JOBSTREAM SIMULATION USING A CHANNEL MULTIPROGRAMMING FEATURE

S. E. McAulay
International Business Machines Corporation
San Jose, California

Summary

A rotational position-sensing feature designed for direct-access devices reduces the time a hardware channel is busy searching for disk records. This is accomplished by electronically dividing the track of a disk into equally spaced timing sectors, using new hardware commands to access desired sectors. When an I/O record located some distance away in rotation is requested, the channel disconnects to allow concurrent servicing of requests from other devices. Through a combination of hardware and software, multiprogramming at a channel/disk level is achieved, thus improving channel performance.

In this study, overall effectiveness of the feature was evaluated in a multiprogramming environment. Simulation at high and low levels simultaneously required a unique approach in the models. Hardware models were based on close tolerances and exacting operations, while software jobstream models were more generalized. The objective was to simulate a typical jobstream in a typical multiprogramming environment on specific hardware.

Introduction

Data transfer within a computer can be an inefficient process. I/O channels may be busy most of the time, yet only a fraction of that time is used in the actual transmission of data.

A solution to speeding up the retrieval of data from a disk has been introduced in the form of a feature called rotational position sensing (RPS). This feature combines hardware and software to allow multiprogramming of direct-access programs on a channel. The multiprogramming provides the means for enabling a channel to increase its throughput, thus improving overall system performance.

The use of RPS on a direct-access device reduces the time a channel is busy searching for a disk record. A sector concept is employed; that is, the surface of the disk is divided into equally spaced timing sectors. Through new hardware commands the individual sectors are accessed with negligible channel activity. Record searching, which requires a dedicated channel, may now be initiated just before the desired record passes under the read/write head. Track formatting and record placement is unchanged, although each record has a sector location as well as a record address (see Figure 1).

Simulations of system performance were made to determine the effectiveness of the RPS feature. The model for carrying out the simulations is described in the next section. This is followed by the results of the simulation runs.

Simulation Vehicle

CSS/360 was used in this study. It is a simulation program designed to analyze the operation of computer systems, especially System/360 configurations. In concept it is similar to the General Purpose System Simulator (GPSS), differing in one aspect: it is not general, but applies specifically to computer systems. Thus, specific hardware may be accurately described with parameters such as data transfer rate, seek timings, channel data rate, CPU speed, etc.

Jobstream Concepts

The CSS package which was developed looks like OS/360. Instead of performing work, the models merely add time increments to a counter. However, OS-like tables are kept, tasks are created, and I/O models are called. The data management models require data control block definitions analogous to OS data management. Figure 2 shows the total CSS environment.

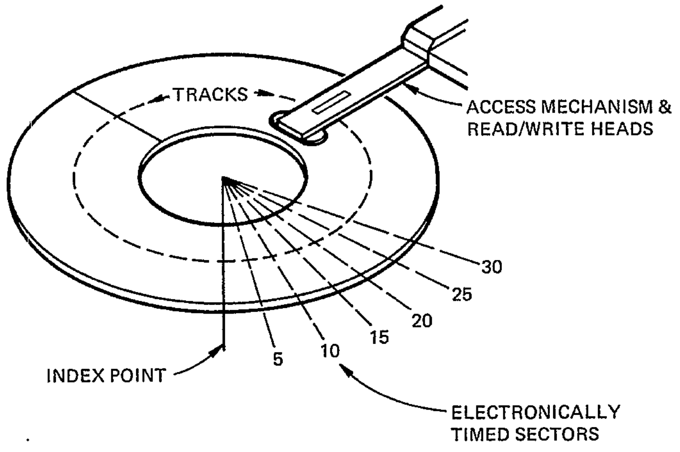
TRACKS
ACCESS MECHANISM & READ/WRITE HEADS
30
25
20
15
10
5
INDEX POINT
ELECTRONICALLY TIMED SECTORS

Figure 1 Disk surface. Tracks follow the circumference of the recording surface with an index point as the logical beginning of the track.

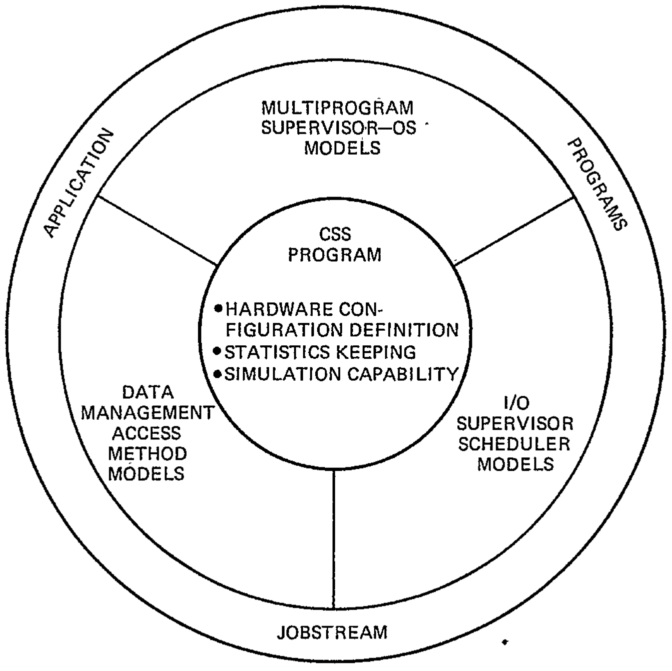
APPLICATION
PROGRAMS
MULTIPROGRAM SUPERVISOR—OS MODELS
CSS PROGRAM
•HARDWARE CONFIGURATION DEFINITION
•STATISTICS KEEPING
•SIMULATION CAPABILITY
DATA MANAGEMENT ACCESS METHOD MODELS
I/O SUPERVISOR SCHEDULER MODELS
JOBSTREAM

Figure 2 Total CSS environment.

The simulated application program jobstreams were created to simulate typical user jobstreams. Most often used was a GET-GET-PROCESS-PUT-PUT algorithm. This algorithm images the action of updating an account. Old master and update records are processed, and a new master is written along with a report of the transaction. This report under normal multiprogramming systems is initially written on a disk in an output queue before printing.

Other I/O request algorithms were simulated, including the following:

GET-PROCESS-PUT	-- a file update operation
GET-PROCESS	-- a simple retrieval
PUT-PROCESS	-- format operation, addition or file creation

Each algorithm represents processing within an individual task. These tasks were combined in several variations to simulate a multiprogramming environment. A four-task operation was simulated most often, representing typical multiprogramming usage with 512K core size. This core size is assumed to be common for a large segment of potential users.

Three record sizes were used in most simulations: full track, one-fourth track, and one-eighteenth track. Full-track blocks represent efficient use of device capacity. One-fourth track blocks represent a compromise, losing some device capacity and gaining core from decreased buffer space. The one-eighteenth track blocks (298 bytes on a 2314) approximate frequently used sizes with 75% efficient use of device capacity and small core requirements for record handling.

The simulated unit record equipment is assumed to be on a separate channel, not affecting operation of the RPS. The only effect is an increase in processing time within the I/O request algorithm.

Processing times were determined experimentally. In some cases the values were set to match the System/370 runs. In other cases, values were set to match CPU utilizations of some IBM systems as revealed by recent studies. Many of the simulations show CPU utilizations around 15-30%. Test simulations showed little or no I/O performance change in most cases, when increasing CPU utilization as high as 50-60%.

Assumptions Used in Defining Hardware Configurations and Jobstreams

The CSS program includes many assumptions relating to S/360 operations:

1. Timings -- based on S/360 Model 65.

2. Disk geometric characteristics -- full track capacity = 7294 bytes.

3. Seek functions -- defined with a continuous function having minimum of 25 milliseconds and maximum of 130 milliseconds.

4. Data rate -- 312,000 bytes/sec.

5. Channel disconnection and reconnection -- after disconnection of channel, attempt to reconnect is made 1.4 milliseconds before record (lead time). Reconnection attempted for 31 microseconds (pulse width of hold time).

6. Rotation speed -- approximately 2400 RPM (+0 to -0.8%, defined in terms of 25,000 to 25,200-microsecond cycle time).

7. Priority -- No I/O Supervisor priority other than default priority scheme when searching queues; in this case highest numbered partition or region has priority in scheduling new activity.

8. Nonformatting writes -- actual output under control of Data Management is usually done with formatting write (writing high-density one's after the data until the beginning of the track is again sensed, to erase old data).

9. Search direct -- strategy simulated and used with fixed record input processing.

10. Queued requesting -- overlapped processing used for most simulations (sequential processing on one volume).

11. Buffering -- default of two program buffers used most often. Some simulations made using from 1 to 18 buffers. Normal buffer management assumed with refilling of buffers by I/O supervisor routines automatically when retrieving.

12. Simulated time -- ten seconds of processing time used most often. This value was found to represent a minimum point of stabilization within the simulator.

13. MFT logic -- multiprogramming with fixed core partitioning was assumed with no considerations for core restrictions. Where multiple tasks were simulated, simultaneous starting was assumed with no delays for such things as operator interventions.

14. System configuration -- as in Figure 3.

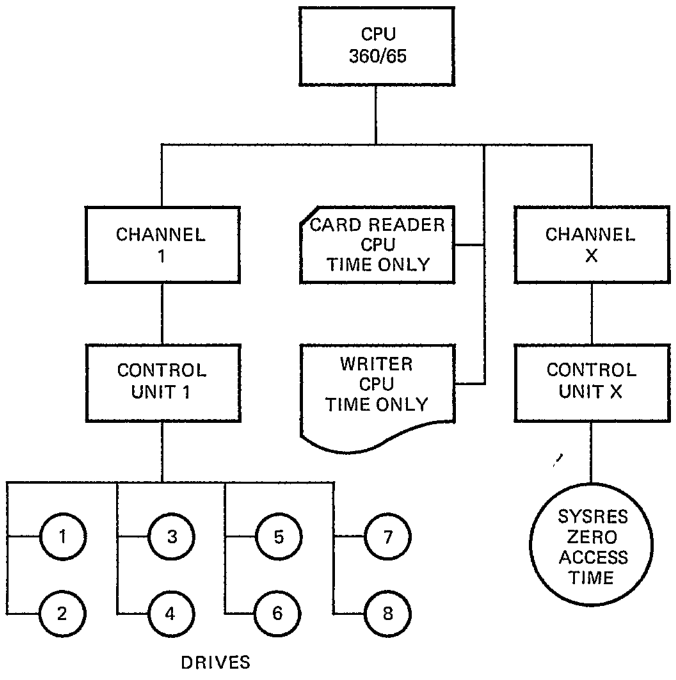

Figure 3 System configuration.

CSS Modifications

The CSS models represented a current operating system. To simulate both present and future systems, they were modified as follows:

1. I/O Supervisor (IOS) models -- changes were made to bypass the stand-alone seek for devices where the I/O lead and hold times are specified (RPS devices). Timings were added for the handling of a channel-available interrupt by the I/O Supervisor. The CSS seek-complete interrupt scheme logically allows for this interrupt as it is being implemented in OS/360. The IOS modifications were made in the IOS portions of the data access method models (Basic Sequential, Queued, and Direct Access Methods).

2. IBM 2314 device characteristics -- geometric characteristics for the 2314 Direct Access Storage Facility (i.e., 7294 bytes per track, 20 tracks per cylinder) were added under a new generic device type. The new device type also allowed variance of rotation speeds within CSS/360.

3. Write validity check -- the ability to simulate disconnection of the channel and position again on the record being checked was added.

4. Search strategy -- simulation of the search-previous technique was added where only search-direct strategy had existed.

5. Formatting write -- this is the method of writing used most often by data management. CSS modifications were made to allow simulation of formatting writes. Formatting causes the control unit to remain in a busy condition after data transfer, until index-point. With non-RPS, device-end and channel-end will be signalled at the end of data-transfer. This allows IOS to handle the resulting interrupt simultaneously with the formatting and frees the channel. With RPS, the last command may be a read-sector if processing sequentially and not calculating sectors. In this case, channel-end and device-end occur after formatting is complete although the channel disconnects until index-point. If the write is not the last command of the chain, as with a read-sector command addition, the effect is one of altering channel program completion point and changing interrupt handling overlap points.

Simulation Run Results

A comparison was made of RPS versus non-RPS runs. In some cases, mixtures of the two environments were used. Results of several jobstream simulations are shown in Table 1. These runs were made simulating both single-task jobstreams and 4-task jobstreams. Our interest focused on changes to channel utilization relative to changes in number of RPS drives, and on overall jobstream improvement. Note that the channel utilization significantly decreased when the environment was totally RPS even though throughput increased. This relief of the channel bottleneck was expected to be a major advantage of RPS. In a multiprogramming system the associated channel availability would provide a potential for increased throughput. This potential would be dependent, however, on additional CPU space or cycle time availability (i.e., whether or not the CPU is already working at full load).

Table 1 Simulations of S/370 runs

Run	RPS drives	Task	Total drives	Channel busy,%	CPU used*,%	Total I/O requests thru/ minute**	Improvement,†%
1A	None	1	4	95	12	2232	Base
1B	All	1	4	64	12	2670	+20
2A	None	4	4	94	21	3234	Base
2B	1 of 4	4	4	94	22	3420	+6
2C	2 of 4	4	4	92	29	4404	+36
2D	3 of 4	4	4	95	16	2401	-26
2E	4 of 4	4	4	23	29	5100	+58

*CPU speeds similar to S/360-65.
**Ten seconds total processing time simulated (minimum stabilization point).
†Performance improvement is relative to non-RPS runs which are shown as the base.

The addition of any drive with non-RPS channel programs jumps the channel utilization to over 90%. This increase is mostly due to additional search time which can be considered non-productive time.

The performance improvement figures are certainly not conclusive. The simulations shown in Table 1 were combined with other runs made in the study to draw some conclusions about mixtures of RPS and non-RPS programs. This is discussed further in the section on Simulation of Interference.

The timings of all simulations assumed no I/O time required for system functions. That is, accesses of system libraries were instantaneous with no channel time required. If system libraries were simulated on the same channel, results would be altered significantly.

CSS and System/370 Comparisons

Shown in Table 2 are the comparative runs on System/370 and Mod 44 hardware using an experimental 2314. The measurements were made on experimental hardware and software and were subject to implementation errors. CSS simulation results of identical jobstreams are also shown. The comparative runs were made on a jobstream particularly chosen to show optimistic results, although saturation of the channel was not the objective. No RPS and non-RPS mixtures were run.

Table 2 Performance comparison for System/370, Mod 44, and CSS

Run	Tasks	Jobstream	Record size	(RPS improvement,%) S/370	Mod 44	CSS
1	1	PUT-PUT-PUT-PUT	298,7294	+8	+11	+19
2	1	GET-GET-PROC-PUT-PUT	298	+15	+21	+21
3	1	RUN 2 WITH 2ND PUT VERIFY	298	+13	*	+17
4	1	GET-GET-PROC-PUT-PUT	7294	+5	**	+12
5	4	GET-PROCESS IN EACH TASK	298	+35	+6	+26
6	2	GET PROCESS PUT (EACH TASK)	298,1693	+16	+18	+19

*Write verify not implemented.
**Not sufficient core.

The comparisons of simulations and hardware measurements show a trend. CSS results tend to track about 3 to 10% high in comparison to the System/370 results. Assuming that the trend remains accurate, the simulation models may be used to predict performance results in areas not yet measured by actual runs.

Predictable Variations

The simulations as compared to measurements are slightly optimistic; however, the CSS runs have some variations which, in part, explain the optimism. At the time the simulations were being done, the real implementation was not known. (See the section on CSS Modifications.) First, for output, CSS simulates a nonformatting write in the comparison runs. Actual OS data management in most cases formats unless the disk has been preformatted with rigid record sizes and placement. CSS simulations of the eight runs did not maintain the control unit busy condition while formatting after data transfer. This condition was not available in the model. Second, CSS models, as used for the comparison runs, simulated a search-direct strategy, while actual measurements used a search-previous strategy (e.g., search on record 2 to write record 3).

Simulation of Interference

Interference in an RPS environment refers to the situation where RPS and non-RPS channel programs are concurrently active on the same channel. The non-RPS channel programs tend to pre-empt the others, due to the long periods of channel-busy time associated with their search operations. The RPS channel programs disconnect while the control unit locates sectors. If the channel is busy when the sector is found, the control unit waits one full rotation and attempts reconnection again. The non-RPS programs do not disconnect after starting, thus insuring completion within one rotation. This is efficient for an individual request, but impacts total system performance.

To simulate interference, RPS and non-RPS channel programs were intermixed in a multiprogramming environment. For non-RPS, a model of the present OS/360 queued access method was used as the simulated access method (representing, for example, a user program which does not use IBM standard data management access methods, or the result of mixing a non-RPS disk file on the same channel).

The simulations revealed significant variations in priorities, channel utilizations, and system throughput. In the study, four drives were used with combinations of RPS and non-RPS to determine the effects of interference. Table 1 shows the resulting throughput relationships.

As is obvious from Table 1, the case with only one non-RPS program should be avoided. A performance gain is made with a 2:2 ratio although channel utilization is not lowered. All simulation runs indicated that the case with only one non-RPS channel program mixed with all RPS produces the poorest performance.

Individual Task Performance With Interference

One factor not mentioned so far is the effect of RPS and non-RPS mixtures on individual job performance. Listed in Table 3 are the I/O request relationships for the simulation runs shown in Table 1. Four tasks were checked in the study, mixing RPS and non-RPS channel programs to examine variations of I/O completion rates. The table shows how tasks without RPS channel programs can unbalance a multiprogramming environment. For 2314's, the maximum possible for one drive would be one per rotation which is equivalent to 40/second (2400 rpm).

Table 3 Four-task mixture study showing I/O requests completed per second

Task	Run 1 All non-RPS	Run 2 3 non, 1 RPS	Run 3 2 non, 2 RPS	Run 4 1 non, 3 RPS	Run 5 All RPS
1	13	19	36	40	20*
2	13	19	37	1*	20*
3	13	19	1*	2*	23*
4	13	0*	1*	1*	22*
Total	52	57	75	44	85

*With RPS.

In the mixed runs, the individual tasks supporting RPS are penalized. Note that in no cases are the non-RPS tasks penalized.

The effect produced by changing record sizes is similar to that shown above. One simulation was made with a mix of 298-byte requests with full track requests. The algorithm for requests simulated an updating process. The non-RPS task again shows improvement in the mixed environment at the expense of the RPS task. Shown in Table 4 are the results.

Table 4 I/O completions per second

Task	Run 1 Both non-RPS	Run 2 1st task non-RPS	Run 3 2nd task non-RPS	Run 4 Both RPS
1 (298 byte)	12	64	2.4*	5*
2 (full track)	20	2.5*	24	25*

*Tasks with RPS support.

If only one segment of processing is considered, in light of bytes transferred and I/O request rates, the following results:

Task 1 non (298), and Task 2 RPS (full track)	Task 1 RPS, and Task 2 non	Both RPS
-85% bytes transferred*	-1.5% bytes	+2.9% bytes
+104% requests complete*	-17.6% requests	-7.7% requests

*Total throughput.

The above statistics consider only one segment of time and continuous operation of both tasks. With non-RPS tasks initially dominating the channel, jobstreams tend to end with a pure RPS environment. Under these conditions, it is possible for the pure RPS environment to show sufficient improvement over non-RPS to overcome results of the earlier mixed environment. The potential improvement would, however, be dependent upon length of the non-RPS versus RPS task.

An Additional Performance Factor (Rotational Speed Variations)

Obviously, a shorter cycle time for any given device yields improved performance. One factor often not considered in developing models of hardware, however, is the cycle time of each drive in relationship to the others. Tolerances of the 2314 drives allow ±2% rotation variance from the desired 2400 RPM, while experience reveals an actual variance between 2390 and 2400 RPM, with no samplings showing slower disk cycle times. Intuitively, closer tolerances would seem desirable. In sequential operations, however, a wider range of RPM's between drives improves simulation performance. This apparently results from the randomizing effect applied to record positioning. Without rotation variance, the simulator performance as though all index points were rigidly aligned, definitely not a real world case. Figure 4 shows one case where only rotational speed was varied. A base on drive 1 was used as 25,000 microseconds. Drives 2, 3, 4, 5, 6, 7, 8 have the rotations time lengthened (i.e., 25,010 for Drive 2, 25,020 for Drive 3, etc.).

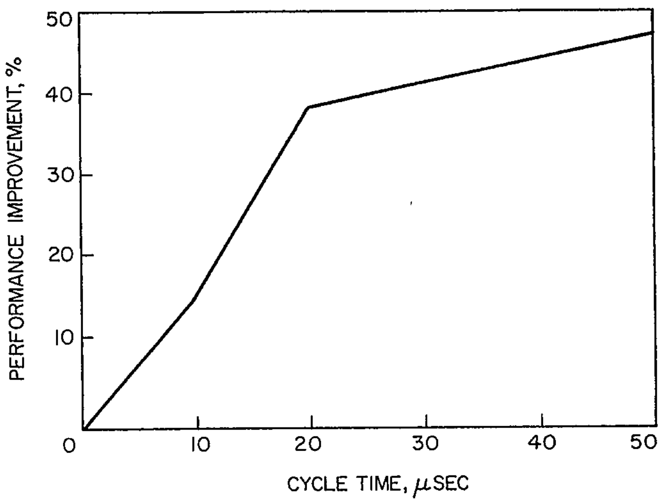

Figure 4 Results of variations in rotation speed.

Conclusions

Performance simulations revealed the following general trends:

1. Pure RPS environments show potential for good results. Future jobmix throughput performance compared to present performance should show general improvement from 5% to 30% without changing hardware speeds, etc., and with no change to application programs.

2. Priority queueing has little effect on the order of completion of the request in a mixed environment, as the non-RPS requests always have priority over RPS requests. The overall effect on priorities, however, may be negligible.

RPS is a desirable concept and, if fully utilized, can produce good performance gains. Mixtures of RPS and non-RPS devices on the same channel should be avoided to realize the greatest throughput potential.

Acknowledgements

Initial simulation studies were made using a GPSS model developed by N. L. Chase. Additional studies, including the one described herein, were directed by J. S. Barney. A Shibamiya provided initial modification of the CSS models for RPS simulation while educating the other team members in the design and use of the simulation tool. L. R. Shipman provided modifications and enhancements to the CSS models.