We have found that the best form of presentation for management is graphic in nature. Generally, graphs present considerable information in a concise manner. Tabular reports have been aggresively avoided. We believe this approach offers superior final products, with higher degrees of acceptability to management. The basic graph to management presents some dependent variable, such as utilization or response time, graphed against transaction rate. With many dependent variables to be graphed, management will find some continuity in being able to find the faithful transaction rate on the independent axis. Find what suits your immediate management, and use it to your advantage.

Acknowledgements.

The author wishes to gratefully acknowledge the following individuals for their continued support during the modeling studies: Raul Araujo, Dave Brown, Al Gonda, Fran Hildebrand, Tim Jones, Arlie Lamb, and Mike McElwee, all of Pacific Telephone.

SHARE

SHARE SESSION REPORT

SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE
61	A085	A Research Queuing Package (RESQ) Model of a Transaction Processing System with DASD CACHE	45

PROJECT	SESSION CHAIRMAN	INST. CODE
Modeling and Analysis	Bruce Hibbard	PJE

SESSION CHAIRMAN'S COMPANY, ADDRESS, AND PHONE NUMBER

Project Software and Development, Inc., 14 Story Street, Cambride, MA 02138
617-661-1444, Ext. 214

A Research Queueing Package (RESQ) Model of A Transaction Processing System with DASD Cache

G. A. Marazas
IBM Corporation
P. O. Box 1328
Boca Raton, FL 33431

Alvin M. Blum and Edward A. MacNair
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

SHARE Installation Code: IBM

SHARE Project: Modelling and Analysis

SHARE Session Number: A085

Abstract: The Research Queueing Package (RESQ) is a tool for constructing and solving models of contention systems. A contention system is a collection of interconnected resources and jobs which demand service from these resources. Examples of contention systems are computer systems, communication networks, manufacturing systems, office systems and distributed systems. We first illustrate the basic facilities available in RESQ for representing such systems and provide a simple example in order to illustrate their use.

Next we describe how RESQ has been used as an analysis tool to assist in the development of the disk cache portion of the IBM 4967 disk control unit for the IBM Series/1 computer system. The discussion here has wider application because the same design problems considered for the 4967 will also occur in one form or another in disk controllers connected to systems ranging in size from the Personal Computer to the top of the line MVS and VM systems. Also, programming design has a need similar to hardware design as to modeling and understanding sequence relationships and overlap in a complex system with many process steps. Based on such modeling experience, it is the authors' opinion that the RESQ approach involving a network of queues and the facility of passive queues is very well suited for investigation of many design issues associated with development of hardware as well as both operating systems and applications programming.

4/Y/LEJ/1

CONTENTS

1. INTRODUCTION

The continuing use of computer modeling techniques [2,4] has led to a need for a comprehensive set of tools for representing systems which are becoming increasingly complex. The Research Queueing Package (RESQ) [8-12] had its genesis in the solution of simple (by current standards) queueing networks that were not feasible by manual methods. As new numerical algorithms (e.g., Mean Value Analysis [3]) and simulation constructs (e.g., "passive" queues [4]) have been developed, they have been implemented in RESQ to extend its application to more general types of systems and to enhance its computational efficiency. In order to familiarize the reader with RESQ concepts and notation, this section describes how RESQ is used in modeling systems. In the remaining sections, we discuss the use of RESQ as an analysis tool to assist in the development of the disk cache portion of the IBM 4967 disk control unit for the IBM Series/1 computer system.

(A). THE RESEARCH QUEUEING PACKAGE (RESQ)

The Research Queueing Package (RESQ) is a set of programs for the construction, analysis, and solution of models of contention systems. Contention systems are collections of interconnected resources through which basic entities (jobs) flow and demand service. RESQ provides extensive facilities for representing both the resource network and the behavior of jobs within that network. Models can be solved by simulation, or where the model structure, parameters, and assumptions permit, by one of several known analytical methods. Currently under development are approximation techniques that will offer numerical solution methods for a larger set of system configurations.

In RESQ, system resources are represented by "queues." Queues are grouped into two general categories, "active" and "passive." Jobs in active queues cannot interact with other model elements while remaining in the queue; those in passive queues can simultaneously occupy other system resources and perform other activities. Active queues are further subdivided into types based upon their service disciplines: first come first served (FCFS), last come first served (LCFS), priority (PRTY), preemptive priority (PRTYPR), infinite server (IS), processor shared (PS), and a generic queue type called ACTIVE. This last type permits representation of active queues (such as multiple servers with different rates) for which there is no predefined RESQ queue entity. Associated with active queues are one or more "classes" (waiting lines) which may be used to distinguish jobs with different service time distributions and, where appropriate, priorities.

Passive queues are used to model concurrency of events. Each passive queue has an associated "token pool." When jobs arrive at a passive

queue, the RESQ program attempts to allocate a user-specified number of tokens from the associated token pool. If the pool does not contain the required tokens, the job is delayed until they become available. Passive queues are often used for modeling contention for shared resources such as computer memory, input/output channels, controllers, etc. Response time measurements and the representation of communication protocols are two additional typical uses for passive queues.

The structure of a model is defined by linking resources together by means of "chains." Chains designate the permissible paths over which jobs can be routed. Chains are either "closed" or "open," and also either "external" or "internal." Closed chains have a fixed "population" of jobs which circulate in the chain for the entire model execution time. In general, open chains contain a variable number of jobs that are generated at input "sources" to the chain and terminated at "sinks." External and internal chains are used with "submodels," which are parameterized templates of user defined subsystems. External chains are connected to chains within the exterior model (or submodel) that invokes the submodel. Internal chains on the other hand are contained entirely within a submodel. Both external and internal chains can be either open or closed; however, an external chain is open (closed) if the chain to which it is connected is open (closed).

Several RESQ entities are provided to add flexibility in modeling alternative timings, configurations, job routing, etc. "Numeric" and "distribution parameters" can be modified before each solution of a model to test the effect of variations in arrival and service rates and distributions, number of users, memory size, etc. "Job variables" are associated with each job to define its behavioral characteristics, such as its "type", memory and service requirements, and routing decisions. "Chain variables" can be identified with each chain and are accessible only to jobs within that chain. "Global variables" may be accessed from anywhere within their defined (model or submodel) scopes to allow for communication between jobs and other RESQ entities. Job, chain, and global variables are assigned values at "set" nodes by "expressions" with rules closely resembling those found in most programming languages. In RESQ simulation models, expressions containing job, chain, and global variables, can be used to model status dependent assignments and decisions.

Some other RESQ entities are "split", "fission," and "fusion" nodes which permit the splitting of jobs into copies, another method for modeling simultaneity as well as signaling between, and sychronization of, processes. The split and fission nodes create additional copies of a job passing through the node; a fusion node joins those jobs that have previously separated at a fisson node.

A variety of measures are available for evaluating system performance. These include resource utilization and throughput, mean queue length, queue length distribution, mean queueing time, and queueing time distribution. Similar output is also available for the usage of passive queue tokens. For numerically (as opposed to simulation) solved models, only utilization, throughput, mean queue length, and mean queueing time are provided.

Since simulation is equivalent to a statistical sampling experiment, RESQ makes available three methods for generating confidence intervals for performance measurements. Under the "independent replications" method, the simulation is executed several (user specified) times with different sequences of random numbers. An "initialization" period may be specified for each replication; data collected during this period will be discarded when calculating output statistics. The effect of transient results may thus be minimized, permitting model equilibrium to be better approximated. A confidence interval with a (user) specified confidence level is then calculated after all replications have been completed. The "spectral" confidence interval method [1] uses the correlation properties of sequential values of the measured parameter instead of relying soley upon the assumption of independence. It too provides options for specifying a confidence level, run limits, and initialization period, as well as a sequential stopping rule whereby the user designates a confidence interval width criterion for terminating the simulation. The "regenerative" method [2,4] has some very special requirements for its use. A state in the model must be identified at which the system "regenerates." That is, the future behavior of the system is independent of all states prior to entrance into the regeneration state. The regenerative method provides all the options of the spectral method except for initialization period specification.

The next section describes a simple mutiprogramming computer system model illustrating some of the RESQ facilities mentioned above.

(B). A RESQ EXAMPLE

We will describe a simple model of an interactive computer system to illustrate some of the features of RESQ. Figures 1.1 and 1.2 are diagrams of the model. The model is structured hierarchically with a submodel representing the computer system with its memory, CPU and two input/output devices. The terminals are defined at the highest level of the model.

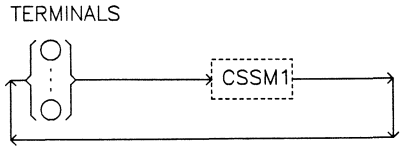

Figure 1.1. Terminals and submodel

Commands submitted from the terminals go through a set node to determine the type of command which was submitted. Each command requests a certain number of memory page frames at allocate node GETMEMORY. If the request can be satisfied, the command is permitted to enter the CPU-I/O

subsystem. Otherwise the command will wait until there is a sufficient number of page frames. The command then cycles between the CPU and one of the I/O devices. The number of cycles is determined by the type of command. When the command is finished, it releases its allocation of memory page frames at release node FREEMEMORY and sends a response back to the terminals.

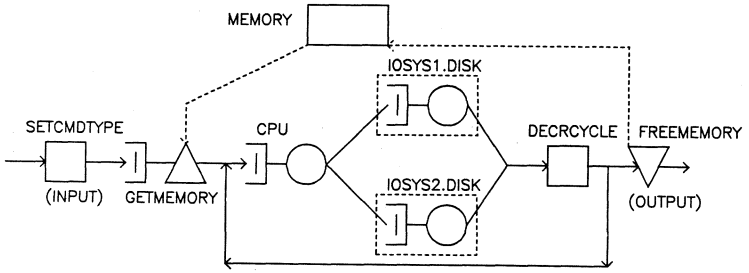

Figure 1.2. Computer system submodel

First we will describe the contents of the model with the submodel removed. The following is a listing of the model.

```
MODEL:CSM
   METHOD:simulation
   NUMERIC PARAMETERS:thinktime users
   NUMERIC IDENTIFIERS:userframes
      USERFRAMES:50
   MAX JV:1 /*0: command type, 1: cycle count*/
   QUEUE:terminalsq
      TYPE:is
      CLASS LIST:terminals
         SERVICE TIMES:thinktime
   INCLUDE:cssm
   INVOCATION:cssm1
      TYPE:cssm
      PAGEFRAMES:userframes
      INTERACTIV:interactiv
   CHAIN:interactiv
      TYPE:closed
      POPULATION:users
      :terminals->cssm1.input
      :cssm1.output->terminals
   QUEUES FOR QUEUEING TIME DIST:cssm1.memory
      VALUES:1 2 3 4 5 6 7 8
   QUEUES FOR QUEUE LENGTH DIST:cssm1.memory
   MAX VALUE:users
   CONFIDENCE INTERVAL METHOD:spectral
   INITIAL STATE DEFINITION -
```



```
   CHAIN:interactiv
   NODE LIST:terminals
      INIT POP:users
   CONFIDENCE LEVEL:90
   SEQUENTIAL STOPPING RULE:yes
   CONFIDENCE INTERVAL QUEUES:terminalsq cssm1.memory cssm1.cpuq
         MEASURES:qt
         ALLOWED WIDTHS:10
   CONFIDENCE INTERVAL QUEUES:cssm1.iosys1.diskq cssm1.iosys2.diskq
         MEASURES:qt
         ALLOWED WIDTHS:10
   INITIAL PORTION DISCARDED:10 /*percent*/
   INITIAL PERIOD LIMITS -
      SIMULATED TIME:3600
      EVENTS:50000
      QUEUES FOR DEPARTURE COUNTS:cssm1.memory
         DEPARTURES:500
   LIMIT - CP SECONDS:5
   TRACE:no
END
```

The name of the model is CSM. Simulation will be used as the solution method. Two numeric parameters are defined: THINKTIME and USERS. Numeric parameters are variables which will receive values when the model is evaluated. THINKTIME will be the service time at the terminals. USERS will be the number of terminals. A numeric identifier called USERFRAMES is defined to have a value of 50; this will be used as the total number of memory page frames. Each job will have two job variables: JV(0) will indicate the type of command, and JV(1) will be a counter indicating the number of times a command cycles through the CPU-I/O subsystem. The TERMINALSQ is an infinite server queue with the think time being a numeric parameter.

The definition of the computer system submodel CSSM is retrieved by using the INCLUDE facility. This facility permits a submodel definition to be retrieved from a library and logically inserted into the model definition. A copy of the submodel is created with the CSSM1 invocation. The two submodel parameters, PAGEFRAMES and INTERACTIV, are assigned values USERFRAMES and INTERACTIV. A closed chain is defined with a population equal to the numeric parameter USERS. The routing is from the terminals to the submodel input, and from the output of the submodel back to the terminals; routing within the submodel is contained in the submodel defintion. The model also requests RESQ to collect statistics for the queueing time distribution and the queue length distribution at the getmemory queue (within the submodel) in addition to collecting the other standard statistics.

The remainder of the model specifies information related to the confidence intervals and the length of the simulation run. We are using the spectral method [1] for generating the confidence intervals. All of the users will be initialized at the terminals. We will use 90 percent as the confidence level. The sequential stopping rule is employed to determine when the desired level of accuracy has been attained. Confidence inter-

vals for the mean queueing times at the specified queues will be constructed by RESQ. When the confidence interval width divided by the point estimate is less than 10 percent, the simulation program will stop. 10 percent of the initial portion of the run will be discarded. This could represent a transient portion of the simulation. The initial period for the sequential stopping rule will be when 3600 time units have elapsed, 50000 events, or 500 departures from the MEMORY queue.

```
SUBMODEL:cssm /*Computer System Submodel*/
   NUMERIC PARAMETERS:pageframes
   CHAIN PARAMETERS:interactiv
   NUMERIC IDENTIFIERS:cmdtype cyclecount
      CMDTYPE:0 /*JV(0) to be used to indicate command type*/
      CYCLECOUNT:1 /*JV(1) to be used to count CPU-I/O cycles*/
   NUMERIC IDENTIFIERS:cpiocycles(3) pageneed(3)
      CPIOCYCLES: 8 15 50
      PAGENEED:  20 24 30
   NUMERIC IDENTIFIERS:cputime
      CPUTIME:.025 /*mean time in seconds*/
   QUEUE:memory
      TYPE:passive
      TOKENS:pageframes
      DSPL:fcfs
      ALLOCATE NODE LIST:getmemory
         NUMBERS OF TOKENS TO ALLOCATE:pageneed(jv(cmdtype))
      RELEASE NODE LIST:freememory
   QUEUE:cpuq
      TYPE:ps
      CLASS LIST:cpu
         SERVICE TIMES:cputime
   SET NODES:setcmdtype
      ASSIGNMENT LIST:jv(cmdtype)=discrete(1,.8;2,.15;3,.05),++
         jv(cyclecount)=cpiocycles(jv(cmdtype))
   SET NODES:decrcycles
      ASSIGNMENT LIST:jv(cyclecount)=jv(cyclecount)-1
   INCLUDE:iosys
   INVOCATION:iosys1
      TYPE:iosys
      INTERACTIV:interactiv
   INVOCATION:iosys2
      TYPE:iosys
      INTERACTIV:interactiv
   CHAIN:interactiv
      TYPE:external
      INPUT:setcmdtype
      OUTPUT:freememory
      :setcmdtype->getmemory->cpu->iosys1.input iosys2.input
      :iosys1.output iosys2.output->decrcycles
      :decrcycles->cpu freememory;if(jv(cyclecount)>0) if(t)
END OF SUBMODEL CSSM
```


The CSSM submodel has a numeric parameter, PAGEFRAMES, which is the total number of memory page frames. A chain parameter is defined in the submodel to connect the chain in the submodel to the chain in the model. The numeric identifiers CMDTYPE and CYCLECOUNT will be used as mnemonic subscripts for the job variables. The numeric identifiers CPCIOCYCLES and PAGENEED are vectors; their values are the number of CPU-I/O cycles and the number of pages required by the three different types of commands. CPUTIME is a numeric identifier representing the average amount of time a command spends running on the CPU during each visit.

Next we have the queue definitions. The memory queue is a passive queue with PAGEFRAMES equal to the number of tokens. The number of tokens to allocate to a command will be selected from PAGENEED according to the type of command. The CPU queue has the processor sharing queueing discipline. At SETCMDTYPE, the job variables representing the command type and the number of CPU-I/O cycles are assigned values. Each time the command finishes a cycle, the job variable representing the cycle count is decremented by one at DECRCYLES. We INCLUDE a submodel for an I/O device. Notice that this submodel is nested within the CSSM submodel. Two copies of the IOSYS submodel are created with the two invocations IOSYS1 and IOSYS2. The only parameter is a chain parameter.

The chain in CSSM is an external chain because it is be connected to the chain defined in the model. Input and output nodes are defined to indicate the nodes at which jobs enter and leave the submodel. A command which enters the submodel starts at SETCMDTYPE. After setting the command type and the cycle count, a request is made at the allocate node, GETMEMORY, to allocate page frames. When a sufficient number of page frames are available, the command cycles between the CPU and an I/O device. When a job leaves the I/O submodel, the number of remaining cycles is decremented by one, and the job goes back to the CPU if the number of cycles if greater than zero. Otherwise, the job releases its page frames and leaves the submodel.

```
SUBMODEL:iosys
   CHAIN PARAMETERS:interactiv
   QUEUE:diskq
      TYPE:fcfs
      CLASS LIST:disk
         SERVICE TIMES:.06
   CHAIN:interactiv
      TYPE:external
      INPUT:disk
      OUTPUT:disk
END OF SUBMODEL IOSYS
```

The IOSYS submodel consists of only an active queue for a disk device. It has a chain parameter because the chain in this submodel is connected to the chain defined in the CSSM submodel. Since the DISK class is the input and output node for this submodel, no routing statements are necessary.

2. A MODEL OF A TRANSACTION PROCESSING SYSTEM WITH DASD CACHE

The question of consistency in system level response is used as the vehicle for demonstrating how the facilities provided by RESQ are used to good advantage. In particular, the consistency in response time is affected by the operation of DASD cache in the presence of application determined parameters such as cache hit ratio, the ratio of disk reads to disk writes, and the degree of overlap between CPU service and disk service. Additionally, the consistency of systems level response time is influenced by the design of the cache management algorithms as well as the fact that the cache control and cache buffer have been located in the disk control unit. The intent of this paper is to emphasize the approach taken in the development of the model rather than any specific results determined from exercise of the model.

It must be noted that any performance data referenced herein was determined in a controlled environment, and therefore, the results which may be obtained in other operating environments may vary significantly. Users of this paper should verify the applicable data for their specific environment. It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming or services that are not announced in your country. Such reference or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

3. DESCRIPTION OF TRANSACTION PROCESSING SYSTEM

Figure 3.1 provides a summary in RESQ terminology of the transaction processing which is the subject of this paper. For purposes of being specific, the IBM Series/1 is used as the example system. During the course of the following discussion, selected portions of the systems such as the disk controller and cache buffer are identified as the focal points for design issues. These selected portions are then modeled in additional detail so as to reflect the major hardware and programming interactions which are judged to be important. Other portions of the system such as the operating system and central processor unit (CPU) are not modeled in any further detail because they are not involved in the design questions being investigated.

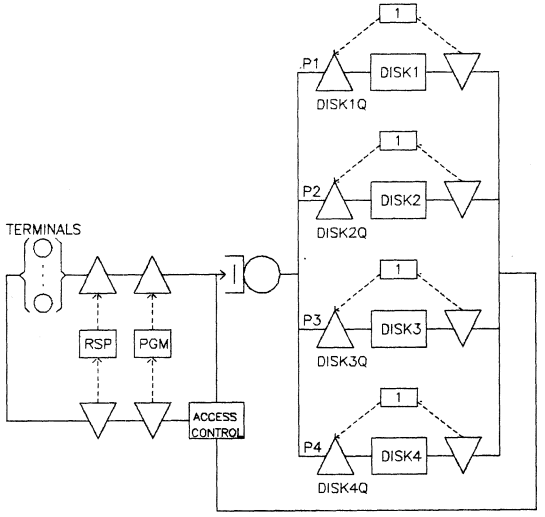

Figure 3.1. High Level Simulation Model of Transaction Processing System

As shown in Figure 3.1, transactions originate from a set of connected terminals. The system level response time is measured using a passive queue with a large supply of tokens such as 10,000. The response time interval begins with the start of a service request message from a given terminal directed to the CPU. The end of the response time interval occurs when the reply message is sent from the CPU back to the originating

terminal. For purposes of the present investigation, it is not necessary to model the length of either the service request message or the reply message. Neither is it necessary to consider the speed of the transmission line or the communications protocol connecting the terminal to the CPU. Our interest is focused on the amount of CPU and disk service performed in response to the request portion of the transaction.

A second passive queue controls the number of transactions (the multiprogram level) which may be concurrently present in the main memory attached to the central processor unit. In the case of maximum interest for consistency of response time, the multiprogram level should be in the range of 50 to 100 larger than the number of attached disk units. Additionally, the number of attached terminals and terminal think time should be selected so as to ensure the number of transactions active in the CPU are near the multiprogram level. In this manner, the disk control unit will have a high prospect for overlapped operation relative to its attached disk units.

Figure 3.1 shows one passive queue for each of the four attached disk units. A larger number of disk units and controllers are capable of being connected to the Series/1 but the number shown in Figure 3.1 is sufficient for this paper. For each passive queue, the token pool equals one. In this manner, it is assured that one and only one access at a time is directed to any given disk unit. A disk unit not presently occupied with an access may accept a new access. This represents the actual state of affairs in which queueing for individual disks is maintained within the CPU. The probabilities P1, P2, P3, and P4 reflect the probabilities of an access to the given disk units one through four. To represent the well known phenomenon of unequal accessing of disk units, (also known as disk skew), the probability P1=0.4 and the probabilities P2, P3, and P4 are each 0.2.

The access control mechanism is provided to ensure that an exact number of disk accesses are involved in the service of each transaction. In this manner, the amount of disk burden is removed as a variable influencing the consistency of system response time. For the study reported here, a relatively heavy disk burden of 20 accesses per transaction is used.

4. OPERATION OF DISK CACHE SUBSYSTEM

Figure 4.1 illustrates the four possibilities relative to an individual access in a system with disk cache implemented in the disk control unit.

Read Hit: Transfer Data Amount 1 From Cache Buffer to the CPU

Read Miss: (a). Read Data Amount 2 From Disk Into the Cache Buffer
(b). Transfer Data Amount 1 From Cacge Buffer to the CPU

Write Hit: (a). Transfer Data Amount 1 From CPU to Disk
(b). Optionally, Update the Cache Buffer

Write Miss: Transfer Data Amount 1 From CPU to the Disk

Figure 4.1. Process Responses for the Four Possibilities of an Access in a Disk Cache --- Cache Located in Disk Controller

---

The terminology of read/write reflects the perspective of the disk. Thus, a disk read involves a transfer of data from the disk to the CPU; and a disk write involves the transfer of data from the CPU to the disk. A hit is the case in which the referenced data is presently in the cache buffer. A miss is the case in which the data is not in the cache buffer. A given disk access is either a hit or a miss; with the result that the probability of a hit and the probability of a miss add up to one.

The primary information conveyed by Figure 4.1 is the response of the Series/1 IBM 4967 disk cache control system to the four cases of read hit through write miss. The cache system and its response pattern is modeled in RESQ terms as illustrated in Figure 4.2.

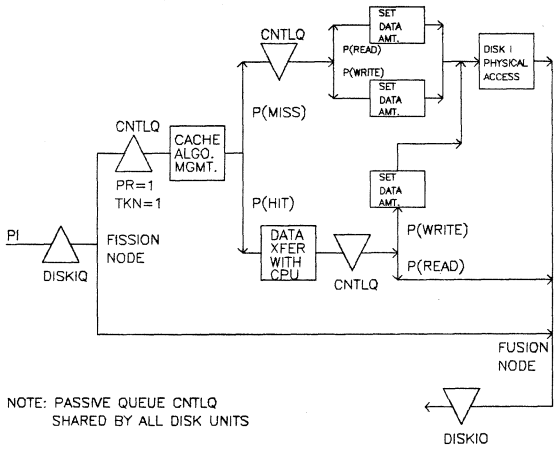

Figure 4.2. Cache Control Portion of Disk Unit I Model

Before discussing the modeling of the cache process sequence, the reader's attention is directed to the relation between Figure 3.1 and Figure 4.2. Specifically, Figure 4.2 is an expansion in detail relative to cache control functions for a single disk unit shown in Figure 3.1. Thus, the structure shown in Figure 4.2 is repeated four times in the system model, once for each disk unit connected. In a later section, there will be provided a still further expansion in detail as to the service time associated with a physical access to a disk unit.

(A). HIGH-LEVEL MODEL OF DISK CACHE

The process flow depicted in Figure 4.2 is as follows. A task cannot enter a disk unit until it acquires the single token for that queue. For DISK1, the entrance controlling passive queue is called DISKIQ. As stated earlier, this passive queue guarantees that only one task at a time is in service for any given disk.

A special feature shown in Figure 4.2 is the representation of overlap in operation between disk service and execution of instructions in the CPU. The fission node represents the creation of two related simulation tasks in relationship to a single service action for DISK1. RESQ operates to prevent exit from the companion fusion node until both of the two created tasks are completed. This fission/fusion node pairing models the situation in which the total service time of CPU execution or disk service.

Consequently, the improvement caused by successful operation of disk cache is moderated by the amount of CPU overlap time.

Repeating for emphasis, two tasks - a CPU overlap task, and a disk service task - leave the fission node and are later recombined at the companion fusion node. This disk service task enters a new passive queue denoted as CNTLQ in Figure 4.2. A discussion of the operation of passive queue CNTLQ is deferred until the section devoted to modeling the physical access portion of disk service. For the present, the disk service task eventually receives its token and proceeds with the service time represented by the box with label "CACHE ALGO MGMT". The given box represents the processing time undertaken by the disk controller to determine if the given disk access is a cache hit or a cache miss. Other housekeeping details associated with the management of the cache buffers are also included within this service block.

(B). MODEL REPRESENTATION OF HIT PROCESSING AND MISS PROCESSING

Continuing in Figure 4.2, the disk service task proceeds through a probabilistic routing choice to obtain service as a cache hit or a cache miss. The probabilities P(HIT) and P(MISS) reflect the success of the cache buffer management policies in the presence of the given application environment. Based on Figure 4.1, the model in Figure 4.2 reflects the buffer management decision that a hit involves transfer of data between the CPU and the cache buffer. Note that the token from CNTLQ is not released until after the data transfer is completed. The case of a miss requires a physical access to the disk which is represented in Figure 4.2 by entry to the DISK1 physical access model. Note further that for the case of a miss, the token from CNTLQ is released immediately after completion of the cache algorithm management function. RESQ permits two or more release nodes in relation to a given allocation node.

It should be observed in Figure 4.1 that several distinctions are made between a read access and a write access. In particular, the data transfer amount differs between a read and write. Another difference is that a write hit requires a physical access to the disk whereas a read hit does not. These circumstances are accommodated in Figure 4.2 by inclusion in both the hit processing and in the miss processing sequences by a probabilistic branch based on the probability of a write access P(WRITE), and the probability of a read access P(READ). As shown in Figure 4.2, both the read hits and the write hits experience the service time associated with data transfer with the CPU. The read hit then goes to the fusion node; the write hit goes to the model of the physical access time portion of disk service. As dictated by Figure 4.1, set nodes are used to set the data transfer time used in the physical access model to correspond to data transfer amount 2.

(C). DISCUSSION OF THE PHYSICAL ACCESS MODEL AND CONSISTENCY OF RESPONSE TIME

Figure 4.3 provides the final level to be considered in the expansion of detail for the disk cache subsystem.

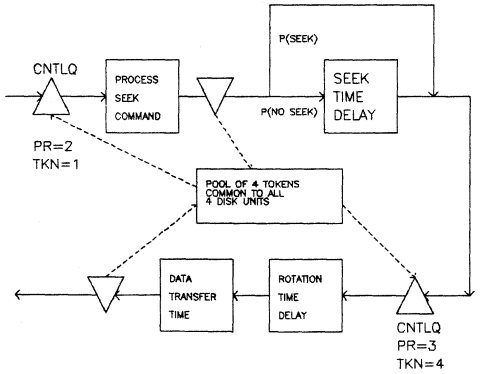

Figure 4.3. Cache Control Portion of Disk Unit I Model

The physical access model shown in Figure 4.3 represents the operation of both the disk controller and the actual disk unit during the conduct of a physical access to the disk unit. The major items are:

- Processing the command in the disk controller so that the seek operation is sent to the proper disk actuator.
- The delay associated with completing the seek operation at the disk actuator.
- The delay associated with waiting for the proper data record to rotate under the disk read/write head. The time necessary to transfer the requested data to or from the disk surface.

In terms of consistency of system level response time, the determining factors are the degree to which there is overlap of process times among several disks balanced against blockage by one disk of operations on another disk.

The potential impact of the disk cache can be reviewed in conceptual terms. Without the cache, all disk service requests become physical disk accesses and there is a great need to achieve overlap. However, under heavy load, the existence of skew and the operation of precedence relations described in the next section act to delay many disk accesses within the disk controller. With the disk cache, many of the disk service

requests are satisfied very rapidly by the cache with the service model shown in Figure 4.2. with disk cache, a much reduced number of disk service requests become physical disk access. For those physical accesses remaining, there is a much improved chance for normal service. Consistency response time is potentially improved by the combination of the rapid service associated with the cache buffer and the much improved chances for the slower physical accesses to proceed without delay.

5. MODEL OF SERVICE TIME FOR PHYSICAL DISK ACCESS

The model structure depicted in Figure 4.3 is intended to simulate the sequence relationships shown in Figure 5.1.

---

* Overlap is Possible Among Seek Operations for Different Disks

* No Overlap of Rotation Delay and Data Transfer for Different Disks

* Cache Algorithm Management and Processing of Seek Commands can Continue if Initiated Prior to Start of Rotation Delay Data Transfer

* Start of Rotation Delay and Data Transfer Blocks Start of New Cycle of Cache Algorithm Management and Processing of Seek Commands

* Once Started a Seek Operation on One Disk can Continue During Rotation Delay and Data Transfer on Another Disk

Figure 5.1. Sequence Relationships for Operations Involving Disk Controller and Disk Unit

---

Such sequence precedence indicates which process of events can be overlapped among multiple disk units and the circumstances under which one disk can block the start of other events on a different disk. These sequence relations are dictated by the design considerations and limitations of available disk controllers. In particular, controllers of interest within a Series/1 environment implement only a single read/write path between the I/O channel and the attached disks. Thus, only one disk at a time can be actively involved in transferring data. By way of contrast, each disk can undertake an independent seek operation and these can be overlapped.

In comparing the degree to which the model in Figure 4.3 correctly represents the functions of Figure 5.1, special emphasis is placed on the role played by the passive queue CNTLQ. This same passive queue appears in Figure 4.2 relative to process time for cache management algorithms. The key feature of queue CNTLQ is that it allocates tokens on a priority basis and in different amounts at each of the allocation nodes. The start of rotation time delay and data transfer time processes on one disk in the IBM 4967 controller will prevent the start of a new seek operation or any process of a new seek command for any other disk attached to the same controller. However, the seek operation could continue on other disks in overlap fashion if started before the rotation time delay/data transfer delay on the other disk. Thus, there is much motivation to initiate if

possible new seek operations on the remaining disks prior to start of rotation delay on a given disk having previously completed its seek operation.

Figure 4.3 shows one way in which the desired sequence relations can be simulated. It is seen that the allocation node prior to rotation time delay has the lowest priority (priority three, denoted as PR=3) among all allocation nodes for CNTLQ. Also, the rotation delay allocation node requires all four of the available tokens (denoted as TKN=4). Assume that while a seek command or algorithm processing cycle is underway on DISK1 that DISK2 has completed its seek operation. The operation of the priority allocation of tokens allows the command processing on DISK1 to run to completion. The seek operation itself requires no tokens so that the seek operation can start immediately after completion of processing the seek command. Thus, DISK1 releases its token at the completion of processing the seek command. DISK2 may acquire that token and thereby fulfilling the token requirement for it to proceed. The objective is met for overlapping a seek on DISK1 with rotation delay on DISK2.

If the seek operation on DISK1 finishes before DISK2 is finished with its service, then DISK1 must wait to start its rotation delay until DISK2 releases its four tokens. DISK2 could then seize them if no high priority request existed.

Completing the description is the case in which a rotation delay period has started on DISK2 and afterwards a seek command arrives at DISK3. In this case, DISK2 has all the available tokens and blocks the start of command processing on DISK3. Finally, DISK2 finishes its service and releases its four tokens. If DISK1 were waiting for the start of its rotation delay, that rotation delay on DISK1 waits to get started until DISK3 completes processing of the seek command. Finally, DISK1 is able to obtain all four tokens for start of its operation. As a minor point, both the allocation node and service processing for management of cache algorithms are given higher priority than allocation node and service processing for the seek command. Thus, if a service request has arrived at DISK4, its cache processing is completed before processing of the seek command on DISK3 or the rotation delay on DISK1.

6. A MEASURE FOR CONSISTENCY OF SYSTEM RESPONSE TIME

As an example of some of its more valuable outputs, RESQ provides for evaluation of both the average value of system level response time as well as the statistical distribution of this response time. The calculation process for the distribution is based on counting the fraction of responses which are longer than specified check point values. As experience is gained through repeated exercise of the model, the required check point values for response time can be given in relation to the previously determined average value of response time. In this manner, the fraction of responses can be found which are two times or even three times longer than the average value of all response times.

In many heavily used systems, it is not at all unusual to find that the slowest 10 of response times are up to three times the average response time. Such a state of affairs is very disturbing to the users at the terminals. Often it is the low fraction of very slow responses which make the impression on the user community. In the authors' opinion, these users tend to perceive the average value of response time to be much longer than it actually is based on their frustration with the inconsistent longer response times. Consequently, it is desirable to achieve a consistency level wherein much less than 10 of the slowest responses are more than 1.5 or 2.0 times the average response.

Ideally, the disk cache provides for improvement in consistency as well as improvement (reduction) in the value of average response time for the work load previously applied to the system. A further desirable feature for disk cache is an ability to preserve consistency in response times as the average value increases due to increases in workload which can be applied to the computer system.

7. RESULTS

Reported below are representative results to the consistency of system level response times for a Series/1 system based on a composite of installations visited by the author. Without DASD cache, the system is disk bound. The typical access at the disk unit can be taken as 38 milliseconds; and the typical CPU process time per access can be taken as 22 milliseconds, with perhaps 10 or 20 of this in overlap with the disk time. The case is considered where there are exactly 20 disk accesses per transaction, a hit ratio of 75 and four read accesses per each write access. Without the disk cache, the RESQ simulation showed the average response time to be 1.6 seconds and nearly 8 of responses were longer than three times the average. Under the same workload and with disk cache, the average response time is reduced to .96 seconds - a 40 improvement. As to consistency, there is also a significant improvement with fewer than 5 of responses being no more than twice the average response time. Similar benefits are maintained when the number of attached terminals is increased significantly.

8. CONCLUSIONS

The RESQ simulation provides a powerful and useful tool for gaining increased understanding regarding responses and interactions in a complex environment such as the transaction processing system with DASD cache described here.

Quantification is made possible for the significant improvement in response time and consistency achievable in many application environments with disk cache. As to RESQ, the real-system features of a priority structure for processing with combinations of overlap and precedence relations are easily represented by simulation facilities provided by RESQ. Additionally, simulation provides the only available means for determining the distribution in response time so crucial to the present study.

9. ACKNOWLEDGEMENTS

The first author gratefully wishes to acknowledge the support and encouragement of Dr. Jerry Merckel and Mr. Richard Robinson, as well as the significant contributions to engineering development of the IBM 4967 disk controller achieved by Mr. Jerry Dixon and Mr. Andrew MacNeil.

We are also grateful to the many colleagues and RESQ users who have helped us in this work. We particularly wish to thank Charles H. Sauer.

REFERENCES

1. P. Heidelberger and P.D. Welch, "A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations," CACM, Vol.24, No.4, pp. 233-245, (April 1981).
2. S.S. Lavenberg (Editor), E.A. MacNair, H.M. Markowitz, C.H. Sauer, G.S. Shedler and P.D. Welch, Computer Performance Modeling Handbook, Academic Press (1983).
3. M. Reiser and S.S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks," JACM 27, 2 (April 1980) pp. 313-322.
4. C.H. Sauer and K. M. Chandy, Computer System Performance Modeling, Prentice-Hall (1981).
5. C.H. Sauer and E.A. MacNair, "Queueing Network Software for Systems Modeling," Software-Practice and Experience 9, 5 (May 1979).
6. C.H. Sauer and E.A. MacNair, "The Research Queueing Package Version 2: Availability Notice," IBM Research Report RA-144, Yorktown Heights, New York (August 1982).
7. C.H. Sauer and E.A. MacNair, Simulation of Computer Communication Systems, Prentice-Hall (1983).
8. C.H. Sauer, E.A. MacNair and J.F. Kurose, "The Research Queueing Package Version 2: Introduction and Examples," IBM Research Report RA-138, Yorktown Heights, New York (April 1982).
9. C.H. Sauer, E.A. MacNair and J.F. Kurose, "The Research Queueing Package Version 2: CMS Users Guide," IBM Research Report RA-139, Yorktown Heights, New York (April 1982).
10. C.H. Sauer, E.A. MacNair and J.F. Kurose, "The Research Queueing Package Version 2: TSO Users Guide," IBM Research Report RA-140, Yorktown Heights, New York (April 1982).
11. C.H. Sauer, E.A. MacNair and J.F. Kurose, "The Research Queueing Package: Past, Present and Future," Proceedings of the National Computer Conference, pp. 273-280 (June 1982).
12. C.H. Sauer, E.A. MacNair and S. Salza, "A Language for Extended Queueing Network Models," IBM Journal of Research and Development, Vol.24, No.6, pp. 747-755, (November 1980).