Multiserver queuing models, coded for a handheld calculator, permit the evaluation of computer system performance, using powerful decision-making simulations to evaluate equipment, judge prospective reconfigurations, and modify existing operations

Computer Queuing Analysis On a Handheld Calculator

Ronald Zussman

Securities Industry Automation Corporation
New York, New York

Queuing theory can be practically applied to estimating computer performance and locating system bottlenecks. This is demonstrated by a generalized multiserver queuing model coded to run on the Texas Instruments SR-52 handheld calculator. The model is interactive so that effects of prospective design changes can be evaluated immediately, allowing an analyst to make the iterative design decisions required to optimize system effectiveness. Having the model preprogrammed for a portable calculator further increases its availability and convenience.

Where a complicated problem can be separated into a number of basic queuing situations, it is often possible to use this calculator model instead of an exten-

Multiserver Queuing Equations

Input Parameters

Average arrival rate: λ

Average service rate: u

Number of servers: s

Performance Attributes

Facility utilization: $U = \lambda/us$

Probability of finding no items* in the system:

$$P(0) = 1/\left(\left(\sum_{N=0}^{s-1} (\lambda/u)^N/N!\right) + (\lambda/u)^s/(s!(1-U))\right)$$

Probability of finding n items* in the system:

$P(n) = P(0)\ (\lambda/u)^n\ (1/n!)$,$n < s$

$P(n) = P(0)\ (\lambda/u)^n\ (1/(s!\ s^{n-s}))$,$n \geq s$

Probability of finding all servers busy (ie, that $\geq$ s items are in the system):

$$B = \sum_{n=s}^{\infty} P(n) = (\lambda/u)^s\ P(0)/(s!(1-U))$$

Average waiting time in the queue:

$TW = B/(su(1-U))$

Average system response time (ie, total time an item spends in the system):

$T = TW + 1/u$

Standard deviation of system response time:

$\sigma(T) = \sqrt{B(2-B) + s^2\ (1-U)^2}\ /(su(1-U))$

$\sigma(T) = \sqrt{B(2-B) + s^2\ (1-U)^2}\ TW/B$

Probability of waiting time in the queue exceeding t:

$P(TW>t) = B\ e^{-su(1-U)t}$

Average number of items in the queue:

$\overline{Q} = \lambda(TW)$

Average number of items in the system:

$\overline{N} = \lambda(T)$

*Items in the system either reside in the queue or are being processed by one of the servers.

TABLE 1

SR-52 User Instructions for Multiserver Queuing Model

Step	Procedure	Enter	Press			Display
1.0	Load program card (sides A and B)					
2.0	Input parameters:					
2.1	Average arrival rate	λ	STO	0	1	
2.2	Average service rate	u	STO	0	2	
2.3	Number of servers	s	STO	0	3	
3.0	Calculate P(0) before proceeding to Step 4					
3.1	Probability that 0 items are in the system*		P(0)			Probability
3.2	Facility utilization of each individual server		U			Utilization
4.0	Request the following in any order					
4.1	Probability that n items are in system*	n	P(n)			Probability
4.2	Probability of finding all servers busy (Probability that $\geq$s items are in system)		B			Probability
4.3	Average waiting time in queue		TW			Time
4.4	Probability that waiting time in queue >t	t	P(TW>t)			Probability
4.5	Average system response time* (Total time item spends in system)		T			Time
	Standard deviation of system response time		RUN			σ(T)
4.6	Average number of items in queue		$\overline{Q}$			Number
4.7	Average number of items in system*		$\overline{N}$			Number

*Items are in system when they reside in queue or are being processed by one of servers

sive and expensive real simulation. Although the discussion concentrates upon computer central site queuing, this model also has wide application in communication systems and industrial engineering.

Assumptions

Several simplifying and yet conservative assumptions are used to derive the analytic equations listed as "Multiserver Queuing Equations." Random arrival and random service are assumed. All servers are considered identical, with the same capacity and equal loading. Dispatching is first-in first-out (FIFO). No input traffic is lost because all arrivals are queued until they can be serviced.

Mathematically, the input traffic rate is assumed Poissonian in nature, with interarrival times following an exponential distribution. These distributions are characterized as memoryless and their arrivals as random. For this type input, arrival probability is constant and traffic load is independent of time. This means that different input rates must be modeled individually. Another attribute of the Poisson distribution is nonzero interarrival time, which rules out the occurrence of simultaneous inputs. Lastly, arrivals are independent of past events and have no influence on future inputs.

The model assumes exponential service times, which are mutually independent, identically distributed, random variables. Service time specifications parallel those for input traffic, as stated previously. Exponentially distributed interarrival and service times are Erlang 1 distributions, a special case of the gamma function. Time periods are so widely fluctuating and random that the standard deviation equals the mean.

$$(\text{mean} \div \text{standard deviation})^2 = \text{Erlang 1}$$

Their coefficient of variation is also 1.

$$\text{coefficient of variation} = (\text{standard deviation} \div \text{mean}) = 1$$

Modeling with exponential distributions is a standard conservative approach. In contrast, simulations using constant arrivals and service times predict best-case performance with minimal queuing. Constant distributions are Erlang infinity with standard deviations of zero.

TABLE 2

SR-52 Coding Form For Multiserver Queuing Model

Loc	Code	Key	Comments	Loc	Code	Key	Comments	Loc	Code	Key	Comments
000	46	LBL	RCL	075	55	÷		150	95	=	
	19	D′			42	STO			22	INV	
	42	STO			00	0			23	ln x	
	01	1			06	6			65	X	
	09	9			05	5			46	LBL	B
005	36	IND		080	19	D′		155	17	B′	
	43	RCL			55	÷			06	6	
	01	1			03	3			19	D′	
	09	9			19	D′			65	X	
	56	rtn			29	x!			07	7	
010	46	LBL	U	085	85	+		160	19	D′	
	16	A′			07	7			55	÷	
	01	1			19	D′			03	3	
	42	STO			95	=			19	D′	
	00	0			20	1/x			29	x!	
015	05	5		090	42	STO		165	55	÷	
	01	1			00	0			05	5	
	19	D′			07	7			19	D′	
	55	÷			56	rtn			95	=	
	02	2			46	LBL	P(n)		56	rtn	
020	19	D′		095	12	B		170	02	2	σ(T)
	55	÷			90	if zro			75	—	
	42	STO			11	A			17	B′	
	00	0			42	STO			65	X	
	04	4			00	0			17	B′	
025	03	3		100	00	0		175	85	+	
	19	D′			75	—			03	3	
	95	=			03	3			19	D′	
	22	INV			19	D′			40	x^2	
	44	SUM			95	=			65	x	
030	00	0		105	80	if pos		180	05	5	
	05	5			87	1′			19	D′	
	56	rtn			00	0			40	x^2	
	46	LBL	P(0)		19	D′			95	=	
	11	A			29	x!			30	$\sqrt{x}$	
035	16	A′		110	41	GTO		185	55	÷	
	03	3			46	LBL			53	(	
	19	D′			87	1′			17	B′	
	75	—			48	EXC			65	X	
	01	1			00	0			41	GTO	
040	42	STO		115	03	3		190	13	C	
	00	0			45	y^x			46	LBL	$\bar{N}$
	07	7			48	EXC			10	E′	
	95	=			00	0			01	1	
	90	if zro			03	3			19	D′	
045	00	0		120	65	X		195	65	X	
	07	7			03	3			53	(	
	00	0			19	D′			46	LBL	T
	42	STO			29	x!			14	D	
	00	0			46	LBL			02	2	
050	00	0		125	46	LBL		200	19	D′	
	04	4			55	÷			20	1/x	
	19	D′			07	7			85	+	
	45	y^x			19	D′			46	LBL	TW
	43	RCL			55	÷			13	C	
055	00	0		130	04	4		205	02	2	
	00	0			19	D′			19	D′	
	55	÷			45	y^x			20	1/x	
	43	RCL			00	0			55	÷	
	00	0			19	D′			03	3	
060	00	0		135	95	=		210	19	D′	
	29	x!			20	1/x			55	÷	
	95	=			81	HLT			05	5	
	44	SUM			46	LBL	P(TW>t)		19	D′	
	00	0			18	C′			65	X	
065	07	7		140	65	X		215	41	GTO	
	58	dsz			03	3			17	B′	
	00	0			19	D′			46	LBL	$\bar{Q}$
	05	5			65	X			15	E	
	01	1			02	2			01	1	
070	04	4		145	19	D′		220	19	D′	
	19	D′			65	X			65	X	
	45	y^x			05	5			41	GTO	
	03	3			19	D′			13	C	
	19	D′			94	+/−					

Labels: A = P(0), B = P(n), C = TW, D = T, σ(T), E = $\bar{Q}$, A′ = U, B′ = B, C′ = P(TW>t), D′ = RCL, E′ = $\bar{N}$

Registers: 00 = dsz, n; 01 = λ; 02 = u; 03 = s; 04 = λ/u; 05 = 1-U; 06 = $(\lambda/u)^s$; 07 = P(0)

The model delivers realistic results, even when actual distributions deviate from exponential. Practical job scheduling algorithms introduce even greater randomness than the FIFO discipline assumed in the model. Empirical measurements indicate that although coefficients of variation may exceed 1 at the central processing unit (CPU), they are usually less than 1 for the input/output (I/O) peripherals. Fortunately, these departures from exponential distributions have opposite effects on computer system performance and tend to negate one another.

Validity of predicted results depends on the accuracy of a model's input data as well as on the adaptability of the model to the system being simulated. Determining input data using a monitor is the most accurate approach. Using a hardware monitor to measure input parameters and the same monitor to validate the model's predictions, such queuing models for IBM and Univac mainframes have proven correct to within 5%. More typically, performance can be predicted to an accuracy of 10% either way of observed values.

The Model

Challenges encountered in programming this model for the SR-52 calculator included the problems of trying to fit the code within 224 instruction steps, minimize calculator execution time, and provide convenient user operation. The resultant program, listed in Tables 1 and 2, is an optimal tradeoff between these three sometimes conflicting objectives.

Subroutines were used extensively to enable the code to fit into 224 program steps. However, as subroutines take more time to execute than inline code, they were avoided in iterative loops. To further speed execution, frequently used labels were placed in low address locations and absolute addressing was used wherever possible.

Ten user-defined functions can be executed by pressing the top row of keys on the SR-52 calculator. To provide maximum user convenience, these user-function keys are named and referenced as follows:

Key	Name	Key	Name
A	P(O)	A′	U
B	P(n)	B′	B
C	TW	C′	P(TW > t)
D	T, σT	D′	—
E	$\overline{Q}$	E′	$\overline{N}$

With minor modifications this program is upward-compatible to TI-58 and -59 calculators, which use the same algebraic hierarchy and have the same number of user label keys as the SR-52. Using their additional program step capacity and companion PC-100A printer, the code can be expanded to print out alphanumeric messages for prompting and headings or to plot results.

Readers without access to TI calculators can use the multiserver model, by reprogramming the "Multiserver Queuing Equations" for whatever calculator is available.

The model requires input parameters λ, u, and s, which reflect characteristics of both workload and service facilities. As a preliminary step, these variables are stored in the calculator's memory registers. Then, as indicated in Table 1, "User Instructions," the simulation is run (Step 3.1) and any of the ten performance attributes listed in "Multiserver Queuing Equations" can be displayed.

Central Processor Simulation

Seven practical examples are given in Tables 3 and 4 to illustrate the technique for modeling CPU configurations. In Table 3, the three input parameters necessary for modeling are determined. In Table 4, the simulation results are listed.

One measure of processor speed is the time it takes to execute instructions. Average instruction execution time (AIET) depends on both the hardware and the software application. Scientific, business, and control system users have different AIETs on the same computer. To measure this value empirically, connect a counter to the instruction fetch strobe signal, then divide CPU active time by this instruction strobe count. The CPU in Table 3 has a given AIET of 1.072 μs. The reciprocal of AIET is the maximum number of instructions that can be executed by the processor in a unit time period. For modern full-scale computers, this number is so large that it is usually quoted in terms of millions of instructions per second (MIPS), which is the instruction execution rate for a saturated CPU running at 100% utilization. In this example, it is equal to 0.932836.

Before CPU service rate can be computed, the average number of instructions processed for each transaction or job must be established. Transactions are studied in real-time systems, while jobs are used for batch. If the system under investigation is still in the planning or design stage, the only way to proceed is to use a combination of time-consuming instruction counts (if code actually exists) and/or judicious estimating; this means tracing execution paths and actually counting or estimating the number of lines of code. The level of preliminary work done here will establish your confidence level in the simulation results. If the system is already operational, an exact count can be determined via a software accounting package, software monitor, or hardware monitor. Make certain that the final count includes instructions in the application code and a proportion of the operating system, utilities, and I/O handler overhead. In Table 3, the average number of instructions executed for each transaction is specified at 67,500.

CPU service rate is MIPS divided by the instruction-per-transaction overhead, then multiplied by 10^6 to keep units in line. In the previous example

$$\text{CPU service rate} = (0.932836 \div 67{,}500) \times 10^6$$
$$= 13.82 \text{ transactions/s}$$

Thus maximum CPU throughput is 13.82 transactions/s. At higher rates, queues continuously build, eventually outgrowing their buffers. Batch systems have their service rates quoted in jobs per second.

Two other parameters needed are number of servers and input transaction (or job) arrival rate. Number of servers (s) is taken equal to the level of multiprocessing. In uniprocessor configurations s = 1. For multiprocessors, where s is greater than 1, the model uses the actual overall arrival rate. In multiple uniprocessor configurations, only one CPU is modeled and input transactions are assumed to be uniformly distributed to all pro-

TABLE 3

Determining Input Parameters For Modeling CPU Configurations

Configuration	Total No. CPUs	Transaction Arrival Rate (per second)	CPU Service Rate (per second)	Model Input Parameters λ	u	s
2-CPU multiprocessor	2	25.00	13.82	25.00	13.82	2
3-CPU multiprocessor	3	25.00	13.82	25.00	13.82	3
4-CPU multiprocessor	4	25.00	13.82	25.00	13.82	4
2 uniprocessors	2	25.00	13.82	12.50	13.82	1
3 uniprocessors	3	25.00	13.82	8.33	13.82	1
4 uniprocessors	4	25.00	13.82	6.25	13.82	1
1 CPU (100% faster)	1	25.00	27.64	25.00	27.64	1

AIET = 1.072 μs

$$\text{MIPS} = \frac{10^{-6}}{\text{AIET}} = \frac{1}{1.072} = 0.932836$$

Average number of instructions-executed-per-transaction = 67,500
(Includes application + operating system + utilities + I/O handler)

CPU service rate = MIPS x 10^6 ÷ (instructions/transaction)

$$= \frac{932{,}836}{67{,}500} = 13.82 \text{ transactions/s}$$

TABLE 4

Queue Statistics (for CPU Configuration in Table 3)

Configuration	Simulation Results U	P(0)	P(1)	P(2)	B	TW	P(TW>.1)	T	σ(T)	$\bar{Q}$	$\bar{N}$
2-CPU multiprocessor	0.9045	0.0502	0.0907	0.0821	0.8591	0.3254	0.6598	0.3978	0.3819	8.1356	9.9446
3-CPU multiprocessor	0.6030	0.1443	0.2610	0.2361	0.3586	0.0218	0.0691	0.0941	0.0861	0.5446	2.3536
4-CPU multiprocessor	0.4522	0.1601	0.2896	0.2619	0.1304	0.0043	0.0063	0.0767	0.0742	0.1077	1.9166
2 uniprocessors	0.9045	0.0955	0.0864	0.0781	0.9045	0.6852	0.7926	0.7576	0.7576	8.5652	9.4697
3 uniprocessors	0.6027	0.3973	0.2394	0.1443	0.6027	0.1098	0.3481	0.1821	0.1821	0.9146	1.5173
4 uniprocessors	0.4522	0.5478	0.2477	0.1120	0.4522	0.0597	0.2121	0.1321	0.1321	0.3734	0.8256
1 CPU (100% faster)	0.9045	0.0955	0.0864	0.0781	0.9045	0.3426	0.6946	0.3788	0.3788	8.5662	9.4697

cessors. Therefore, arrival rate to the one modeled CPU is the overall arrival rate divided by the number of uniprocessors.

Computers often process several distinct classes of input simultaneously. When this occurs, these different classes may be combined and modeled as one representative type of input. The example in Table 5 shows how to merge three transaction types: inquiries, orders, and reports. Total arrival rate is a summation of individual arrivals:

17.0 + 20.0 + 4.5 = 41.5 transactions/s

The equivalent instructions-executed-per-merged transaction is weighted average, based upon relative arrival frequency and instructions-executed-per transaction:

(0.41 x 1500) + (0.48 x 6700) + (0.11 x 74,000)
= 11,971 instructions/transaction

As previously discussed, CPU service rate is computed by multiplying the given 0.500000 MIPS rate by 10^6 and then dividing this product by 11,971 instructions/transaction, yielding a result of 41.8 transactions/s.

Once input parameters are known for the merged transaction, a calculator simulation can be run to determine queue statistics.

When facility utilization exceeds 80%, both queue size and waiting time increase exponentially. The processor of Table 5 has a utilization just below 100% (U = 0.9928). This is too busy for satisfactory performance. Almost all of the 138.3333 transactions in the system ($\bar{N}$) are immobilized in the queue. Simula-

TABLE 5

Merging Transaction Types Into One Equivalent Transaction

MIPS = 0.500000

Three Transaction Types	Arrival Rate (per second)	Relative Arrival Frequency	Instructions Executed per Transaction	CPU Service Rate (per second)	CPU Utilization U
Inquiry	17.0	0.41	1,500	333.3	0.05
Order	20.0	0.48	6,700	74.6	0.27
Report	4.5	0.11	74,000	6.8	0.66
Totals	41.5	1.00			0.98

One Equivalent Transaction

Arrival rate = 41.5 transactions/s

Instructions/transaction = (0.41 x 1500) + (0.48 x 6700) + (0.11 x 74,000) = 11,971

CPU service rate = 500,000 ÷ 11,971 = 41.8 transactions/s

Model Input Parameters			Simulation Results				
λ	u	s	U	TW	T	$\bar{Q}$	$\bar{N}$
41.5	41.8	1	0.9928	3.3094	3.3333	137.3405	138.3333

tion predicts 137.3405 transactions queued ($\bar{Q}$), each waiting for an average of 3.3094 s (TW). Overall response time (T) is 3.3333 s. Therefore, only 3.3333 − 3.3094 = 0.0239 s are actually consumed for processing. Most of the response time (3.3094 ÷ 3.3333 = 99.28%) is spent waiting; only 0.72% is used in obtaining actual service.

While it is cost-effective to make good use of hardware, always consider the tradeoffs of adequate response time and realistic buffer size. No system can be expected

TABLE 6

Modeling Preemptive Priority Queue Disciplines

Three Transaction Types	Priority Level	Arrival Rate (per second) λ	Instructions Executed per Transaction	MIPS (for Subdivided Models)	CPU Service Rate (per second) u	U Subdivided Models
Inquiry	high	17.0	1500	0.500000	333.3	0.05
Order	medium	20.0	6700	(1-0.05) (0.500000) = 0.475000	70.9	0.28
Report	low	4.5	74,000	(1-0.05) (1-0.28) (0.500000) = 0.342000	4.6	0.98

Subdivided Models

Three Transaction Types	Model Input Parameters			Simulation Results					
	λ	u	s	U	TW	T	Throughput 1/T	$\bar{Q}$	$\bar{N}$
Inquiry	17.0	333.3	1	0.0510	0.0002	0.0032	316.3000	0.0027	0.0537
Order	20.0	70.9	1	0.2821	0.0055	0.0196	50.9000	0.1108	0.3929
Report	4.5	4.6	1	0.9783	9.7826	10.0000	0.1000	44.0217	45.0000

to operate safely with a facility utilization above 80% anywhere in the processing chain (CPU or channels). At high utilizations, any small increase in arrival rate or deviation of distribution may result in severely degraded performance. Data can be lost because of overflowing buffers or the system can be brought down entirely.

Until now, different transaction types have been implicitly assumed to be at the same priority level. However, modeling need not be so restricted. Table 6 develops the concept of job stream priorities. The approach is to subdivide one computer into a number of separate pseudomachines, one processing each transaction type, and each with its own MIPS rate. These pseudomachines are modeled independently to determine performance characteristics for the various classes of input.

Since Table 6 considers three transaction types, three pseudocomputers are modeled. Inquiries receive precedence and are given the full benefit of the 0.500000 MIPS. CPU service rates, for the pseudoprocessors, are calculated by multiplying their respective MIPS by 10^6 and then dividing by instructions per transaction. For inquiries, this service rate calculation is

$$0.500000 \times 10^6 \div 1500 = 333.3 \text{ inquiries/s (u)}$$

Once λ, u, and s are determined for the pseudoprocessors, the model can be run to display their queue statistics. Such simulation indicates that the inquiry processing pseudocomputer is only 5% utilized (U). Table 6 also lists waiting time (TW), response time (T), throughput (1/T), and queue length ($\bar{Q}$, $\bar{N}$) statistics.

Order processing ranks second, having preemptive priority over reports. The MIPS available to orders (0.475000) is computed by multiplying the original MIPS rate (0.500000) by the inverse of CPU utilization for the inquiry pseudoprocessor:

$$(0.500000) \times (1 - \text{inquiry CPU utilization})$$
$$= (0.500000) \times (1 - 0.05) = 0.475000 \text{ instructions/s}$$

The third pseudocomputer, for reports, gets whatever MIPS is left over (0.342000). This computation is analogous to the one for orders.

$$(0.475000) \times (1 - \text{order CPU utilization})$$
$$= (0.475000) \times (1 - 0.28) = 0.342000 \text{ instructions/s}$$

Resulting queue statistics, in Table 6, are valid performance indicators for the three transaction types. However, utilization of the actual processor remains to be resolved. The percentage of real CPU utilization, contributed by each input class, is calculated by multiplying arrival rate by instructions per transaction and then dividing by the product of the overall 0.500000 MIPS rate times 10^6.

		CPU Utilization
Inquiry:	$(17.0 \times 1500) \div (0.500000 \times 10^6) =$	5.1%
Order:	$(20.0 \times 6700) \div (0.500000 \times 10^6) =$	26.8%
Report:	$(4.5 \times 74{,}000) \div (0.500000 \times 10^6) =$	66.6%
	Total	98.5%

Combined CPU utilization for inquiry and order processing is only 5.1% + 26.8% = 31.9%. This indicates that inquiries and orders are processed more expeditiously than in the single priority system of Table 5. Report processing, having lowest priority, is degraded the most. Average report queue length ($\bar{Q}$) is 44.0217, and mean response time (T) is 10.0000 s.

Subdividing one computer into a number of pseudoprocessors assumes smooth and homogeneous CPU execution for transactions of all priority levels. However, in practice, CPU activity occurs in bursts interspersed with idle periods. Modeling errors will be small so long as either (1) instruction overhead for high priority transactions is shorter than for those of lower priority and/or (2) high priority transactions only lightly load the system. The degree to which these two assumptions are met determines result accuracy.

In Table 6, errors caused by our homogeneous processing assumption are less than 5%. Conditions ensuring accurate results are:

Instructions Per Transaction	Inquiry < Order < Report
	1500 < 6700 < 74,000
CPU Utilization	Inquiry + Order = 31.9%

Similar stipulations can be expanded to any number of priority classes. Generally, high priority transactions and jobs have shorter instruction execution paths than those of lower priority. In practice, preemptive priority queue modeling is applicable and its results are realistic. Case histories have been documented in which such pseudoprocessor analytic models were as accurate as discrete simulation packages which had run and accumulated statistics for many thousands of transactions.

Disc and Drum Channel Simulation

Five examples of channel configurations are given in Tables 7 and 8. Table 7 develops the three input parameters (λ, u, s) needed for modeling. Queue statistics resulting from these simulations are listed in Table 8.

Throughput capacities for three types of drums are computed in the upper half of Table 7. All quoted times and rates represent averages. Access time includes seeking and rotational delay. Drum seek times are always zero because heads are fixed and stationary. Average rotational delay (latency) is the time consumed by one-half a disc or drum revolution. Data transfer time depends on the physical characteristics of the device, r/min, and bit-packing density, as well as on blocking size. Adding access to data transfer time results in I/O service time. The reciprocal of service time is service rate u, the maximum possible number of device I/O operations per second.

When different classes of transactions reference the same I/O device, the relative frequency of their accesses is used to compute a weighted average block size. This is done by multiplying the block size of each transaction

TABLE 7

Modeling Channel Configurations

Channel Characteristics

Drum Type	Access Time(s)	+	Transfer Time/Byte(s)	x	Bytes/Transfer	=	Service Time per I/O(s)	Service Rate(/s)
A	0.01700		4.25×10^{-6}		560		0.019380	51.60
B	0.00425		4.25×10^{-6}		2048		0.012954	77.20
C	0.09200		7.00×10^{-6}		384		0.094688	10.56

Channel Configurations

Example	Drum Type	Description	Total No. Channels	I/O Rate/s	Model Input Parameters λ	u	s
1	A	1 single channel	1	22.20	22.20	51.60	1
2	B	2 single channels	2	150.00	75.00	77.20	1
3	B	1 dual channel	2	150.00	150.00	77.20	2
4	C	4 single channels	4	4.32	1.08	10.56	1
5	C	2 dual channels	4	4.32	2.16	10.56	2

TABLE 8

Queue Statistics
(for Channel Configurations in Table 7)

Example	Model Input Parameters λ	u	s	Simulation Results U	P(0)	P(1)	P(2)	B	TW	P(TW>0.1)	T	σ(T)	$\bar{Q}$	$\bar{N}$
1	22.20	51.60	1	0.4302	0.5698	0.2451	0.1055	0.4302	0.0146	0.0227	0.0340	0.0340	0.3249	0.7551
2	75.00	77.20	1	0.9715	0.0285	0.0277	0.0269	0.9715	0.4416	0.7796	0.4545	0.4545	33.1194	34.0909
3	150.00	77.20	2	0.9715	0.0145	0.0281	0.0273	0.9575	0.2176	0.6166	0.2306	0.2274	32.6407	34.5837
4	1.08	10.56	1	0.1023	0.8977	0.0918	0.0094	0.1023	0.0108	0.0396	0.1055	0.1055	0.0117	0.1139
5	2.16	10.56	2	0.1023	0.8144	0.1666	0.0170	0.0190	0.0010	0.0028	0.0957	0.0952	0.0022	0.2067

type by the percent of accesses attributable to that transaction type. Summing all these products gives the weighted average block size. Then, device service rate is calculated as shown in Table 7. Multiplying average block size by the transfer time per byte and adding access time gives the service time per I/O, the reciprocal of which is service rate (u).

Peripherals having the most influence on system performance are mass data storage discs and drums. Channel configurations interface sets of these I/O devices to the CPU. A one-path interconnection is interpreted as a "single" channel; two access paths represent a "dual" channel, where any two peripherals in the set can be accessed simultaneously.

Drum channel queuing is more pronounced than queuing at the device level; so rather than model individual peripherals, aggregate channel activity is considered. A channel is in use for the total busy time of all drums accessed through it. The underlying supposition is that both a drum and its channel are seized and released simultaneously. Queues form when all channel access paths to a device are in use. "Single" channels are represented by one queue and one server. A "dual" channel is modeled as two servers replenished from one queue.

Channel service rate is determined by the peripherals interfaced, and equates directly to drum service rate in Table 7 because all drums on each channel have identical specifications. In practice, there may be cases where the same channel is used by several different types of drums. Channel service rate is then an average of drum service rates, weighted according to frequency of drum accesses. This is calculated by multiplying the service rate of each drum on the channel by the percent of channel accesses attributable to that drum. Channel service rate is the sum of these products.

Selector channels handling command-chained channel programs are busy during the total time that their discs

TABLE 9

Determining Optimum Multiprogramming Level For Computer System

Model Entire Multiserver CPU (where each initiator is a server)						Drum Type B on One Dual Channel					Total System	
Number of Initiators N	λ	u	s	U	Response Time T_1	λ	u	s	U	Response Time T_2	Response Time $T=T_1+T_2$	Message Throughput N÷T
1	3	40.00	1	0.0750	0.0270	1.5	77.20	2	0.0097	0.0130	0.0400	25.03
2	6	20.00	2	0.1500	0.0512	3.0	77.20	2	0.0194	0.0130	0.0642	31.17
3	9	13.33	3	0.2250	0.0761	4.5	77.20	2	0.0291	0.0130	0.0891	33.68
4	12	10.00	4	0.3000	0.1013	6.0	77.20	2	0.0389	0.0130	0.1143	35.00
5	15	8.00	5	0.3750	0.1269	7.5	77.20	2	0.0486	0.0130	0.1399	35.74
6	18	6.67	6	0.4500	0.1530	9.0	77.20	2	0.0583	0.0130	0.1660	36.15
7	21	5.71	7	0.5250	0.1800	10.5	77.20	2	0.0680	0.0130	0.1930	36.27
8	24	5.00	8	0.6000	0.2087	12.0	77.20	2	0.0777	0.0130	0.2217	36.08
9	27	4.44	9	0.6750	0.2410	13.5	77.20	2	0.0874	0.0131	0.2541	35.43
10	30	4.00	10	0.7500	0.2807	15.0	77.20	2	0.0972	0.0131	0.2938	34.04
11	33	3.64	11	0.8250	0.3389	16.5	77.20	2	0.1069	0.0131	0.3520	31.25
12	36	3.33	12	0.9000	0.4600	18.0	77.20	2	0.1166	0.0131	0.4731	27.48
13	39	3.08	13	0.9750	1.2227	19.5	77.20	2	0.1263	0.0132	1.2359	10.52
14	42	2.86	14	1.0500	—	21.0	77.20	2	0.1360	0.0132	—	—

and drums are active; this includes seek (arm movement), latency (rotational delay), and data transfer times. Access time is the sum of seek and latency. Drums, having fixed heads, always have a seek time of zero. Once access time is established, channel calculations are made as in Table 7.

If one disc channel program initiates a standalone seek and a second channel program initiates a data transfer operation (after the seek is complete), the selector channel is freed during the seek operation and its access time is equal only to latency. The tradeoff is the additional overhead caused by the extra interrupt and I/O supervisor processing which needs to be added to the CPU model.

Block-multiplexer channels are more efficient because they automatically disconnect during seeks, even with only one command-chained channel program. Utilization of a disc block-multiplexer channel can be as low as one-half or one-third of the combined utilizations of its individual discs. Disc block-multiplexer channels are modeled by proceeding as indicated in Table 7 without adding disc seek time into access time. Block-multiplexer channels with "rotational position sensing" are also free during latency and their access time is therefore zero.

Rules for representing channel arrival rates (λ) and number of servers (s) are similar to those for CPUs. Single channels are modeled as one server. λ is the total I/O arrival rate divided by the number of single channels. Only one of the single channels in the group is actually simulated, with its results applying to all. A dual channel is modeled as two servers fed from one queue; dividing the total I/O arrival rate by the number of dual channels computes to λ. Table 7 provides additional details.

The model assumes an absence of dual channel device lockouts, attempting to simultaneously use one peripheral device via each of the two access paths. One request is blocked until completion of the other. With a number

of I/O devices on each channel and an even distribution of files, the probability of device lockout is small. Instances of high probability are best represented by the single channel model.

Central Site Modeling

When transation flow can be approximated by sequential queues, CPU and channel models can be consolidated. Total central site response time is estimated by summing response times for processor and each appropriate channel.

Table 9 shows how to determine the optimum number of initiators. This example is a CPU and drum channel system with two queues in series. Total transaction response time (T) is computed by adding together computer (T_1) and drum channel (T_2) response times. Dividing the number of initiators (N) by this total response time (T) approximates a theoretical system throughput capacity.

The processor is multiprogrammed with initiators modeled as individual servers. Total CPU service rate remains fixed at 40.00 transactions/s. Number of initiators corresponds to level of multiprogramming. An increasing number of servers is balanced by a diminished processing power allocated per server. Each initiator is consigned a service rate (u) of $40.00 \div N$.

The goal is maximizing real transaction throughput for the system. This corresponds to the maximum transaction arrival rate (λ) which can be handled. The last column in Table 9 lists a theoretical system throughput capacity ($N \div T$). Optimal real system throughput is the highest transaction arrival rate which satisfies the relationship $\lambda \leq (N \div T)$. In Table 9 the best choice for N is 10 ($\lambda = 30$ and $N \div T = 34.04$); at this point, the CPU arrival rate and theoretical system throughput capacity curves effectively intersect. Higher traffic rates cannot be supported by the system and will result in queue backup and overflow. For example, at $N = 11$, arrival rate (λ) of 33 transactions/s is higher than the theoretical system throughput capacity ($N \div T$) of 31.25 transactions/s.

Total system throughput depends on both processor and drum I/O. As the number of initiators is increased, the CPU saturates much more rapidly than the channel. At $N = 11$, CPU utilization has already reached 82.5% while channel utilization is only 10.69%. The computer determines the level of multiprogramming, and its utilization controls the shape of the theoretical system throughput capacity curve. Drum hardware characteristics establish a lower bound ($T_2 = 13$ ms) on system response time.

Summary

The generalized computer system simulation package presented can evaluate any level of multiprogramming and multiprocessing with any combination of interactive and batch workloads. Elaborate queuing formulas have been reduced to a simple technique for efficiently evaluating complex systems on a programmable calculator.

A wide range of problems facing manufacturers of mainframes and peripherals as well as end-users can be addressed. The impact of hardware configuration changes and software modifications on performance can be anticipated. The effect of totally new hardware, improvements to existing machines, and the interplay of components can be assessed. The consequences that projected increases in a system's online workload will have on response times, batch throughput, and turnaround times can be predicted. Bottlenecks that are likely to arise as workload increases can be located.

Before using the model, the computer designer merely characterizes system resource demands and capabilities with a small set of easily researched parameters, determined by some combination of monitor measurements, estimates, or design specifications. Pressing a few calculator keys simulates the effect of job contention for processor and I/O, predicting the performance of any computer system. With such a queuing model, the programmable calculator becomes a convenient and indispensable tool for predicting and optimizing system performance.

Bibliography

C. W. Churchman, R. L. Ackoff, and E. L. Arnoff, *Introduction to Operations Research*, John Wiley & Sons, Inc, New York, 1957, Chap 14

J. Martin, *Design of Real-Time Computer Systems*, Prentice-Hall, Inc, Englewood Cliffs, NJ, 1967, Chap 26

J. Martin, *Systems Analysis for Data Transmission*, Prentice-Hall, Inc, Englewood Cliffs, NJ, 1972, Chap 31

M. Reiser, "Interactive Modeling of Computer Systems," *IBM Systems Journal*, No. 4, 1976, pp 309-327

Univac Applications Engineering, "Central Site Queuing Model," *System Design and Simulation Technical Memorandum*, Univac DPD, PO Box 8100, Philadelphia, Pa, June 1969

J. Webb, "Queuing Theory and Performance Analysis," *Computer Measurement and Evaluation*, Vol III, SHARE, Inc, Chicago, Ill, Dec 1973-Mar 1975, pp 309-332

R. Zussman, "Computer Simulation On a Pocket Calculator," *Computer Design*, May 1977, pp 105-109

Ronald Zussman, currently senior consultant and project leader of computer measurement and evaluation at SIAC, has experience that includes development of interactive models of shipboard computer systems and a prototype of an automated trading post. He holds BSEE, MSEE, and professional EE degrees from Pratt Institute, New York University, and Columbia University, respectively.