Previously, computer system simulation was highly complex, time-consuming, and tedious, requiring man-months of software derivation and hours of computer run time. Now it is possible to evaluate and optimize a large number of computer hardware design configurations in minutes, using a simple program and a handheld calculator

Computer Simulation On a Pocket Calculator

Ronald Zussman

Securities Industry Automation Corporation
New York, New York

Modeling a proposed computer system helps to specify and document the design. Being forced to ask the right questions is valuable, even if the modeling effort progresses no further. After gathering preliminary specifications, an early simulation model will give the designer a feel for performance, so that hardware can be chosen correctly. As software is written, programmers can use models to make intelligent gross decisions by experimenting with various design alternatives. After the computer system is built, actual performance can be measured to refine earlier estimates and fine-tune the model. Enhancements, modifications, and additions then can be simulated without disturbing or jeopardizing the integrity of the actual system.

Practical analytical models have not been well publicized. Usually scholarly and abstruse, their derivations appear in highly mathematical journals, and involve heavy probability and queuing theory. For the past 12 years, the author has applied classical multi-server queuing equations to computer system design with great success. The simulation model presented is based on this foundation.

Modeling a computer system no longer requires man-months of programming work or hours of computer run time. Specialized programming languages or proprietary software packages are useful tools but are not always necessary. This article describes an understandable analytic model, which is based on queuing theory and is programmed for the Texas Instruments SR-52 pocket calculator. It is ready for immediate and practical application.

Having this computer system model preprogrammed on a pocket calculator will expand the design engineer's decision-making capacity; the ability to immediately evaluate and optimize a large number of computer configuration problems is readily available. Instruction listing and simulation examples for the SR-52 are included; with reprogramming, the simulation model can run on a Hewlett-Packard HP-67.

Model Objectives

An entire computer system has been modeled, including central processor unit (CPU), main memory, peripheral auxiliary storage, and terminals or card readers. Any level of multiprocessing and multiprogramming is accommodated. The model is extremely powerful, and only a few simplifications are necessary. Multiprogramming is taken to operate at a fixed level, M. The model can overlap accesses to peripheral storage devices (I/O) with CPU processing for different jobs, but serializes them within each job. Average access service time, i, is assumed to be independent of the number of outstanding I/O requests. Although this means that a simulation does not explicitly account for I/O queuing or optimization, service time i may be judiciously adjusted so that they are taken into con-

sideration. Such parameter fine-tuning could be part of a validation process.

Three measures of computer system performance are processor utilization, throughput, and response time. CPU utilization is of interest mainly to the designer, while the operations manager and users are more concerned with the job processing rate and in job turnaround. As the model simulates both batch and real-time systems, the terms job, message, and inquiry are used synonymously. Job processing rate is throughput. Average response and turnaround time take on the same meaning: the total time a job spends in the system from input to output.

The objective is to optimize performance at minimal cost. Throughput should be maximized and response time minimized, all within cost constraints. Very low CPU utilizations are undesirable as they show that the processor is not operating anywhere near its rated capacity. On the other hand, allow for future expansion and never attempt to fully utilize a CPU.

Simulation results must be interpreted correctly. Queue sizes are not predicted by the models. However, classic probability and queuing theory indicates that queues grow alarmingly as CPU utilizations exceed 80% (see Martin, p 384). When this occurs, there is the danger of buffers and files overflowing. Operating system performance is also usually severely degraded at very high CPU utilization. In effect, the operating system is so busy that it becomes confused and does not know what to do next. Therefore, when the model predicts CPU utilizations above 80%, recognize this as a danger zone and proceed with caution.

Constant and Exponential Service Times

Derivation of analytic models is so complex that certain compromises must be made for the sake of simplicity. Establishing service times as either constant or exponential is standard procedure in deriving analytic queuing models. These are the two distributions which produce manageable equations.

When service times are constant, there is no variation from the mean. The standard deviation is zero. Probability of service completing is a direct function of what has transpired earlier in time. There is no uncertainty to complicate calculations.

Assuming exponential service time is equivalent to stating that there is a Poisson service pattern. This translates to a fluctuating service rate where the standard deviation squared equals the mean. At any point in time, the probability of completing service is entirely independent of what has occurred earlier. It is this lack of memory that simplifies the derivation.

Think of constant and exponential services as inverses—one always fixed, the other fluctuating. Fortunately, this works out well, because most situations encountered in real systems lie somewhere between the two. Usually, constant and exponential services are both modeled. Then, both models are simulated to establish boundaries. Real system performance lies somewhere between these limits.

Service times are assumed to be either all exponential or all constant. Both models have been programmed for the SR-52. The consequences of an exponential assumption are a lower limit for CPU utilization and throughput, and an upper boundary for response time. Postulating constant service times results in an upper boundary on CPU utilization and throughput, and a lower limit for response time.

Calculator Input Parameters

CPU utilization keys marked Ue and Uc apply to the exponential and constant models, respectively. Throughput (T) and response time (R) keys are used by both models. Simulation requires storing input parameters in calculator memory registers 1 through 7, and depressing the appropriate CPU utilization key. After utilization has been displayed, either throughput (T) or response time (R) can be computed.

Concise user instructions for the SR-52 programmable calculator are listed in Table 1. Inquisitive readers may wish to consult Boyse and Warn for the equations and derivations. However, in a practical sense, the computer designer is given all that he needs to run and use the model in the instructions that follow.

CPU utilization cannot really exceed 100%. However, as a mechanism to indicate the degree of a possible overload, Uc is permitted to display higher values. When this occurs, the model equates Uc to 100% in any following calculations; the user should do the same.

Handheld calculator and preprogrammed card simplify computer system simulation for fast, accurate, and convenient design analysis

TABLE 1

SR-52 User Instructions for Analytic Computer Model

Step	Procedure	Enter	Press	Display
1.0	Load program card (sides A and B)			
2.0	Input parameters, all with same time unit			
2.1	Average CPU time (application + OS) per interaction or job $C = \frac{\text{Total CPU active time}}{\text{Number of jobs}}$	C	STO 0 1	
2.2	Average CPU time period between I/O operations $c \leq C$ $c = \frac{C}{\text{Number of I/Os/job}}$	c	STO 0 2	
2.3	Average service time for an I/O request. You may add in I/O queuing time i = (% I/O drum) x (delay + transfer) + (% I/O disc) x (delay + transfer + seek)	i	STO 0 3	
2.4	Fixed level of multiprogramming Number of jobs simultaneously in main storages of all CPUs	M	STO 0 4	
2.5	Number of active terminals or customers $N \geq M$	N	STO 0 5	
2.6	Average user think time If $N = M$, $Z = 0$	Z	STO 0 6	
2.7	Number of multiprocessing CPUs. Jobs execute on any CPU $j \leq M$	j	STO 0 7	
3.0	Exponential model: Lower bound on CPU utilization and throughput Upper bound on response time			
3.1			Ue	CPU utilization
3.2	Interactions or jobs processed per unit time		T	Throughput
3.3	Turnaround time		R	Average response time
4.0	Constant model: Upper bound on CPU utilization and throughput Lower bound on response time			
4.1			Uc	CPU utilization
4.2	Interactions or jobs processed per unit time		T	Throughput
4.3	Turnaround time		R	Average response time

Seven input parameters are required for a simulation. They can be determined by reviewing documentation and/or monitoring actual system performance.

The time the processor is active for each job or message is denoted as C, which is a summation of application plus operating system components. Within a representative time period, dividing the total time the CPU is active by the number of jobs processed calculates to C. Store this quotient in calculator memory 1.

TABLE 2

Simulation Model Examples

	Calculator Input Parameters							Computer System Results: Exponential Model			Constant Model		
Example	C	c	i	M	N	Z	j	Ue (%)	T	R (s)	Uc (%)	T	R (s)
1	0.10	0.006	0.035	6	240	10	2	43.21	8.64	17.77	43.90	8.78	17.33
2	0.10	0.006	0.035	8	240	10	2	56.55	11.31	11.22	58.54	11.71	10.50
3	0.10	0.006	0.035	14	240	10	2	87.66	17.53	3.69	100.00 (102.44)	20.00	2.00
4	0.10	0.006	0.035	18	240	10	2	97.15	19.43	2.35	100.00 (131.71)	20.00	2.00
5	0.10	0.006	0.022	6	240	10	2	61.21	12.24	9.61	64.29	12.86	8.67
6	0.10	0.006	0.015	6	240	10	2	76.66	15.33	5.65	85.71	17.14	4.00
7	0.10	0.006	0.012	6	240	10	2	84.44	16.89	4.21	100.00	20.00	2.00
8	0.10	0.006	0.005	6	240	10	2	98.70	19.74	2.16	100.00 (163.64)	20.00	2.00
9	0.20	0.012	0.035	6	240	10	2	70.60	7.06	23.99	76.60	7.66	21.33
10	0.10	0.006	0.035	6	240	10	1	74.69	7.47	22.13	87.80	8.78	17.33
11	0.05	0.003	0.035	6	240	10	1	45.45	9.09	16.40	47.37	9.47	15.33
12	0.10	0.006	0.035	6	240	0	2	43.21	8.64	27.77	43.90	8.78	27.33
13	0.10	0.006	0.035	6	400	10	2	43.21	8.64	36.28	43.90	8.78	35.56

Note: All input parameters are required to be expressed in the same time unit (ie, seconds)

C = (total CPU active time) ÷ (number of jobs processed)

For the same time period, divide the number of accesses to disc and drum by the number of jobs processed. The result is the number of I/O accesses per job. Divide the previously calculated C by this new quantity. The resulting quotient, c, should be stored in calculator memory 2.

c = C ÷ (number of I/O accesses per job)

References to disc and drum take time. Among the factors to be considered are rotational delay, data transfer time, and seek time. All three need to be included for discs. Fixed head drums, by definition, have no arm movement and therefore zero seek time.

Calculator memory 3 must contain the average service time for a peripheral storage access. To determine this quantity, designated by i, compute the service time for each peripheral storage device in the system. This is the sum of its rotational delay, data transfer time, and seek time. Multiply each device's service time by its percentage of I/O accesses. Finally, add these weighted service times. Store the resulting sum of i into memory 3.

Dx = rotational delay for device x

Tx = data transfer time for device x

Sx = seek time for device x

n = number of storage devices in the system

i = (% of accesses to device 1) x (D1 + T1 + S1) +
(% of accesses to device 2) x (D2 + T2 + S2) + ... +
(% of accesses to device n) x (Dn + Tn + Sn)

The number of jobs simultaneously active within a system is the level of multiprogramming, M. Often this is equivalent to the number of specified initiators. Once accepted into main memory, a job is considered active, albeit in queue, in processing, or accessing peripheral storage. Where there is multiprocessing, M includes all jobs active within all processors. Store M in calculator memory 4.

N, indicating the number of system users, is stored in calculator memory 5. For a real-time system, N represents the number of terminals.

How soon after receiving a printout does a programmer submit his next job? What is the time period between a customer's inquiry and his terminal's preceding response? These are "user think" times. The average contemplation time period between a system response and a user's succeeding reply is given notation Z and loaded into calculator memory 6. A user submits jobs or messages with an interarrival time that is the summation of his think time (Z) plus system response time (R).

Multiprocessing level, designated by j, is the number of CPUs embodied in the system. For example, a uniprocessor equates j to 1. This is the last of the model's input parameters and is stored in calculator memory 7.

Simulation Model Examples

Thirteen illustrative simulation model examples (see Table 2) provide useful exercises. Both input parameters and simulation results are tabulated. Actual computer system performance is somewhere between the boundaries indicated by the exponential and constant models. Examples 1 through 4 show the effect of increasing the level of multiprogramming (M). Note that CPU utilization and throughput both rise, but response time falls. Once the CPU reaches saturation, throughput

TABLE 3

SR-52 Coding Form for Analytic Computer Model

Loc	Code	Key	Loc	Code	Key	Loc	Code	Key
000	46	LBL	075	00	0	150	44	SUM
	19	D′		09	9		01	1
	42	STO		58	dsz		02	2
	00	0		00	0		58	dsz
	00	0		07	7		01	1
005	56	rtn	080	03	3	155	01	1
	46	LBL		09	9		05	5
	10	E′		10	E′		43	RCL
	42	STO		44	SUM		01	1
	01	1		00	0		03	3
010	09	9	085	08	8	160	55	÷
	36	IND		43	RCL		07	7
	43	RCL		01	1		10	E′
	01	1		00	0		85	+
	09	9		19	D′		01	1
015	56	rtn	090	01	1	165	95	=
	46	LBL		22	INV		19	D′
	16	A′		44	SUM		81	HLT
	43	RCL		01	1		46	LBL
	00	0		00	0		13	C
020	04	4	095	58	dsz	170	00	0
	75	—		00	0		10	E′
	43	RCL		06	6		75	−
	00	0		09	9		01	1
	00	0		08	8		95	=
025	85	+	100	10	E′	175	22	INV
	01	1		20	1/x		80	if pos
	95	=		94	+/−		01	1
	55	÷		42	STO		08	8
	43	RCL		01	1		01	1
030	00	0	105	03	3	180	00	0
	03	3		07	7		85	+
	65	x		10	E′		01	1
	43	RCL		49	PROD		95	=
	00	0		01	1		65	x
035	02	2	110	03	3	185	07	7
	55	÷		42	STO		10	E′
	53	(		01	1		55	÷
	43	RCL		02	2		01	1
	00	0		19	D′		10	E′
040	00	0	115	08	8	190	95	=
	75	−		10	E′		56	rtn
	07	7		20	1/x		46	LBL
	10	E′		42	STO		14	D
	54	)		01	1		13	C
045	80	if pos	120	01	1	195	20	1/x
	00	0		16	A′		65	x
	05	5		49	PROD		05	5
	03	3		01	1		10	E′
	00	0		01	1		75	−
050	10	E′	125	58	dsz	200	06	6
	95	=		01	1		10	E′
	56	rtn		02	2		95	=
	07	7		01	1		81	HLT
	10	E′		43	RCL		46	LBL
055	95	=	130	01	1	205	12	B
	56	rtn		02	2		04	4
	46	LBL		75	−		10	E′
	11	A		07	7		55	÷
	01	1		10	E′		07	7
060	42	STO	135	95	=	210	10	E′
	00	0		65	x		55	÷
	08	8		43	RCL		53	(
	04	4		01	1		01	1
	10	E′		01	1		85	+
065	42	STO	140	95	=	215	03	3
	01	1		44	SUM		10	E′
	00	0		01	1		55	÷
	19	D′		03	3		02	2
	01	1		43	RCL		10	E′
070	42	STO	145	01	1	220	54	)
	00	0		02	2		95	=
	09	9		19	D′		19	D′
	16	A′		01	1		81	HLT
	49	PROD		22	INV			

Labels: A = Ue, B = Uc, C = T, D = R

Registers: 00 = —, 01 = C, 02 = c, 03 = i, 04 = M, 05 = N, 06 = Z, 07 = j

and response time level off. In examples 5 through 8, average I/O access time (i) is successively decreased. Again, CPU utilization and throughput both increase while response time decreases, that is, until the CPU becomes a bottleneck. Terms C and c are increased 100% in example 9. This means that the CPU is twice as slow as in the preceding examples. Example 10 is a uniprocessor (j = 1) with the same speed CPU as described earlier. Apparently, it is more effective to use one fast CPU than two slow ones. This is verified by example 11. In example 12, user think time (Z) is reduced to zero. The number of terminals (N) is expanded to 400 in example 13. Note the increases in response time.

Summary

Detailed discrete models can execute for hours on full size computers before reaching steady-state results. What can be expected of an analytic model run on a handheld calculator? The SR-52 program listing in Table 3 has been optimized for speed and user convenience. The Uc and throughput values are computed in 2 s, response time takes less than 3 s, and these operating times are always constant. Only the calculation time of CPU utilization for the exponential model is variable, depending on level of multiprogramming (M) and multiprocessing (j). Running times for Ue range from 8 s to several minutes. A complex system, multiprocessing with three computers and multiprogramming a combined total of 21 simultaneously active jobs, simulates in 7 min.

Whenever possible, simulations of existing systems should be validated with empirical measurements. By rerunning the model through many iterations and adjusting earlier estimates to bring results closer to observed performance, greater confidence and better insight into the computer system's operation will be gained. After validation fine-tuning, examine the effects of prospective design changes. With a handheld programmable calculator, and this model on a program card, complex design decisions can be based on simulation results, rather than intuition.

Bibliography

J. W. Boyse and D. R. Warn, "A Straightforward Model for Computer Performance Prediction," *ACM Computing Surveys*, June, 1975, pp 73-93

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

C. W. Churchman, R. L. Ackoff, and E. L. Arnoff, *Introduction to Operations Research*, Wiley, 1957, Chap 14

Ronald Zussman holds BSEE, MSEE, and professional EE degrees from Pratt Institute, New York University, and Columbia University, respectively. Currently a senior consultant and project leader of computer performance evaluation, his experience includes the development of interactive models of shipboard computer systems.