CONSIDERATION IN THE DESIGN OF A MULTIPLE COMPUTER SYSTEM WITH EXTENDED CORE STORAGE*

Kurt Fuchel and Sidney Heller
Applied Mathematics Department
Brookhaven National Laboratory
Upton, Long Island, New York

Summary

This paper discusses the recent innovation of the use of large quantities of addressable (but not executable) fast random access memory in order to heighten the multiprogramming performance of a multicomputer system. The general design of the hardware arrangement and the software components and functions of such a system are based on Brookhaven's future configuration of dual CDC 6600's sharing one million words of Extended Core Storage. In the generalization of such a design, special emphasis is placed on estimating expected gains compared to the traditional configuration of separate and independent computers without ECS. An observation is made on the use of conventional slower speed random access storage devices in place of the faster memory.

Introduction

The paper grew out of a specific proposal at Brookhaven National Laboratory (BNL) for the acquisition of Extended Core Storage (ECS) destined to upgrade dual CDC 6600's. The throughput of a stand-alone CDC 6600 is most frequently limited by available memory. Several jobs (up to 7) occupy Central Memory (CM) at one time. Whenever the job currently active stops, usually because it is awaiting the completion of an I/O request, the Central Processor (CP) is given to one of the other jobs in CM. However, it frequently happens that all jobs in CM are unable to proceed so that the CP is idle. This condition stems from a combination of two causes: too few jobs occupying CM and too many of these jobs are I/O bound. The idle time thus generated is considerable. On machines with 65K of CM it is often of the order of 65% and on 131K machines it may still be as high as 35%. Of course, such figures are extremely dependent on the job mix, and can change radically within a few minutes.

Figure 1 illustrates the output flow on a typical CDC 6600.

It is not clear under what conditions a stand-alone system with a larger CM will out-perform one with a smaller CM augmented by ECS. However, the price ratio between additional CM and ECS is about 12 to 1 and the fact that ECS may be shared by more than one computer makes it an attractive acquisition.

The role that ECS is expected to play in the system is essentially twofold. First, it is expected to drastically increase the number of jobs available to the CP of each computer. This can only be achieved by swapping out a job which is held up for I/O, and swapping in one which is ready to run. As some manufacturers have learned to their cost, attempting to do this with conventional disk or drum hardware poses certain pitfalls: either disk access is too slow, or the "sophisticated disk scheduler designed to minimize disk access" uses a sizable amount of processor time. The second objective is to improve overall I/O efficiency by accumulating output in ECS until enough data is available for optimizing transfer to I/O devices, and similarly buffering ahead on input. In sum, ECS is expected to smooth the load on CP and I/O equipment and to do this without generating a sizable overhead.

This paper will discuss the hardware and especially software alternatives facing the designers of an ECS based operating system; give some estimates of relative sizes of ECS and the resulting expected performance and then generalize some of the results to other configurations in-

* This work was performed under the auspices of the U. S. Atomic Energy Commission.

volving multiple computers and different equipment.

I. General Design Considerations

A. Hardware. The details of ECS hardware have been reported elsewhere.[1] The salient characteristics are that ECS is available in multiples of 131K 60-bit words to a maximum of 2 million words. Transmission is directly between CM and ECS with no channel or peripheral processor (PP) involved. The rate is 3.2μs for the first word, then 8-10 words/μs depending on the configuration. The CP stops while transmission takes place -- a small price to pay for a transfer rate more than 100 times faster than drum and 40 to 80 times faster than Large Core Storage (LCS).[3] In a sense, ECS is not an I/O device at all but an integral part of the computer, although instructions cannot be executed while residing in ECS.

Up to four devices can be attached to ECS. All can be serviced simultaneously, each getting 8 words in turn. However, when more than one device demands service, there is a delay of up to 3.2μs for each device for every 8 words -- a sizable degradation. It is tempting to make use of the spare ECS ports. For example, a data channel could be attached to one so that PP's can also access ECS. Alternately a special controller could be attached which would allow direct transfer of data between ECS and disks or some data channel to an external device. However, the relatively slow rate of PP or disk transmission does not take full advantage of the ECS port which can transmit many times faster, and thus little can be gained.

When dealing with a memory of this size, the probability of a failure is non-negligible. Consequently, there are provisions for detecting a failure, isolating the bad area through both hardware and software, and allowing the system to continue until scheduled maintenance.

Hardware storage protection exists for ECS as for CM. The bounds are set for each CP program and an attempt to reference outside the allotted area is trapped.

B. Use of ECS. A number of potential applications compete for the use of ECS. A careful study needs to be made to determine which ones deserve priority.

1. Systems Tables. The following tables, accessible to all computers sharing the ECS must reside in ECS:

a) Job Queue Table, containing information on each job within the system. Jobs can be available to any machine for either processing or post-processing (printing or punching the output).

b) File Table, containing crucial information on every active or potentially active file available to the system.

c) Table of available storage in ECS. This will be discussed in more detail under ECS allocation.

d) Table of equipment available to all systems, such as tape drives, printers, etc.

e) A communications area wherein the computers sharing ECS can communicate with each other.

2. Frequently used programs such as the compiler, sections of the operating system, and some library routines.

3. Jobs in setup or swapped out status. Normally a job will be set up in ECS before execution and returned to ECS whenever it is swapped out of CM.

While it may be more convenient to return a job to the computer which started to process it, there is no conceptual difficulty in allowing execution to resume, after swap-in on another computer.

4. I/O Buffers. Each active file has a buffer in ECS. The I/O flow is illustrated for output in Figure 2.

The size of the user's CM buffer is small. The ECS buffer is large - e.g. for files destined for the disk, multiples of a half-track's capacity (a half-track is the maximum which can be written per disk revolution) seems appropriate. When data must be transferred from ECS to an external device, it again goes through a CM buffer. One such buffer is associated with every active channel. These

buffers have a minimum length of a physical record, while the ECS buffers are some convenient multiple of this size. The channel buffers require 14K of CM, distributed as follows:

2 disk channels at 4K each	8K
2 tape channels at ½K each	1K
2 unit record channels at ½K each	1K
1 real time channel at 4K	4K

5. User Area. For certain applications it may be desirable to allow the user direct access to a portion of ECS, either via machine language code or through supplied subroutines, instead of via the system. However, an exchange jump causes an interruption in ECS transmission, which must then be restarted at a later time. (An exchange jump is executed by the monitor program to give control of the CP to a new job). If ECS usage were not channelled through the monitor, the essential unpredictability of users' programs would not allow ECS usage to be scheduled in a manner to avoid conflicts with other machines. For these reasons, permission to use ECS directly must not be given lightly. Normally, the user should write files, which may be randomly accessible in nature, and the operating system will decide on which medium these files will actually reside.

C. Allocation of ECS. Several methods of allocating ECS for each of the above uses suggest themselves.

A straightforward method is to assign a fixed block of the required size for each use as the need arises. Such a method is easy to implement. However, it requires that, periodically, everything come to a halt while a storage move is made in ECS to consolidate the unused areas. A better method is to assign ECS in "pages". Whenever a block of ECS is required, a sufficient number of pages, not necessarily contiguous, are assigned. Each page has a control word associated with it containing the following information:

- pointer to next page in string
- pointer to previous page in string
- number of words used in page, if not full.

Optionally the following information could also be recorded:

- pointer to a table giving full details on the file to which page belongs
- page count, i.e. this is the n^{th} page of the k^{th} logical record of the file
- a bit indicating that when this page is released, it is not to be reassigned. This allows the ECS allocation program to eventually free a certain amount of contiguous ECS space for a special purpose such as direct user access.

Unused pages could also be treated as a string. Thus to release a file, it is sufficient to attach it to the string of available pages. However, such a scheme makes it difficult to obtain a block of contiguous pages.

Needless to say, that portion of ECS reserved for fixed length tables, system pointers and subroutines will not be assigned by pages but by fixed blocks.

The concept of paging raises the question of an optimum page size. If the page is too small, the number of ECS accesses goes up as does the bookkeeping overhead. Make the pages too big, and storage is wasted in ECS because for each file opened the last page of a logical record is not filled.

It seems clear that certain uses require larger pages than others. For example, I/O requires small pages while swap-out larger ones. Thus it could be advantageous to divide ECS into three areas: a fixed one for tables, etc., one assigned by small pages and one assigned by large pages, preferably some multiple of the small ones. The basic page is referred to as an atom and the larger one as a molecule. A floating boundary between the atom and molecule area is desirable. This implies that allocation of pages be done in such a way as to maximize the chance of obtaining consecutive free atoms which can then be combined into a molecule.

The choice for the basic page size depends on both CM and ECS size. For machines with a small CM, an atom of 256 or 512 words seems best, for larger CM's as much as 1024.

The location of the control word is another area of decision. Should it be placed within the page or in a separate table? It turns out that if control words are grouped into a table, the system is vulnerable to a failure within that table area, whereas if the control word is within the atom, an ECS error will cause minimal damage.

D. Multi-Computer Communications. It is expected that two or more computers, sharing ECS will have a better throughput than machines which do not have the ability to distribute their workload.

A possible configuration is shown in Figure 3.

With shared ECS, all computers have access to a common job queue, a common print queue, and a pool of I/O resources.

It is not necessary for one machine to be the master and the other slaves; however it is necessary for consultation of tables in ECS to proceed in an orderly fashion. Thus when computer A modifies a table, it must be free from interference from computer B.

Access to tables is not the only problem facing a multiple computer complex. ECS transfer is severely degraded when more than one computer reads or writes simultaneously. It is feasible for the software in one computer to check when the other one is reading or writing, but this seems a good case for a hardware flag, set when ECS is in use and testable by any computer.

II. Generalization

In this section we permit a certain amount of idealization and simplification. We consider N essentially identical large scale computers which share ECS and we assume the following:

a) Access to ECS is not necessarily limited to 4 ports and its size is greater than the sum of N CM's.

b) The maximum transmission rate from ECS to CM is at minor cycle* speed and we neglect the initial access time of 3.2μs.

c) Some or all of the I/O channels may be pooled so that post-execution processing is treated as if optimized whether or not some computers are down.

d) I/O equipment is modular and can be expanded to cover the system's capacity; also it is assumed that an I/O scheduler exists which optimizes I/O equipment usage.

e) Each of the N machines is multiprogrammed (monoprogrammed machines are then just a particular case).

f) If j is the number of computers reading or writing ECS simultaneously there exists a degradation rate $R(j) > 1$ for $j \geq 2$.

g) When a CP gives a read or write ECS command, the next instruction cannot be executed until transmission is over.

h) I/O may take place independently of CP activity and the small CP degradation caused by the PP's stealing memory cycles is counted as system overhead.

i) The time it takes the monitor to start a new job via an exchange jump or to initiate a swap of jobs between CM and ECS is negligible.

---

* On the CDC 6600, a minor cycle is 100 nanoseconds.

The central questions considered are:

1) What is the advantage, if any, of an ECS configuration over one having N separate stand-alone computers?

2) How may we estimate this advantage or calculate an expected throughput based on statistics from the stand-alone system?

The following notation is used:

For the ECS system the total running time T is divided into the following fractions:

s = system overhead, i.e. the fraction of the time used by the monitor functions in the CP

t = ECS read/write time

u = useful user's CP execution time

p = idle time

The CP is hardly ever stopped so $s+t+u+p = 1$.

For the stand alone (SA) system the corresponding variables are:

σ = equivalent system overhead

ν = useful user's CP time

φ = idle time

and $\sigma+\nu+\varphi = 1$.

By the use of the word "time" we mean the dimensionless "fraction of the time" unless otherwise indicated.

The definition of σ is vague in the SA system since most monitor functions are carried on in a PP.

Throughput is defined as u in the ECS system and ν in the SA system, and the merits of the two systems are to be judged by comparing u and ν. However, if $s-\sigma+t$ is small, then comparison of the idle times p and φ also gives an estimate of the merits of each system. An ECS would only be considered if φ is high (say greater than 25%) or if analysis shows that $u \gg \nu$ or $p \ll \varphi$. Although throughput has been defined based on CP performance, the existence of an I/O scheduler should assure a corresponding I/O performance. In actual practice, the economics of ECS cost versus CM cost must be weighed against any gain or loss in expected throughput.

Let θ be the ratio of the time a job spends in I/O to the sum of its I/O time plus its time spent in or awaiting CP execution. Based upon the simplifying assumption that two or more jobs may do I/O simultaneously, then θ may be taken as the average independent probability that a job is doing I/O. We assume that for an individual job, CP and I/O activity are sequential; it is the multiprogramming which overlaps the CP execution of one job with the I/O activity of others. (In the CDC 6600, I/O is accomplished by PP's with no CP overhead.) Taking the cause of idle time to be that all jobs are doing I/O, then $\varphi = \theta^k$, where k is the number of jobs residing simultaneously in CM. In the ECS system, for a single machine $p = \theta^K$, where K is the number of jobs in CM and ECS. If N computers share the job queue, and select from among $n > N$ jobs, the total idle percentage P_1 (also called the scheme of selection P_1) will be given by:

$$P_1 = 100[N\theta^n + (N-1)\binom{n}{1}\theta^{n-1}(1-\theta) + \ldots + \binom{n}{N-1}\theta^{n-N+1}(1-\theta)^{N-1}]$$

and the average per machine will be

$$P_1^* = \frac{P_1}{N}$$

If assignment of jobs to machines is fixed, i.e. a job cannot be swapped out of one machine and resume execution in another, and there are a certain number of jobs available to each machine, let P_2 be the selection scheme when each machine has approximately an equal number of jobs available to it, $\frac{n}{N}$; and let P_3 be the case when the number of jobs are allocated unequally. Accordingly the corresponding total and average percentages are:

$$P_2 = 100(N\theta^{\frac{n}{N}}), \qquad P_2^* = \frac{P_2}{N}$$

$$P_3 = 100(\theta^{n_1}+\theta^{n_2}+\ldots+\theta^{n_N}), \quad P_3^* = \frac{P_3}{N}$$

where $\sum_{i=1}^{N} n_i = n$, and n_i jobs are allotted to machine i.

The following is then intuitively apparent and can be proven as a theorem:

$$P_1 \le P_2 \le P_3.$$

Proof: We need to demonstrate that

$$\sum_{j=0}^{N-1} (N-j)\binom{n}{j}\theta^{n-j}(1-\theta)^j \le N\theta^{\frac{n}{N}} \le \sum_{i=1}^{N} \theta^{n_i}$$

where N, n, n_i, $\frac{n}{N}$ are positive integers and $0 \le \theta \le 1$. The right hand inequality is a form of the well known relation[4] $\pi a_i^{q_i} \le \Sigma a_i q_i$ where a_i, $q_i \ge 0$, $\Sigma q_i = 1$ with the substitutions: $q_i = \frac{1}{N}$, $a_i = \theta^{n_i}$ for i = 1,...,N. For the left hand inequality, let $\frac{n}{N} = K$ and divide both sides by $N\theta^K$. We must show then that

$$\sum_{j=0}^{N-1} \left(\frac{N-j}{N}\right)\binom{n}{j}\theta^{n-j-K}(1-\theta)^j \le 1 = [\theta+(1-\theta)]^{n-K}$$

By putting

$$\frac{N-j}{N} = \left(\frac{N-j}{N-j+1}\right)\left(\frac{N-j+1}{N-j+2}\right)\cdots\left(\frac{N-1}{N}\right)$$

for j = 1,...,N-1 and

$$[\theta+(1-\theta)]^{n-K} = \theta^{n-K} + \sum_{j=1}^{n-K} \binom{n-K}{j}\theta^{n-j-K}(1-\theta)^j$$

The above inequality reduces to:

$$\sum_{j=1}^{N-1} \left(\frac{N-j}{N-j+1}\right)\left(\frac{N-j+1}{N-j+2}\right)\cdots\left(\frac{N-1}{N}\right)\binom{n}{\theta}\theta^{n-j-K}(1-\theta)^n \le \sum_{j=1}^{n-K} \binom{n-K}{j}\theta^{n-j-K}(1-\theta)^j .$$

Since n-K = K(N-1) ≥ N-1, a term by term ratio of the first N-1 terms of the left side to corresponding terms on the right side of the above gives

$$\left(1 - \frac{1}{N-j+1}\right)\left(1 - \frac{1}{N-j+2}\right)\cdots\left(1 - \frac{1}{N}\right) \cdot \left(1 + \frac{1}{\frac{n}{K}-1}\right)\left(1 + \frac{1}{\frac{n}{K}-1-\frac{1}{K}}\right)\cdots\left(1 + \frac{1}{\frac{n}{K}-1-\left(\frac{j-1}{K}\right)}\right)$$

for j = 1,...,N-1. Regrouping this product and noting that $\frac{n}{K} = N$ we have

$$\prod_{i=1}^{j} \left(1 - \frac{1}{N-i+1}\right)\left(1 + \frac{1}{N-1-\left(\frac{i-1}{K}\right)}\right) = \prod_{i=1}^{j} \left(1 + \frac{(i-1)\left(\frac{1}{K}-1\right)}{(N-i+1)\left(N-1-\left(\frac{i-1}{K}\right)\right)}\right)$$

≤ 1 for j = 1,...,N-1

which proves that $P_1 \le P_2$.

Thus P_1 is the recommended scheme in the ECS system. If P_1 is not possible then P_2 is better than P_3.

For example consider N = 2, θ = .85 (the actual BNL estimate) and let n = 16, n_1 = 3, n_2 = 13. The inequality $P_1 \le P_2 \le P_3$ becomes $2\theta^{16} + 16\theta^{15}(1-\theta) \le 2\theta^8 \le \theta^3 + \theta^{13}$ which at θ = .85 reads .36 ≤ .54 ≤ .74, i.e., 18%, 27%, 37% idle time per machine for the schemes P_1, P_2 and P_3 respectively.

Determination of θ and Estimation of ECS size

If θ is known, then a desired upper bound on the idle time determines n. If the average job size is known we know how much ECS is needed for job swapping. Then the total amount of ECS needed is the sum of swap space, I/O buffers and system space.

At BNL, θ was calculated by two methods based on statistics gathered on the SA system. If we let p_i be the independent probability that the ith job is doing I/O, then the probability of the CP being idle is given by

$$\varphi = \prod_{i=1}^{k} p_i$$

where k is the number of jobs residing in CM. We write $\varphi = \theta^k$ or $\theta = \varphi^{1/k}$. We found φ to lie between .60 and .65 and k was 2 or 3, on the average. Solving $.60 \le \theta^2 \le .65$ and $.60 \le \theta^3 \le .65$ gives us a range for θ: $.78 \le \theta \le .86$.

A second method is to obtain samples of p_i by comparing PP to PP+CP time for each job. The following table was obtained:

PP/CP ratio	r_i = proportion of jobs	p_i
0.5 - 1.5	.12	0.50
1.5 - 2.5	.09	0.60
2.5 - 3.5	.07	0.70
3.5 - 4.5	.08	0.75
4.5 - 5.5	.08	0.80
5.5 - 8.5	.21	0.85
8.5 - 14	.19	0.90
> 14	.16	0.95

The arithmetic mean is $\Sigma r_i p_i = 0.78$ and the geometric mean is $\pi p_i^{r_i} = .76$. Either figure is in good agreement with those obtained by the first method. Since optimization of compiled code tends to increase θ, we took $\theta = 0.85$ as a working estimate.

Figure 4 depicts the average idle time per computer under selection scheme P_1^* as a function of θ and n.

If our target for P_1^* is 10%, we find n = 20 for $\theta = 0.85$ and N = 2. The corresponding number of jobs for P_2^* is n = 28.

The average job length is about 15K so we can estimate the amount of ECS required for job swapping.

To estimate I/O buffer requirements we proceed as follows: a job has associated with it four files on the average; three assigned to the disk, one to a magnetic tape. Allowing 4K words for a disk file buffer and 1K for a tape file, each job requires 13K words of ECS for its I/O buffers.

Thus we can estimate ECS requirements as follows:

Job setup and swapping 20 x 15	300K
Buffers 20 x 13	260K
System tables and programs	150K
Emergency core space for each machine: 2 x 65	131K
Total	841K

Estimate of t

So far we have given answers to questions 1) and 2) of this section by means of the rough comparison of p and φ. A more accurate comparison requires an estimate for t.

The fraction of the time spent in ECS transmission, t, is a function of the multiple access degradation rate R(j), the degree to which jobs are I/O bound, and the frequency of job swapping. The latter is to some degree dependent on the first two, but it also depends on the choice of ECS buffer size and the size of CM. The greater the buffer size, the longer, in general, a job may remain in continuous CP execution. The bigger the CM, the more jobs may be multiprogrammed in the CP and the less frequent the swapping.

Consider t to be the sum $t = t_\ell + t_b + t_s$ where t_ℓ is the time needed for job swapping, t_b is the time for I/O buffer transmission, and t_s is the time for systems usage of ECS.

A sensible condition for swapping is that idle time is about to be created. If CM can hold k jobs, the probability of imminent idle time is θ^k while each mutually disjoint event of a job being in CP execution (or possibly that its share of t or s is occurring) has probability $(1-\theta^k)/k$. Suppose $\Delta\ell$ is the average uninterrupted CP execution period for one job and $k \cdot \Delta\ell$ the average uninterrupted CP period for k jobs. If Δt is the time necessary for a full CM exchange to ECS, then the ratio of swap time to CP execution time is $\frac{\Delta t}{k\Delta\ell}$. Since CP execution time is less than the total time T, this quanity may be taken as an upper bound: $t_\ell \leq \frac{\Delta t}{k\Delta\ell}$.

Given k, and Δt, a substitution of a desired percentage for t_ℓ enables one to solve for $\Delta\ell$. For example, suppose k = 2, Δt = 12 ms. and we want $t_\ell \leq 2\%$. Then $\Delta\ell$ = 300 ms.

The simplest way to find a lower bound on ECS I/O buffer size is to find the average execution period in the SA system, $\Delta\ell_{SA}$, and multiply by the buffer sizes in SA by $\Delta\ell/\Delta\ell_{SA}$. It was found at BNL that there were approximately 3 exchange jumps per second. Each second accounts for about 350 ms. of non-idle time which gives $\Delta\ell_{SA} \approx$ 120 ms. Comparison with $\Delta\ell$ as obtained above, shows that ECS buffers should be about three times as large as in SA.

It should be noted that conditions other than imminent idle time in a CP can cause a swap. The most common of these might be a real-time job which must have CP time with minimum of delay. Any scheduling scheme based on priorities will probably have such a "real-time" category. Hence an over-estimate of ECS I/O buffer sizes will tend to offset the additional swap time caused by real-time interrupts. However, a system which deals almost entirely with real-time interruptions, especially where access and response time are small compared to a desired $\Delta\ell$, may find it impossible to satisfy a desired $\Delta\ell$. Such a system will have to accept a larger percentage for t_ℓ. Systems which specialize in conversational mode (such as line by line or character by character on-line debugging) or which use execution page less than the entire job size will have this problem. The overhead is less severe with an ECS-CM swap than it would be with a slower swap device such as drum or disk (see below for such a comparison).

It should be noted that a swap may be selective, that is, not all of CM need be swapped. Also it may be that θ is so small and k so large that it pays to wait some period of time less than Δt until a job already residing in CM can resume execution rather than initiate a swap immediately. In this case the average idle-causing I/O time period corresponding to $k\Delta\ell$ is $(\frac{\theta^k}{1-\theta^k})k\Delta\ell$. Hence if

$$(\frac{\theta^k}{1-\theta^k})k\Delta\ell < \Delta t ,$$

it is advantageous not to swap.

The best method to obtain t_b is to measure the total number of I/O buffers used by the average job in SA. If statistics on the number of buffers filled or emptied in SA is not available, the following estimation provides an upper bound for t_b. Calculate an upper bound on buffer I/O rates by assuming peak activity on several channels. That is, assume that tapes, disks, card readers are going at maximum rates. Then from this, estimate the total buffer space required. If the actual channel activity is known, a more accurate estimate is possible. For example at BNL we assume that two tapes and two disks per machine transmitting at their maximum rates would give an over-estimate of I/O activity. This generates a 60 bit word every 8μs. and necessitates 2 ECS transfers of 0.2μs. or about 2.5%. This is certainly an over-estimate but takes into account an eventual real time channel whose maximum rate is 1 word every 5μs.

An estimate for t_s can only be made in a hand waving fashion. It seems that systems functions such as table references or other procedures requiring the use of ECS should take considerably less time than t_ℓ or t_b.

If t_s is half as large as either t_ℓ or t_b then $t \leq 5/4(t_\ell+t_b)$, that is t should be less than 25% larger than the time required for both I/O buffering and swap time.

So far the estimation of t has neglected the ECS multiaccess problem: R(j) can increase t appreciably when $j > 1$. Let t_0 be the portion of total running time T that a single computer would use ECS (with no degradation). The total expected ECS fraction of the time, ECS_t, needed to service all N machines is thus the weighted sum:

$$ECS_t = \frac{Nt_0[R(N)t_0^N+\ldots+R(1)\binom{N}{N-1}t_0(1-t_0)^{N-1}]}{1-(1-t_0)^N}$$

For example, suppose we consider a system with many real time demands and $t_0 = .20$. Let us use the actual $N = 4$ and $R(1) = 1$, $R(2) = 2$, $R(3) = 2.7$, $R(4) = 4$. Then $ECS_t = 1.07$ which is more than 100% of the original time T. Thus the following design objective is recommended: either the ECS be constructed

so that R(j) ≈ 1 for all j, or there must be a lockout option to prevent more than one CP from referencing ECS simultaneously. Software flags in ECS may be used, but a neater solution is to have a hardware flag set to allow automatic lockout of one CP while another is transmitting. Under such a provision in the above example, ECS would be busy 80% of the time. The lockout provision is especially important where an ECS type memory is not limited to four ports or to two million words so that N may be large.

Since ECS is shared, it should be noted that t may not exceed $\frac{100}{N}$ percent in each machine when the lockout provision holds. We use the convention then that ECS is busy Nt percent of the time. If as in the above example a given quantity of CP execution is expected to generate a certain percentage for t and it turns out that Nt > 100%, we then normalize in such a way that Nt = 100 and the additional time is added to the idle time (as would be the case for any request for a busy I/O device). As an extreme illustration consider N = 100 and each machine's estimate is that for 80 minutes of CP execution there is 10 minutes overhead and 10 minutes ECS time. To say u = 80%, t = 10%, s = 10%, p = 0 is incorrect. In this example 1000 minutes must pass before 80 minutes of CP time can be used. The "normalized" percentages would be u = 8%, t = 1%, s = 1%, p = 90%.

Comparison to other systems

The simple substitution of a slower random access memory device such as drum, disk or LCS for ECS produces dramatic changes in the behavior of the system. Consider the illustration of the BNL estimates: s = 10%, t = 7%, p = 8%, u = 75%. Now for ECS substitute a device 100 times as slow (10μs./60 bit word), the relative times are then s = 10, t = 700, p = 8, u = 75 or s = 1%, t = 88%, p = 1%, u = 10%. If the memory device is shared between two computers, the normalized percentages are s = 0.7%, t = 50%, p = 44%, u = 5.3%.

If a system designed to service many conversational mode users the swap rate will be high and the execution slice $\Delta\ell$, will be small, with t probably higher than 20%. When this figure is multiplied by 100 and even if CP activity does not cease during swapping or I/O buffering and even if CP idle time is nil, then u is still limited to about 5% which implies that 95% of the time is spent passing data between CM and the slower random memory. This is precisely what is largely responsible for the failures of some of the larger time shared systems. It is the hope that the speed of ECS will allow such systems to operate successfully.

Acknowledgements

This paper owes much to discussions with our colleagues at BNL, in particular, G. H. Campbell, J. E. Denes, D. A. Ravenhall, and Y. Shimamoto, and talks with certain Control Data Corporation personnel, in particular, L. Sleizer.

References

1. Jallen, Gale A., "Extended Core Storage for the Control Data 64-6600 Systems" in AFIPS Conference Proceedings, Vol. 30, 1967. (Spring Joint Computer Conference) p. 729.

2. MacDougall, M. H., "Simulation of an ECS-based operating system", ibid. p. 735.

3. Humphrey, T. A., Large Core Storage Utilization in Theory and in Practice. ibid. p. 719.

4. Hardy, G. S., Littlewood, J. E., and Polya, G., "Inequalities", Cambridge, 1934.

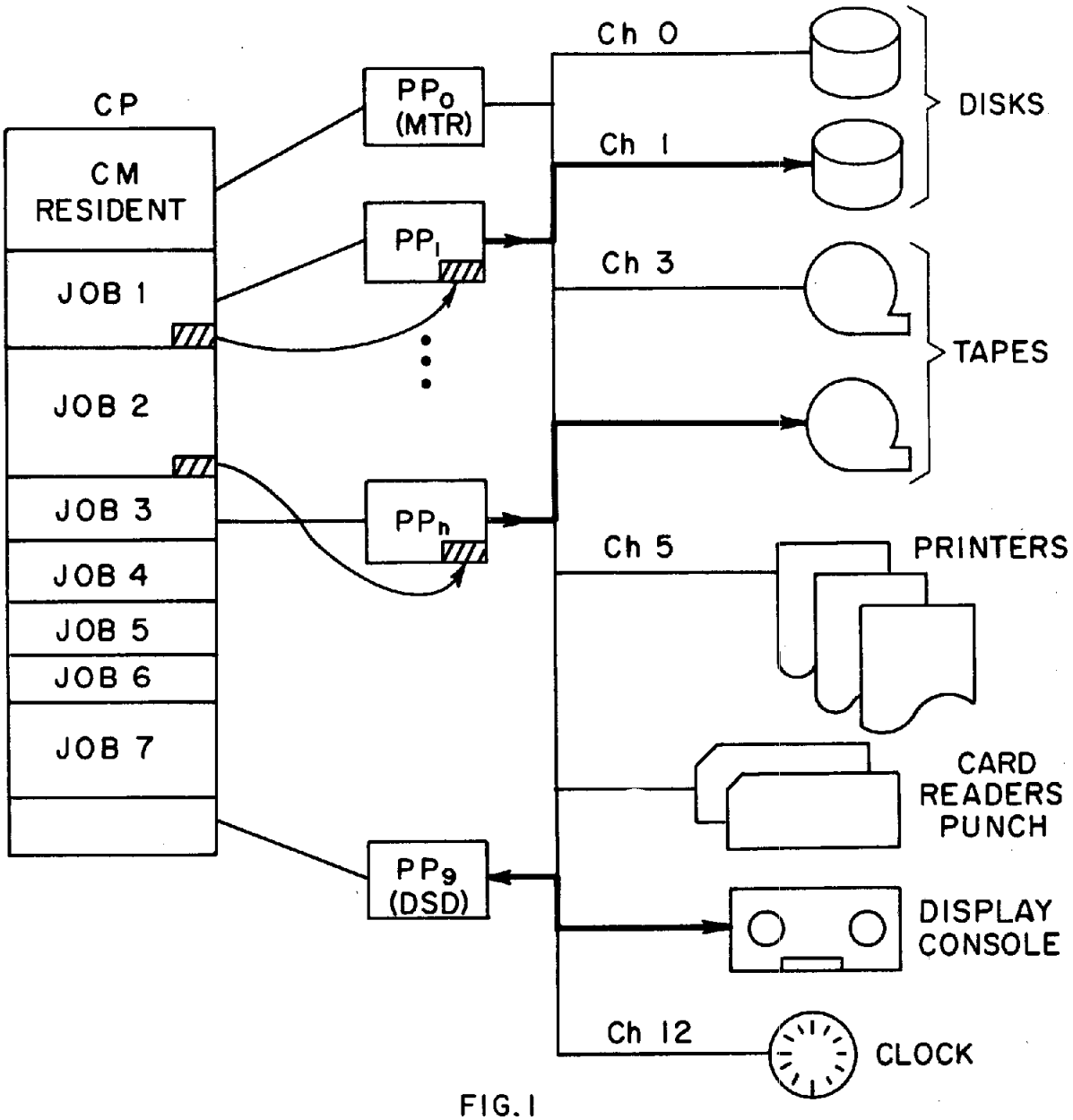

FIG. 1
TYPICAL CDC 6600 CONFIGURATION ILLUSTRATING
OUTPUT FLOW

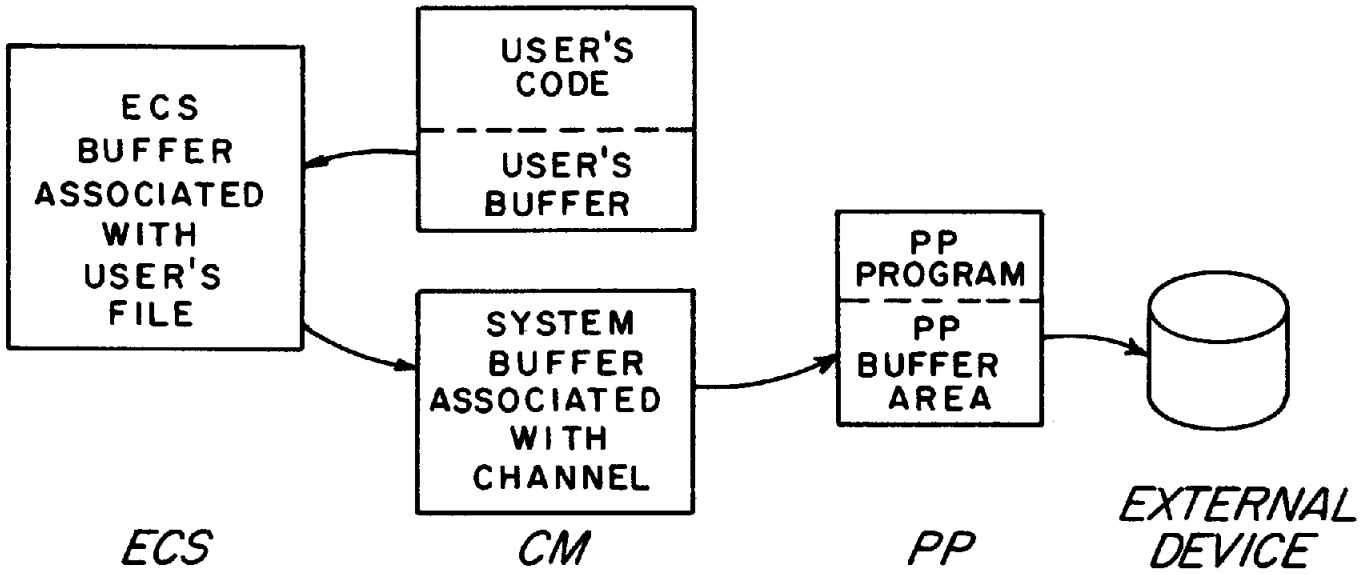

FIG. 2. I/O FLOW FOR OUTPUT

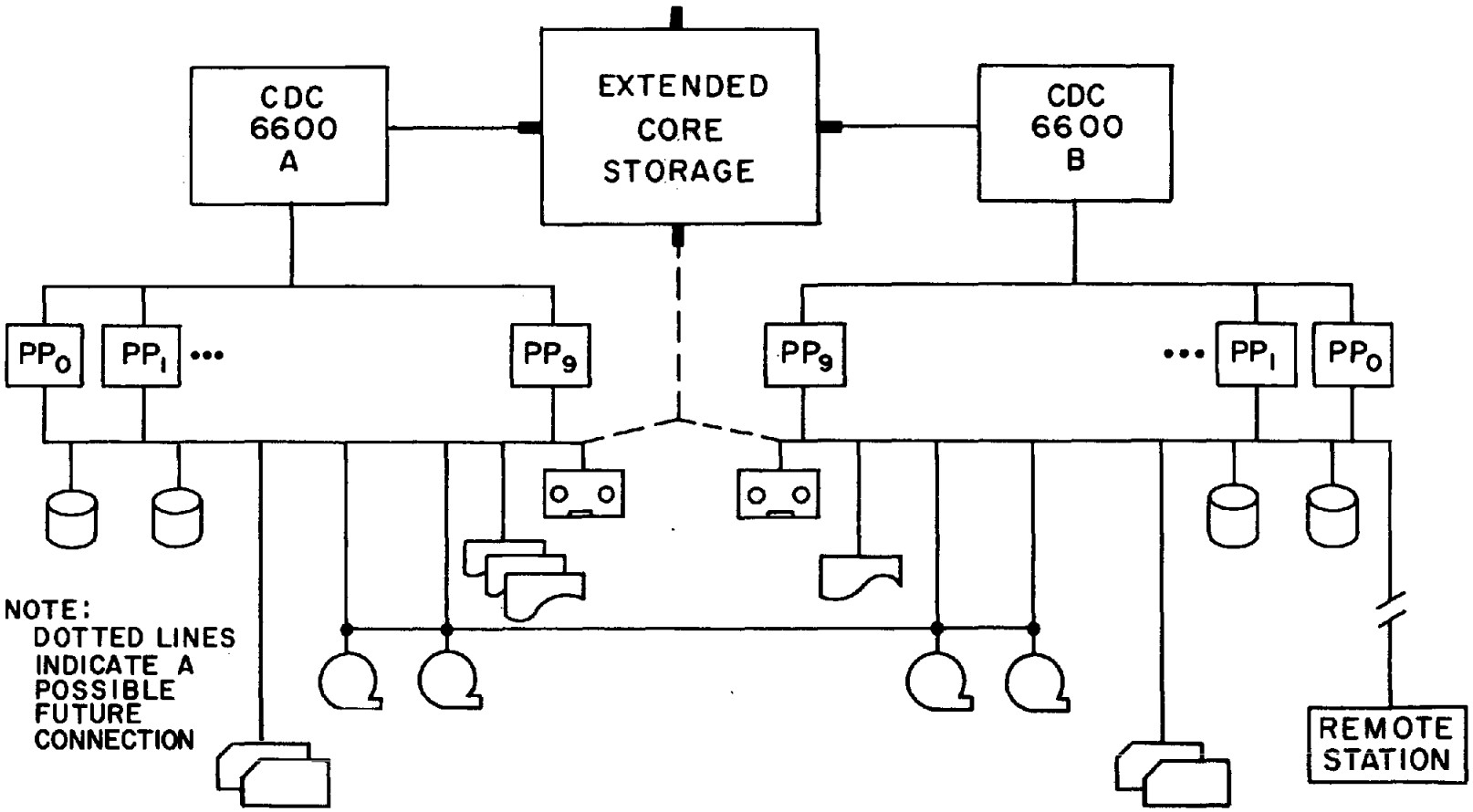

FIG.3
DUAL CDC 6600'S WITH SHARED EXTENDED
CORE STORAGE

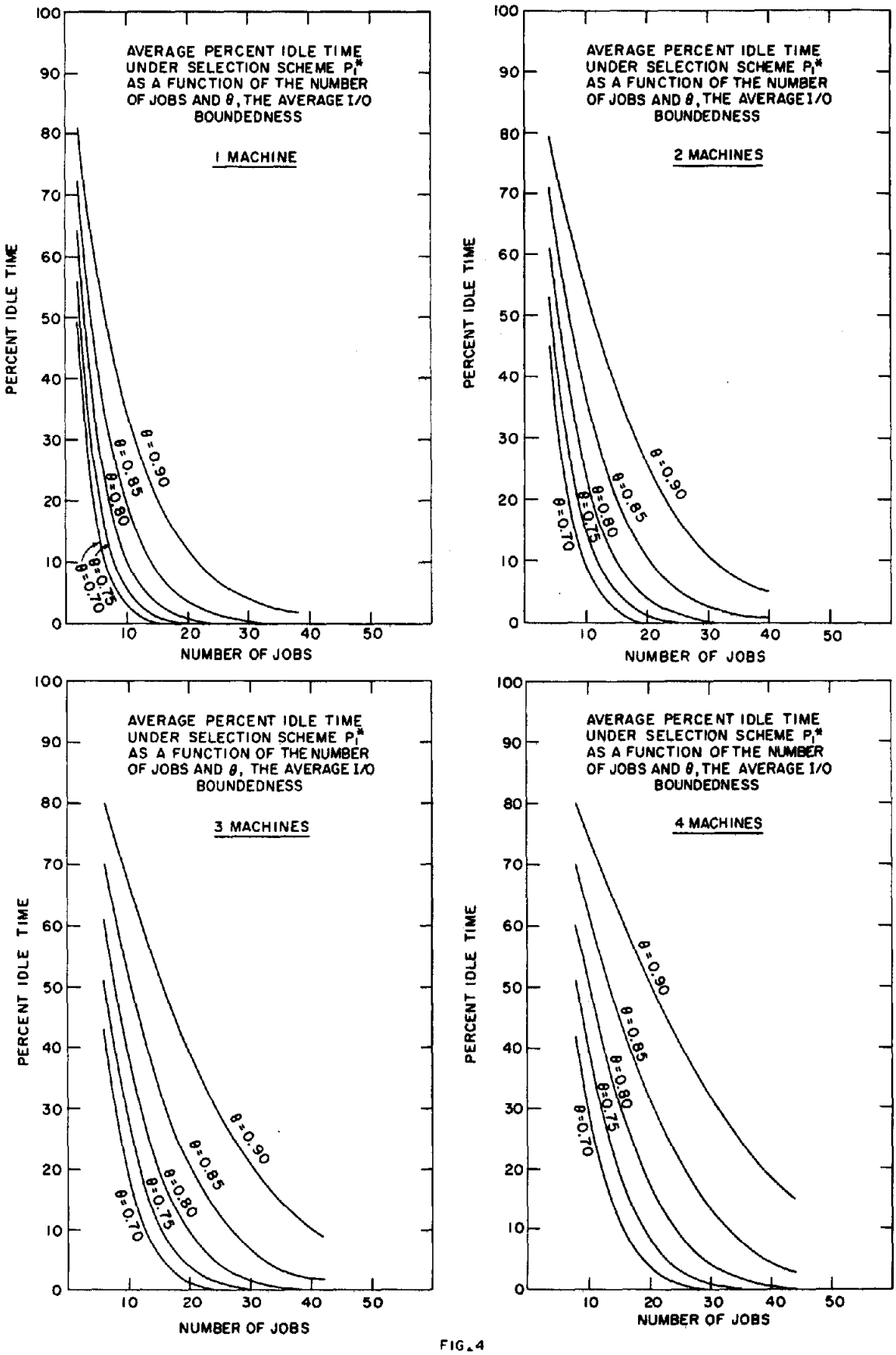
AVERAGE PERCENT IDLE TIME UNDER SELECTION SCHEME P_1^* AS A FUNCTION OF THE NUMBER OF JOBS AND θ, THE AVERAGE I/O BOUNDEDNESS
1 MACHINE
2 MACHINES
3 MACHINES
4 MACHINES
PERCENT IDLE TIME
NUMBER OF JOBS
θ=0.90
θ=0.85
θ=0.80
θ=0.75
θ=0.70

FIG. 4