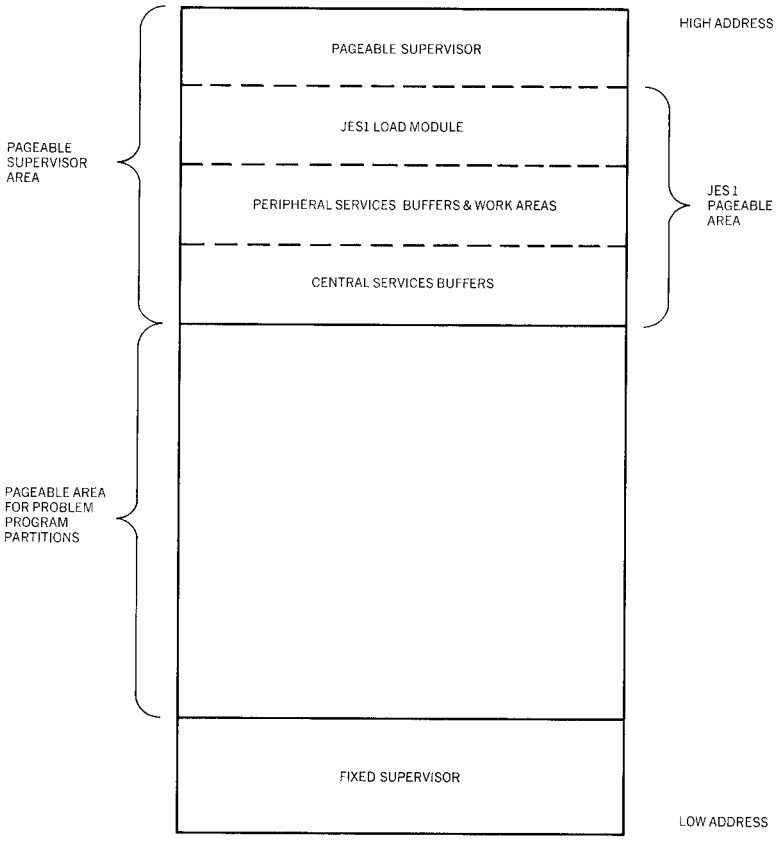


ondary storage estimate formulas give some direction, but the formulas contain such variables as the number of jobs on the input and output queues and the number of data sets in the system at one time. The variables are difficult to estimate because they must be estimated for the variable set of jobs at peak time. To some extent, JES1 has simplified the definition of system data sets.

To estimate the size of the job queue (Figure 3) in OS/MFT, the following questions had to be answered: At peak time, what is the total number of input and output jobs? What is the total number of tables generated from JCL? And what is the total number of system messages? In OS/VS1 the situation is simplified. The question generally is: How many input and output jobs must the job queue be capable of holding?

A scheduler data set is associated with each initiator (Figure 4). Defining the scheduler data set gets the system programmer back to the task of estimating the amount of JCL because the scheduler data set is used for scheduler tables that are the result of JCL interpretation. With JES1, however, the estimate is made for the jobs, or generally the worst-case job, that will run under a single initiator. Answering the question for a worst-case job for one initiator is easier than answering the question for all jobs under all initiators. One other advantage exists with the association of a unique scheduler data set with each initiator. If the estimate for a data set is low, the space-critical problem affects one initiator, not the whole system. The initiator with a problem is stopped and restarted, specifying more space. The rest of the active initiators remain unchanged.

Figure 5 JES1 in the virtual environment of OS/VS1



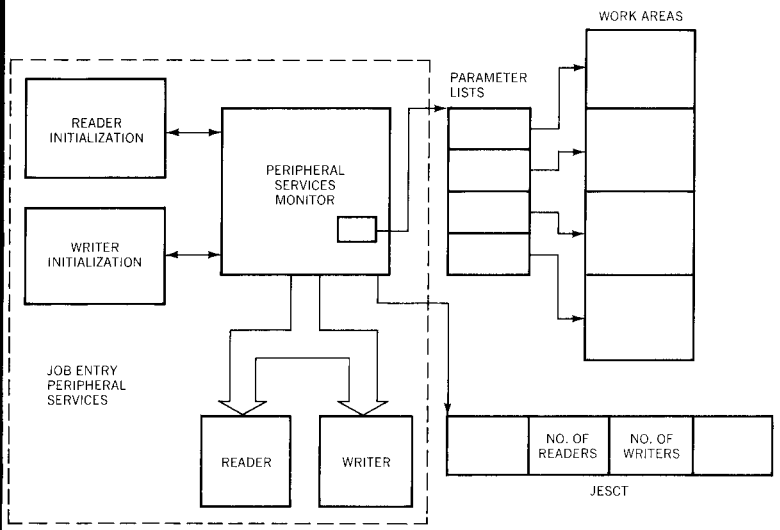
Estimating the size of spool is similar to estimating the amount of system input/output scratch space in OS/MFT. It is directly related to the amount of system input and output that will be in the system at any one time. However, the segmentation problem that existed with the direct-access device storage management of OS/MFT does not exist with JES1. The entire amount of space allocated for spool will be made available.

Internal process

In looking at the external impacts of JES1, we have essentially seen what input is provided to the subsystem. We will now look at JES1 at a lower level and see how the input is processed by each of the JES1 components.

Figure 5 illustrates the virtual storage location of JES1 in the VS1 system. Note how the supervisor area is divided into a fixed

Figure 6 Peripheral services monitor



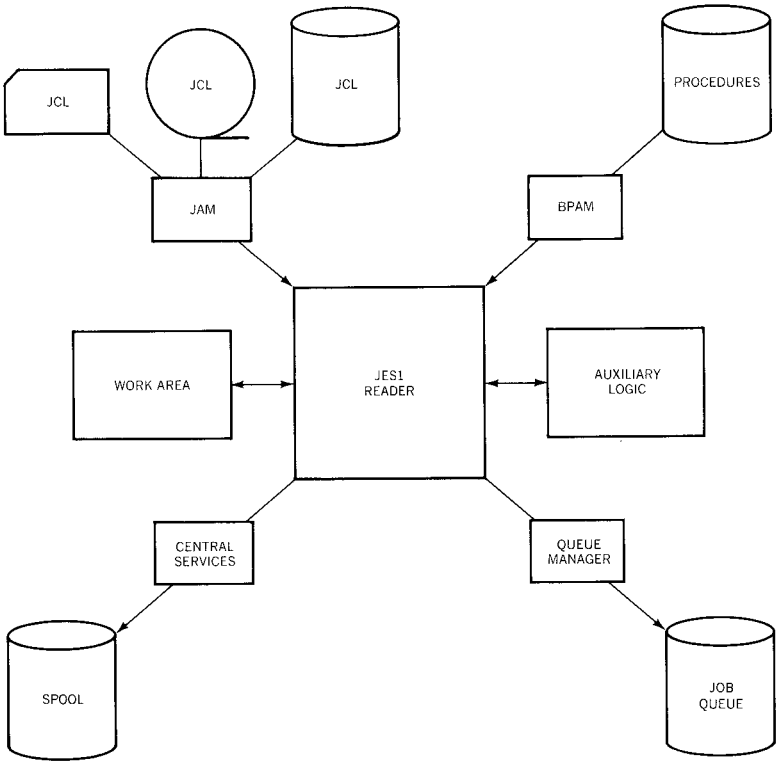
area at the low end of real storage and a pageable area at the high end of virtual storage. Except for about 600 bytes of fixed I/O appendage code, all of the JES1 modules are pageable and are loaded into the pageable supervisor area, directly above the problem program partitions.

peripheral services

Job entry peripheral services consists of a monitor, a reader, and a writer. The monitor (Figure 6) controls the starting and stopping of all JES1 readers and writers. It is the main task in the subtasking structure within the JES1 partition, and attaches[11] all JES1 readers and writers. At the time of system initialization, the monitor obtains the system generation value for the maximum number of concurrent readers and writers from the job entry subsystem communication table and obtains virtual storage for work areas and parameter lists associated with each reader and writer. Each work area is a 2K-byte block aligned on a page boundary. The monitor also opens a data control block for the system procedure library and stores a pointer to it in the communication table. A copy of this control block is used by the interpreter to access the system procedure library. This eliminates the overhead in use of the macroinstructions OPEN and CLOSE[12] by the interpreter for each job.

When a reader or writer is started, the monitor determines if the maximum number of readers or writers is being exceeded by comparing the count of active readers or writers against the maximum number permitted. The monitor obtains a parameter list, saves information in it that will be required during stop processing, then initializes a work area with reader- or

Figure 7 JES1 reader

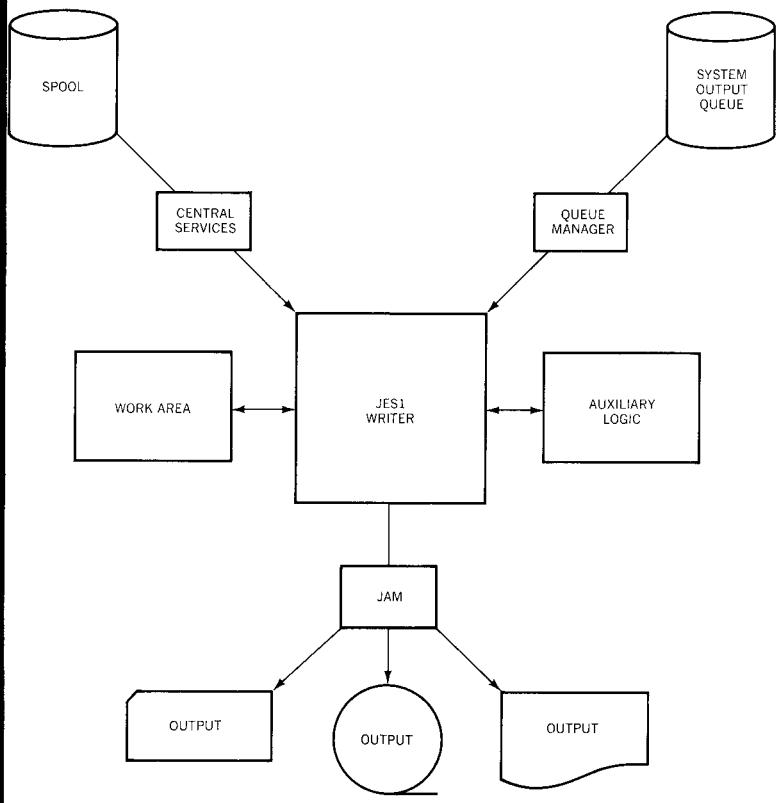


writer-dependent information and attaches the reader or writer task. On the ATTACH macroinstruction, the monitor specifes an exit routine that will receive control when the reader or writer closes. At that time, the monitor uses the information in the parameter list to detach the subtask and invoke a partition to deallocate the devices.

The JES1 reader (Figure 7) consists of a main logic module that reads and spools the input stream and an auxiliary module that processes commands in the input stream, handles the "hot" reader interface, does error processing, spool and queue interlock processing, and procedure handling. Only one copy of the reader code will be loaded no matter how many reader tasks are active. Therefore, the storage overhead for each additional reader task consists of a 2K work-area page and buffer space in virtual storage, plus fixed storage for control blocks.

The reader work area contains the control blocks and parameter lists used to interface with the access methods. The input stream is read using the JES access method (JAM).[13] This access method provides channel command word chaining for unit-record

Figure 8 JES1 writer



devices. The same interface is used for unit-record, direct-access, or tape devices. All JCL and system input data is spooled through central services. A separate spool data set is created for each job's JCL, each system input data set, and, if necessary, the cataloged procedures invoked by each job. If the reader procedure indicates that the cataloged system procedure library is being used, the reader does not read and spool procedures. Instead, they are read directly from the procedure library by the interpreter after the job is selected. If other procedure libraries are required, the reader uses the Basic Partitioned Access Method (BPAM) to access the procedures and spools them for the interpreter. Finally, job-related queuing and accounting tables are created and written to the job queue through the queue manager, and the job is made available for selection by an initiator.

The JES1 writer (Figure 8) consists of a main logic module for printer output, a main logic module for punch and tape output, and auxiliary modules for processing job separators, spanned records, user writers, end-of-class conditions, and commands. Like the JES1 reader, the storage overhead for each additional

writer consists of a 2K work-area page and buffer space in virtual storage, plus fixed storage for control blocks.

The writer work area contains control blocks similar to those used by the reader interface with the access methods. Queue management is used to select an output job and read job-related tables from the job queue. The data from the job queue points to spool, and the writer uses central services to read the system output data. Reading continues until an end-of-data-set condition is returned, with each record being transferred to the output device using the JES access method.

The writer selects and processes system output data sets by class within a job. Multiple copies of the data set will be produced if requested by the programmer in his JCL or by the operator in the writer command. In other words, with the JES1 writer, it is not necessary to rerun the job or use multipart paper to produce multiple copies. After the last system output data set in each class is processed, the writer frees the queue space associated with that class and decrements the count of classes for that job. When the count of classes within the job reaches zero, all output for the job has been processed, and the writer frees the remaining queue and spool space associated with the job.

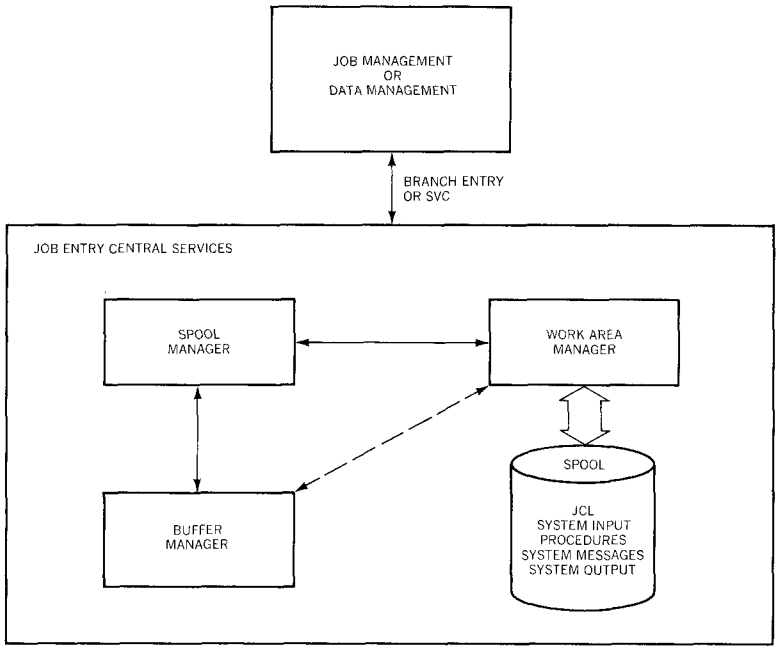
The interface to user-written writers and job separator routines is unchanged from OS/MFT. Control character processing is also compatible with OS/MFT. However, the JES1 writer monitors control characters for the beginning of logical pages to perform the hold, forward-space, and backspace functions of the writer command, and to checkpoint system output data sets.

Summing up peripheral services, we have seen that the subtask structure that permits multiple readers and writers in one system partition, the use of reentrant code, the packaging of the reader and writer into main logic and auxiliary routines, and the use of page-boundary aligned work areas are all intended to enhance performance in a virtual environment.

central services

The second component of JES1 is job entry central services (Figure 9). Central services contain the service routines necessary to spool input and output. It executes under control of the caller, which generally is either job management or data management. Data management calls central services on behalf of the problem program. In order to maintain compatibility for the problem program, the data control block interface to data management is still valid for system input/output data. Data management recognizes when any BSAM (Basic Sequential Access Method) or QSAM (Queued Sequential Access Method) request is for system input or system output and converts the data control block interface into an interface with central services. When

Figure 9 Job entry central services



the problem program calls data management, and data management determines that central services is required, a supervisor call instruction (SVC) is issued. Having data management and the supervisor call as the interface between the problem program and central services provides for data security. It allows central services to receive control in key zero and thus allows for the blocking of the user's data into a protected buffer. Also, the supervisor call instruction itself verifies the caller's key against the key of the caller's data.

The main services performed by job entry central services are to open a spool data set, put a record to or get a record from the data set, close a spool data set, and checkpoint the control blocks. This last service allows for the retrieval of spool data sets and the allocation units that they use. To perform all of these functions, the spool management subcomponent is invoked. It uses work-area management to allocate logical cylinders for the data sets being created and to access data on the spool. It uses the buffer manager to attach buffers to the data sets that are processed.

The first subcomponent of central services is work-area management. In order to understand the function of work-area management, it is necessary to see how the spool configuration is processed at system initialization time. Assuming that two spool

volumes were created in system generation, we see the result of the JES input in Figure 10. JES1 makes a single track range for all input tracks and saves information to relate any track identifier with the correct spool volume. Using the buffer size and spool allocation unit, which were specified by the system programmer in bytes, the system calculates for each volume the number of tracks that will be a spool allocation unit. This number of tracks is called a logical cylinder.

cylinder maps

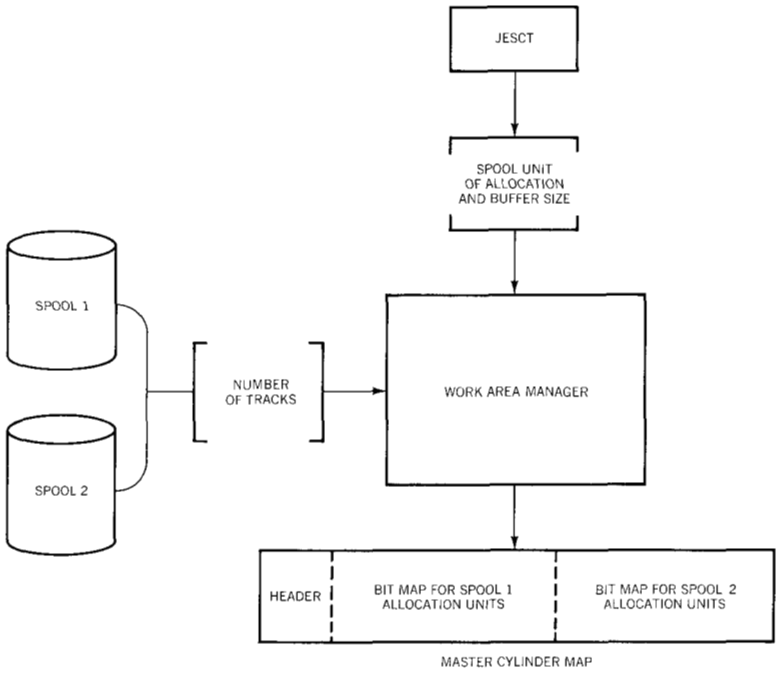
A master cylinder map (MCM) is built and used to manage the spool logical cylinders. The map contains one bit for each logical cylinder. Initially all bits are on, indicating that all logical cylinders are available for allocation. When logical cylinders become unavailable for general use, it means they are owned by some job. Such logical cylinders are recorded in a job-related table called the job cylinder map (JCM). For every bit off in the master map, a corresponding bit is on in the job map of some job. The job maps are the same size as the master map. Thus, when a job is finished with its logical cylinders, a simple OR instruction returns the allocation units to the available pool.

Work-area management is divided into two functions: allocation and the read/write routine. The prior description of the cylinder maps shows what the allocation process is: turning a bit off in the master map and on in the job map. The spool allocation algorithm also consists of balancing the I/O load on spool. In order to achieve the balance, the algorithm goes through two considerations. First, the spool volume chosen for an allocation is the one that has space available and the lowest average spool I/O time. Second, once that volume is determined, the logical cylinder chosen is the one closest to the position of the arm of the direct-access device.

Once an allocation unit is chosen, it is converted into a track identifier that is used by spool management for subsequent I/O requests to the read/write routine. The spool read/write routine converts its input into an interface with the I/O Supervisor (IOS). The interface is a macroinstruction called EXCPVR (Execute Channel Program Virtual).[14] It is similar to the EXCP instruction and was written for system access methods that are critical I/O areas in a virtual storage system. The interface takes advantage of the facts that the spool I/O work area and buffer are adjacent and that the channel program is always the same string of channel command words. With the above two assumptions, the generalized IOS code of EXCP that fixes I/O related areas and translates channel command words from virtual to real addresses is bypassed.

The second component of job entry central services is buffer management. At system initialization time, the central services

Figure 10 Work-area manager



I/O work areas and buffers are formatted. This ensures that a minimum amount of work is done when I/O activity to the spool begins. Buffer management initialization ensures that the first work area and buffer are on a page boundary. By specifying a buffer size so that the work area and buffer together are divisible by 1K, the system programmer ensures that the input and output data will be on the smallest number of pages and that the channel command words within the I/O work area will never cross a page boundary. This helps to reduce the processing time for the EXCPVR instruction because fewer pages need to be fixed and unfixed and the channel command word translation process is simplified.

When a buffer is needed, buffer management attaches the work area and buffer to a spool data set. The same area stays with a data set until close time unless the system programmer specified too few buffers. In this case, buffer management provides for the sharing of buffers between data sets. This process ensures that the system will never stop because of a lack of buffers. However, it involves a high I/O overhead. Thus, when the system programmer specifies the number of buffers, it is recommended that a high estimate be made.

Spool management is the last and main subcomponent of central services. It is responsible for creating new data sets and extend-

ing or retrieving data from old data sets. Spool management can be divided into two areas: data set maintenance, that is, open and close activity, and the get and put process. At open time, control blocks are built, the beginning track identifier for the data set is saved, and the data set is made ready for get and put operations.[12] At close time, control blocks and work areas are cleaned up and the ending track identifier is saved. Both the open and close processes are very efficient because they involve little or no I/O activity.

The get-put modules of spool management perform several functions. They block and deblock the user's data, a facility that allows system input/output applications to forget about blocking. Get and put truncate and expand the user's data. This feature is important for performance because by truncating the ending blank characters, the space in the central services buffer is used more efficiently, resulting in less I/O activity. In the put module, the facility to suballocate the logical cylinder allows for fewer spool allocation calls and is significant in small systems where a call to a subroutine could cause page-fault I/O activity. Also in the put module, spool management chains the records belonging to a spool data set. This has two advantages. First, a logical cylinder can be shared between data sets of a job. This prevents wasting direct-access storage device space when, for instance, several system input data sets of a job each contain only a few records. Second, chaining the data reduces the number of checkpoints that are necessary for recovery at system restart time.

queue management

The last component of JES1 is queue management. Queue management has been centralized with the subsystem and functions much the same in OS/VS1 as in OS/MFT, the main change being in routing data to either the job queue or a scheduler data set. When allocating a record on the scheduler data set, a sequential algorithm is used as opposed to the chained, nonsequential algorithm of OS/MFT. This is possible because the same data set is used by the initiator for each job it selects. These functional changes allowed the scheduler control tables to be removed from the job queue and are the prime reasons why the job queue contention problem does not exist with JES1.

Summary

In this paper, the facilities of the job entry subsystem of OS/VS1 have been described briefly. The comparison between OS/MFT job management and OS/VS1 job management identified several areas where performance could be improved and described the approaches taken by JES1 for such improvement. The input to the subsystem, primarily through system generation parameters, was considered. Compatibility with OS/MFT and the flexibility

provided by the JES1 reconfiguration facility were the main points of this discussion. In a brief analysis of the JES1 subcomponents, some of the techniques that permit JES1 to perform efficiently in a virtual storage environment were illustrated. Thus, it was shown how JES1 extended OS/MFT job management by providing additional functions and improving performance in the virtual storage environment.

CITED REFERENCES AND FOOTNOTES

1. The following publications describe HASP: *OS/VS2 HASP II Verson 4 Operator Guide*, Form No. GC27-6993; *OS/VS2 HASP II Version 4 Logic Manual*, Form No. GC27-7255; *OS/VS2 HASP II Version 4 System Programmer Guide*, Form No. GC27-6992, IBM Corporation, Data Processing Division, White Plains, New York.
2. The term "spool" is coined from "simultaneous peripheral operation on line."
3. The scheduler is the part of the job processor that selects, initializes, and terminates jobs and ensures that the jobs have the required resources. The scheduler is also referred to as the initiator.
4. A scheduler data set is allocated to each initiator at start time. It contains the job-related control tables used by the scheduler. For Release 3 of OS/VS1, the system operator has the option at start initiator time of specifying that virtual storage should be allocated and used in place of the data set.
5. The interpreter is the component of job management that creates scheduler control tables from JCL.
6. Job queue contention and I/O activity are further reduced in Release 3 of OS/VS1 by definition of a job list within virtual storage. The reduction is especially significant during command processing because it replaces nonsequential I/O activity with a virtual storage scan algorithm.
7. Restart recovery is the ability to recover jobs and data after a system failure and/or to restart a job that failed and recover its data. In addition, for Release 3, the job data records that remain on the queue data set are blocked, allowing for another reduction in I/O activity.
8. *OS/VS1 Planning and Use Guide*, Form No. GC24-5090, IBM Corporation, Data Processing Division, White Plains, New York. More information on OS/VS1 can be found in References 9, 10, 13, and 14.
9. *OS/VS1 Job Management Logic*, Form No. SY24-5161, IBM Corporation, Data Processing Division, White Plains, New York.
10. *OS/VS1 System Generation*, Form No. GC26-3791, IBM Corporation, Data Processing Division, White Plains, New York.
11. B. I. Witt, "The functional structure of OS/360, Part II, Job and task management," *IBM Systems Journal* **5**, No. 1, 12–29 (1966).
12. W. A. Clark, "The functional structure of OS/360, Part III, Data management," *IBM Systems Journal* **5**, No. 1, 30–51 (1966).
13. *OS/VS1 OPEN/CLOSE/EOV Logic*, Form No. SY26-3839, IBM Corporation, Data Processing Division, White Plains, New York.
14. *OS/VS1 I/O Supervisor Logic*, Form No. SY24-5156, IBM Corporation, Data Processing Division, White Plains, New York.