Measurement and Modeling of Disk Subsystem Performance

Jozo J. Dujmovic, Daniel Tomasevich, Ming Au-Yeung
Department of Computer Science
San Francisco State University

ABSTRACT. This paper presents techniques for measurement and analytic modeling of disk subsystem performance. We measure disk subsystem performance using benchmark programs and develop an analytic model of the measured system. The model is calibrated and used as a tool for performance prediction. Our goal is to show problems related to practical modeling of real life computer systems and limitations of some traditional modeling techniques. We present models of load-dependent disks, disk cache, and load-dependent Mean Value Analysis models having high predictive power.

1. Introduction

In queuing theory literature there are many models describing dynamic behavior of computer systems. Good sources of such information (e.g. [4,5]) usually include stochastic models based on birth-death formulas, the convolution algorithm [2], load independent and load dependent mean value analysis (MVA) models [7], and BCMP networks [1]. Theoretical queuing models easily explain phenomena such as bottlenecks, saturation, resource utilization etc. However, it is very difficult to find sources that show a second level of modeling, which can be defined as the numerical accuracy in modeling real computer systems running real workloads. While the phenomenology is important in a classroom, it is the numerical accuracy that counts in real life. In practice analysts usually measure system performance, and then derive models that can describe and predict the behavior of analyzed systems with reasonable accuracy. All those who try to model the dynamic behavior or real computer systems running real workloads quickly discover that this is a difficult task. In this paper we present techniques for modeling of real life disk subsystem performance.

2. Performance modeling errors

Differences between queuing theory and experimental results can be quantified as modeling errors. If modeling errors of 30-50% or more are acceptable then many theoretical models can be applied. If modeling errors are required to be less than say 4% then it is rather likely that an effort to develop specific new models will be necessary.

In this paper we present practical techniques for performance modeling and analysis of disk subsystems. Our approach includes the following main steps:

- Specification of drive workload which can be used to create various levels of disk subsystem load.
- Measurement of system performance under a strictly increasing disk subsystem load.
- Development of an analytic model with adjustable parameters that describes the dynamics of a disk subsystem.
- Calibration of the analytic model by adjusting parameters to minimize the difference between the measured values and the values computed from the analytic model.
- Assessment of the predictive power of the analytic model.
- The use of the calibrated model for performance prediction for system with different parameters and or workload.

It is useful to identify two types of modeling errors: *description errors* and *prediction errors*. We define a description error as the mean relative error between the measured performance indicators of a real system and the performance indicators of a calibrated model. The description error is defined only inside the range of measurements. As opposed to that, the prediction error is the error in predicting the values of performance indicators outside the range of measurements, or for different configurations of the analyzed system, or for a different workload. For example, if we measure the response time for the degree of multiprogramming from 1 to 8, then the description error is the mean error between 8 measured values and 8 computed values. The prediction error is the error between the computed response time for 20 jobs and the actual response time if it were measured. The prediction error is also the error between predicted and actual response times for a system with a different number of disks, processors, or for a different workload.

The basic problem of modeling is that the ratio between prediction errors and description errors is usually large, e.g. 2 to 10. This is the reason why description errors must be small, typically just a few percent.

3. A simple model of cached disk access time

The movement of the disk I/O mechanism is usually modeled assuming that movement is caused by applying a constant force for acceleration or deceleration. This is a simple and usually realistic assumption illustrated in Fig. 1. If the mass of the I/O mechanism is *m*, then force *f* causes the acceleration *a=f/m*. After time *t* the mechanism attains the speed *v=at* and travels the distance $x = at^2/2$. After acceleration with force *f* we can apply deceleration with force *-f* and in such a case the total traveled distance is $x = at^2$.

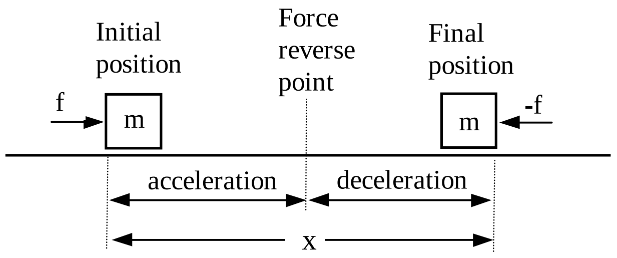

Figure 1. Movement of disk I/O mechanism (seek)

Assuming *f=const.* the time necessary to travel the distance *x* (usually called the *seek time*) is $t = \sqrt{x/a}$. If the maximum distance is $x_{\max}$ then the maximum seek time is $t_{\max}$. So, $a = x_{\max}/t_{\max}^2$ and $t = t_{\max}\sqrt{x/x_{\max}}$. The distance *x* can be expressed as a physical length, or as a number of cylinders traveled (in this case the dimension of acceleration is $cylinder/\sec^2$). If movement of the head is from cylinder *y* to cylinder *z* then the traveled distance is *x*=|*y*-*z*| . The seek time is $t(y,z) = t_{\max}\sqrt{|y-z|/x_{\max}}$. The initial position *y* and the final position *z* can be anywhere in the interval $[0, x_{\max}]$. Therefore, the average seek time is

$$T_{seek} = \frac{1}{x_{\max}^2}\int_0^{x_{\max}}\int_0^{x_{\max}} t(y,z)\,dy\,dz$$
$$= \frac{t_{\max}}{x_{\max}^{5/2}}\int_0^{x_{\max}}\int_0^{x_{\max}}\sqrt{|y-z|}\,dy\,dz = \frac{8}{15}t_{\max} = \frac{8}{15}\sqrt{\frac{x_{\max}}{a}}$$

Each cylinder has a fixed data capacity of *b* bytes. If the file size is *F* then the maximum distance the I/O mechanism can travel is $x_{\max} = F/b$ cylinders. Consequently, the average seek time for this file is $T_{seek} = c\sqrt{F}$, where $c = 8/15\sqrt{ab} = const$. If we access data the range $0 \le F \le F_{\max}$ then the average seek time is the following function of the file size:

$$T_{seek} = T_{\max seek}\sqrt{F/F_{\max}}\ , \quad T_{\max seek} = 8\sqrt{F_{\max}}/15\sqrt{ab}\ .$$

This is a suitable notation because now the constant $T_{\max seek}$ has a suitable interpretation of the maximum value of the average seek time for the file of size $F_{\max}$.

When the I/O mechanism reaches a destination cylinder it is necessary to wait the latency time until the desired sector reaches the read/write head. For the disk that rotates at N_{rev} revolutions per minute the average latency is equal to the half revolution time, i.e. $T_{latency} = 30/N_{rev}$. The disk data transfer time for one sector is $T_{transfer} = 60/N_{rev}N_{sector} = 2T_{latency}/N_{sector}$. Therefore, the mean time to access and transfer disk data from a file whose size is *F* is

$$\begin{aligned} T_{access} &= T_{seek} + T_{latency} + T_{transfer} \\ &= T_{\max seek}\sqrt{F/F_{\max}} + 30(1 + 2/N_{sector})/N_{rev} \\ &\cong T_{\max seek}\sqrt{F/F_{\max}} + 30/N_{rev} \end{aligned}$$

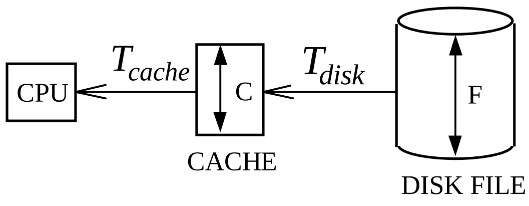

Figure 2. File access using a disk cache

Modern operating systems use main memory as a disk cache. In such a case disk access is performed in a way illustrated in Fig. 2. If data is not in the cache then it is necessary to fetch data from disk. This causes one or more disk accesses, taking time T_{disk}. If data is already in the cache the access is very fast, $T_{cache} << T_{disk}$. The mean access time T_a of a cached disk depends on the cache hit probability *p*:

$$T_a = pT_{cache} + (1-p)(T_{cache} + T_{disk}) = T_{cache} + (1-p)T_{disk}\ .$$

The probability *p* depends on the data access locality properties and for large number of accesses, assuming $C < F$, it satisfies the inequality $p \ge C/F$. The lowest hit ratio, $p = C/F$, is obtained in the case of minimum locality, i.e. in the case of uniform distribution of disk accesses. If $T_{disk} = T_{access}$ then the upper bound of the cached disk access time is a function of the disk file size *F* which can be modeled as follows:

$$T_a(F) = \begin{cases} T_{cache}\ , & F \le C \\ T_{cache} + \left(1 - \dfrac{C}{F}\right)\left[T_{\max seek}\left(\dfrac{F}{F_{\max}}\right)^r + \dfrac{30}{N_{rev}}\right], & F \ge C \end{cases}$$

This model has four adjustable parameters: T_{cache} , *C*, *r*, and $T_{\max seek}$. The case *r=0.5* reflects the ideal situation of constant and perfectly symmetric acceleration and deceleration of disk I/O heads. In real cases the acceleration and deceleration are not identical, and the force that moves heads is not constant. Such cases can be characterized by exponents *r* different from 0.5.

If we measure disk access times $T_1,\ldots,T_n$ for a sequence of file sizes $F_1,\ldots,F_n$ then the calibration of the above model can be performed by selecting the optimum values of T_{cache} , *C*, *r*, and $T_{\max seek}$ which minimize the criterion function

$$E(T_{cache}, C, r, T_{\max seek}) = \sum_{i=1}^{n} |T_i - T_a(F_i, T_{cache}, C, r, T_{\max seek})|^q$$

The exponent *q* is usually selected in the range $1 \le q \le 4$, where larger values are selected in cases where the primary goal is to minimize large errors. The minimization can be performed using the traditional Nelder-Mead simplex method [6].

A verification of this model is shown in Figures 3 and 4 for a 300 MHz PC with 64 MB of memory, Windows NT 4.5, and a 4 GB SCSI disk that has $N_{nev} = 7200$ rev/min. The resulting parameters are $T_{cache} = 96\,\mu\text{sec}$, C = 48 MB, r = 0.234, and $T_{\max seek} = 7.51\,m\text{sec}$ for $F_{\max} = 1400$ MB. In the majority of 245 measured points (only some of them shown in Fig. 3) the mean relative error of this model is less than 1%.

4. Disk access optimization model

The disk access time model described in the previous section assumes a single program that generates disk access requests. In such a case all requests are served strictly in the order they are submitted and disk access optimization is not possible. However, in the case of multiprogramming the disk queue contains multiple requests independently generated by various programs. It is possible to use a disk access optimization algorithm that increases the global disk throughput by minimizing the movement of I/O head mechanism. The simplest such an algorithm is the “shortest seek time first” (SSTF) [8] which can be easily analyzed.

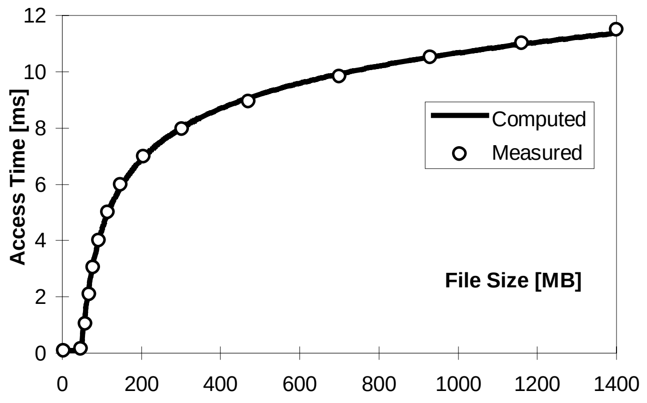

Figure 3. Measured and computed access times

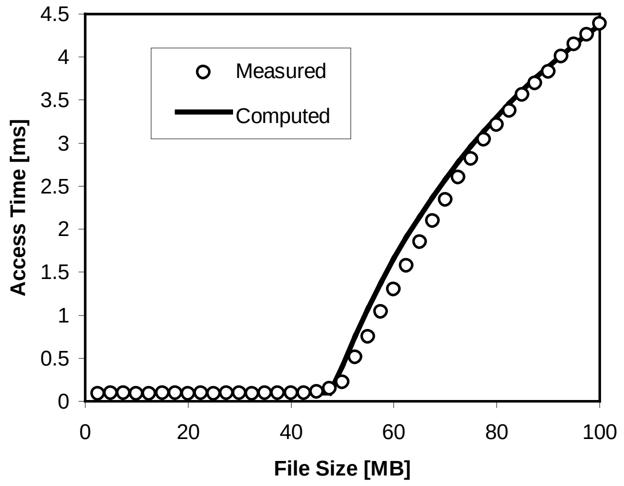

Figure 4. Measured and computed access times showing the cache size of 48 MB (magnified detail of Fig. 3)

A simulator for the SSTF technique can be easily written according to the following algorithm:

1. Create a disk queue with *n* random requests
2. Select an arbitrary initial position of disk heads
3. Find the request that yields the shortest distance *x*
4. Compute the relative seek time $t = \sqrt{x / x_{\max}}$ (this is a normalized time having the largest value 1).
5. The position of the processed request becomes the new position of disk heads .
6. Replace the processed request with a new random request.
7. Repeat steps 3-6 many times and compute the average relative (normalized) seek time.

The results of the simulation yield a decreasing relative seek time function presented in Fig. 5. Taking into account this function, the optimized disk access time can be well approximated using the following model:

$$T_a(n) = t_{\min} + (t_{\max} - t_{\min})e^{\alpha(n-1)}$$

The parameter $t_{\min}$ is introduced to reflect the mean latency time. The value of $t_{\max}$ corresponds to the maximum seek distance. The quality of this approximation is rather good, as shown in Fig. 5. The exponent α has a negative value which reflects the quality of the optimization algorithm (as the quality of optimization increases, so does the absolute value of the exponent α).

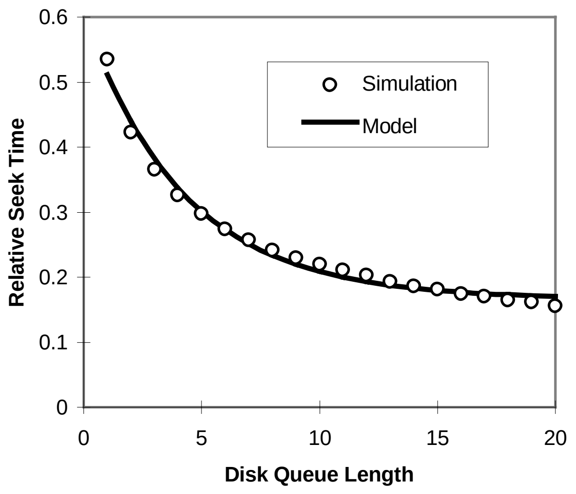

Figure 5. Simulation and analytic model of seek time

5. Disk service time model

Generally, the mean disk access time is a decreasing function of the disk queue length, and an increasing function of file size. If we want to combine the disk cache access model and the disk access optimization model the result can be the following formula for load dependent and cached disk:

$$T_a(F,n) = \begin{cases} T_{cache}\,, & F \le C \\ T_{cache} + \dfrac{F-C}{F}\left[T_{\max seek}\left(\dfrac{F}{F_{\max}}\right)^r e^{\alpha(n-1)} + \dfrac{30}{N_{rev}}\right], & F \ge C \end{cases}$$

However, the disk service time is different from the mean disk access time. Disk service time is affected by caching and access optimization, but it includes only the cases of actual disk access, excluding the cases of fetching data from the cache without disk access. Consequently, we propose the following model of load dependent cached disk service time:

$$S_d(F,n) = T_{\max seek}\left(\frac{F}{F_{\max}}\right)^r e^{\alpha(n-1)} + \frac{30}{N_{rev}} + t_0\,, \quad F > C\,.$$

The first term in this model reflects the optimized seek time, the second term is the latency time, and the third term reflects the data transfer time and related cache I/O operations. This model is successfully applied for the analysis of a cached system presented in Section 8.

6. Disk subsystem benchmark workload

Disk subsystem workload always lies between two obvious extremes: sequential access, and uniformly distributed random access. Sequential access is more frequent, but random access is more general because it includes seek operations. Similarly, benchmark workloads must balance write and read operations. This balance is based on two facts: (1) read operations are generally more frequent than write operations, and (2) write operations are less desirable in benchmarking because performance results are rather sensitive to tuning of disk formats and/or blocking factors, which frequently yields questionable results. Finally, the benchmark workload can be symmetric (i.e. balanced load, where all disks have the same load) and asymmetric (usually with bottleneck disks). Symmetric loads are more desirable in benchmarking because they better expose the capabilities of disk controller(s) and central processor(s).

Table1. Classification of eight basic disk workloads

Workload Type	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Access Method: S = Sequential R = Random	S	S	S	S	R	R	R	R
Operation: W = Write R = Read	W	W	R	R	W	W	R	R
Load balance: S = Symmetric A = Asymmetric	S	A	S	A	S	A	S	A

A simple classification of disk workloads is presented in Table 1. Workloads type 0 or 1 can be used for creating files that are then processed by other workloads. Workloads type 2 and 3 are used for benchmarking systems using sequential access. Similarly, workloads 6 and 7 are used for benchmarking using random access. We use workload type 6 as the basic workload for measurement of disk subsystem performance. It consists of *n* copies of a disk random access program DRAN that uniformly accesses files that are uniformly distributed over all disk units. This is a simple balanced workload that should properly reflect disk subsystem performance and be suitable for both performance measurement and modeling. Typical results of running the DRAN workload type 6 are presented in Figure 6.

The analyzed computer, VAX 8650, has 14 disk units (RA 81) and one or two disk controllers (HSC 50). Each program generates 7000 visits to the central processor, and 7000 visits to disks. Since the disk load is uniformly distributed it follows that each program creates 500 accesses to each disk. In the case of a single program the measured processor time for a single disk controller is 4.99 seconds and the total elapsed time is 163.16 seconds. The processor time can be interpreted as processor demand $D_p = 4.99\,\text{sec}$. A queuing model of this system, in the case of two disk controllers is shown in Fig 7. The next step is to develop an analytic model for the analysis of this system.

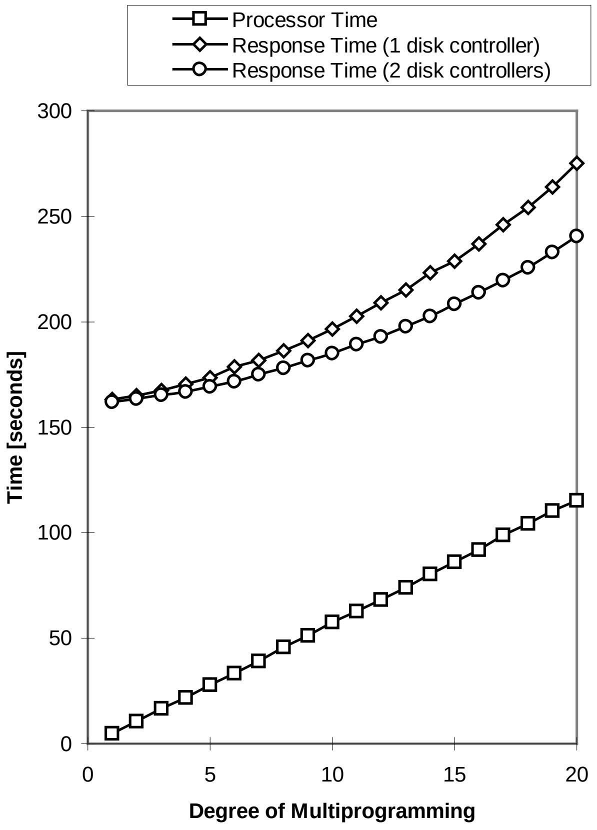

Figure 6. Measured response times for VAX 8650

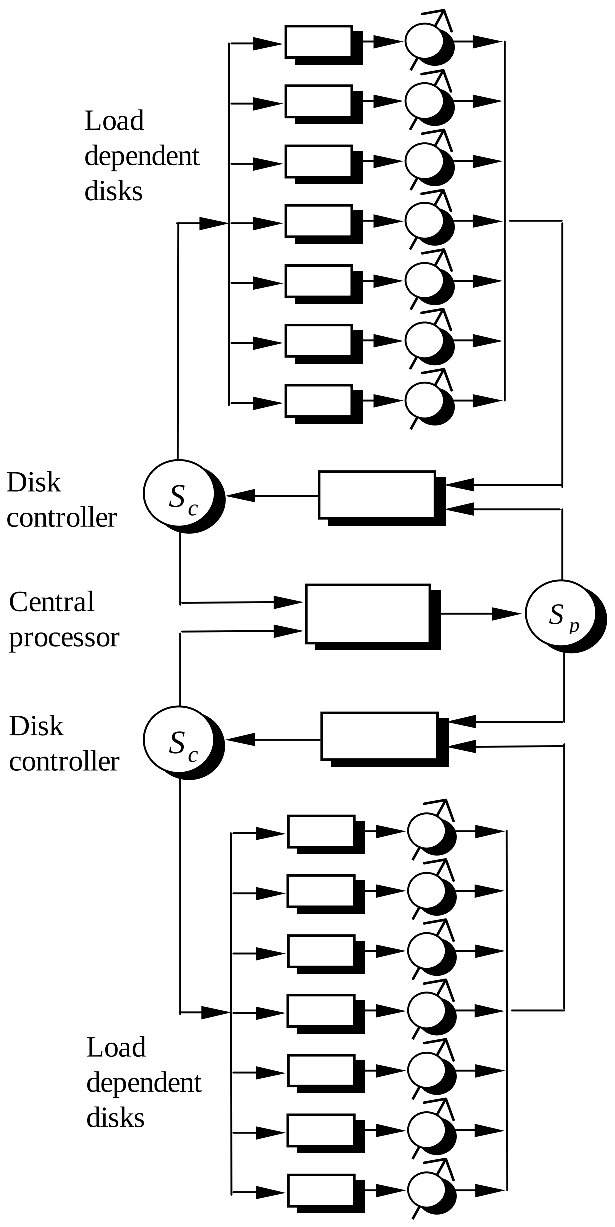

Figure 7. A queuing model of VAX 8650

7. MVA models and their limitations

Let us introduce the following variables:

N = degree of multiprogramming (number of jobs in the system)

K = number of service centers (processors, disks, and disk controllers)

$P_k(j|n)$ = probability that there are *j* jobs at the k^{th} service center, if the total number of jobs in the system (degree of multiprogramming) is *n*

$R_k(n)$ = response time of the k^{th} service center in the case of *n* jobs in the system (*n=1,...,N*)

$R(n)$ = response time of the whole system

$U_k(n)$ = utilization of the k^{th} server

$Q_k(n)$ = queue length at the k^{th} server

$S_k(j)$ = service time of the k^{th} server if *j* jobs are in the queue; if the k^{th} server is load independent then $S_k(j) = S_k = \text{const.},\ k \in \{1,\dots,K\}$

V_k = number of visits to the k^{th} server per job

D_k = service demand for the k^{th} server; in the case of load-independent servers $D_k = V_k S_k$

$X(n)$ = system throughput (completed jobs per time unit)

$X_k(n)$ = throughput of the k^{th} service center when there are n jobs in the system: $X_k(n) = V_k\ X(n)$

The traditional load independent mean value analysis program (LIMVA) is based on assumption that the service times of all servers are constant. The goal of MVA models is to compute system response times, utilizations, and throughputs. Following is the traditional LIMVA model:

```
$Q_k(0) = 0, \quad k = 1, \ldots, K$
for n=1 to N do
    $R_k(n) = S_k[1 + Q_k(n-1)], \quad k = 1, \ldots, K$
    $R(n) = \sum_{k=1}^{K} V_k R_k(n)$
    $X(n) = n / R(n)$
    $X_k(n) = V_k X(n), \quad k = 1, \ldots, K$
    $U_k(n) = S_k X_k(n) = D_k X(n), \quad k = 1, \ldots, K$
    $Q_k(n) = V_k X(n) R_k(n), \quad k = 1, \ldots, K$
end_for
```

For perfectly balanced systems where all demands are equal ($D = V_1 S_1 = V_2 S_2 = \ldots = V_K S_K$) this algorithm yields equal distribution of jobs in service centers $Q_k(n) = n / K$, $k = 1, \ldots, K$. This is a consequence of equal residence times:

$$V_k R_k(n) = D_k[1 + Q_k(n-1)] = D[1 + (n-1)/K],$$
$$k = 1, \ldots, K$$

and their use for computing $Q_k(n) = V_k X(n) R_k(n)$. Furthermore, this yields linear response times, and other relations:

$$R(n) = K V_k R_k(n) = (n - 1 + K)D$$
$$R_k(n) = (n - 1 + K)S_k / K, \quad k = 1, \ldots, K$$
$$X(n) = n / (n - 1 + K)D$$
$$X_k(n) = n / (n - 1 + K)S_k, \quad k = 1, \ldots, K$$
$$U_k(n) = n / (n - 1 + K), \quad k = 1, \ldots, K$$

Of course, in a general case we have different demands, and the response time is no longer linear. Unfortunately, the nature of the LIMVA model is essentially quasi-linear. Even for different demands the response time curves remain similar to straight lines.

The limitations of LIMVA model are exemplified in Fig. 8. In this case the processor service time obtained from measurements is $S_p = 825\,\mu\mathrm{sec}$, and the number of processor visits per job is $V_p = 7000$, yielding processor demand $D_p = 5.775\,\mathrm{sec}$. The number of disks is 14 and the number of disk visits per job is $V_d = 7000 / 14 = 500$. The available resources include the central processor and 14 equally loaded disk units. A spectrum of LIMVA models can be obtained for disk service times varying in the range $12\,m\mathrm{sec} \le S_d \le 24 m\mathrm{sec}$, and yielding disk demands in the range $6\,\mathrm{sec} \le D_d \le 12\,\mathrm{sec}$. The corresponding response times presented in Fig. 8 are in the whole range practically straight lines and obviously inadequate for representing the measured response time function. The best approximation would be obtained for $S_d = 16\,m\mathrm{sec}$, but this approximation is equally poor as the attempt to use a straight line to approximate a parabola. The nature of the dynamic behavior of VAX 8650 is quite different from what can be modeled by LIMVA regardless how well we adjust its parameters. Consequently, a more flexible model is needed. Taking into account that disks are never load-independent, because they regularly use either access optimization or access optimization and caching, it follows that better results should be expected from load dependent MVA models.

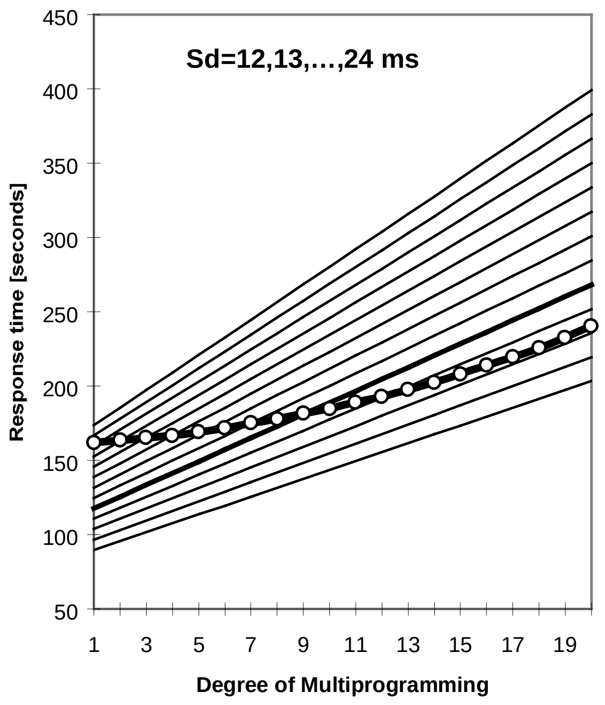

Figure 8. A family of LIMVA models and measured VAX 8650 response time

For batch systems following is the load dependent mean value analysis model (LDMVA) introduced in [1] (see also [4,5]):

```
$P_k(0|0) = 1, \quad k = 1,\dots,K$
for n=1 to N do
    $R_k(n) = \sum_{j=1}^{n} jS_k(j)P_k(j-1|n-1), \quad k = 1,\dots,K$
    $R(n) = \sum_{k=1}^{K} V_k R_k(n)$
    $X(n) = n / R(n)$
    $P_k(j|n) = \begin{Bmatrix} V_k S_k(j) X(n) P_k(j-1|n-1), & j = 1,\dots,n \\ 1 - \sum_{i=1}^{n} P_k(i|n), & j = 0 \end{Bmatrix},$
            $k=1,\dots,K$
    $Q_k(n) = \sum_{j=1}^{n} jP_k(j|n), \quad k = 1,\dots,K$
    $U_k(n) = 1 - P_k(0|n), \quad k = 1,\dots,K$
end_for
```

8. Experimental results for LDMVA model of a disk subsystem with access optimization

Let us now apply the LDMVA model for VAX 8650 disk subsystem modeling. The measured response and processor times can be used to adjust parameters of the LDMVA model. This is called the calibration of queuing model. The calibration process is based on measured response times $T_{m1}(n)$ and $T_{m2}(n)$ which correspond to systems with one and two disk controllers. Suppose that the load dependent disk service time is approximated by $S_d(n) = t_{\min} + (t_{\max} - t_{\min})e^{\alpha(n-1)}$. The parameters of the LDMVA model are $t_{\min}, t_{\max}, \alpha, S_p, S_c$. Let $R_{c1}(n, t_{\min}, t_{\max}, \alpha, S_p, S_c)$ and $R_{c2}(n, t_{\min}, t_{\max}, \alpha, S_p, S_c)$ be the response times computed from the LDMVA model. The model calibration procedure is based on the minimization of the compound criterion function

$$C(t_{\min}, t_{\max}, a, S_p, S_c) = \max\left(\sum_{i=1}^{n} [T_{m1}(n) - R_{c1}(t_{\min}.t_{\max}, a, S_p, S_c)]^2, \sum_{i=1}^{n} [T_{m2}(n) - R_{c2}(t_{\min}.t_{\max}, a, S_p, S_c)]^2 \right)$$

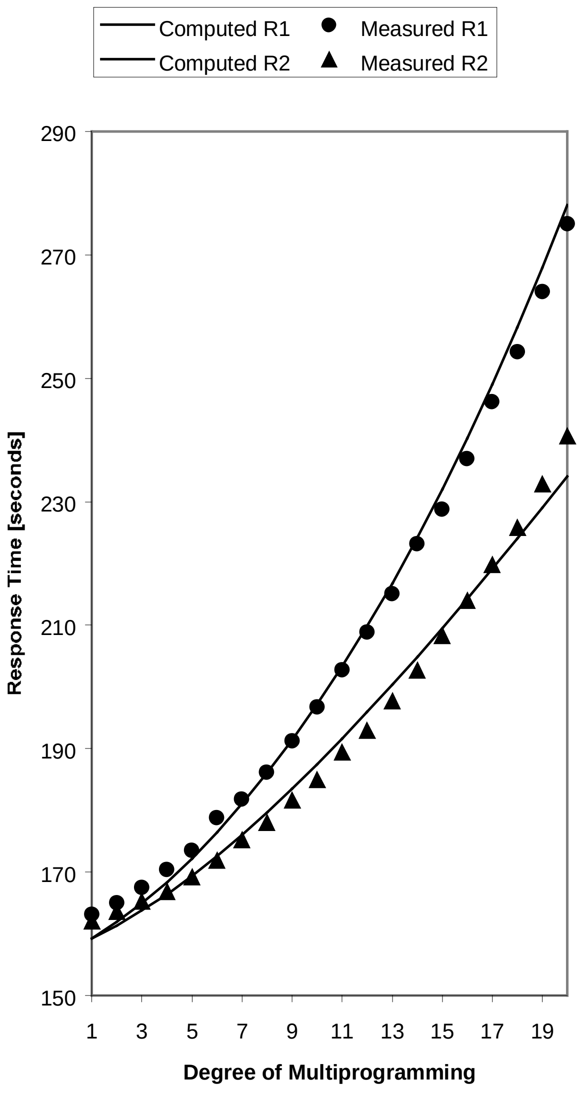

Figure 9. The results of LDMVA model calibration

The results of calibration performed using the Nelder-Mead simplex method [6] are the following values of parameters: $t_{\min} = 11.5\,m\sec$, $t_{\max} = 20\,m\sec$, $a = -4$, $S_p = 0.822\,m\sec$, and $S_c = 1.89\,m\sec$. The mean relative error for all presented points is 1% (Fig. 9). At this level of description error it is realistic to expect good prediction results.

9. Experimental results for LDMVA model of a disk subsystem with caching and access optimization

In this experiment we use the same 300 MHz PC presented in Section 3 and Figures 3 and 4. Its memory capacity is 64 MB and the disk cache size under Windows NT 4.5 is C=48 MB. It uses two disk units.

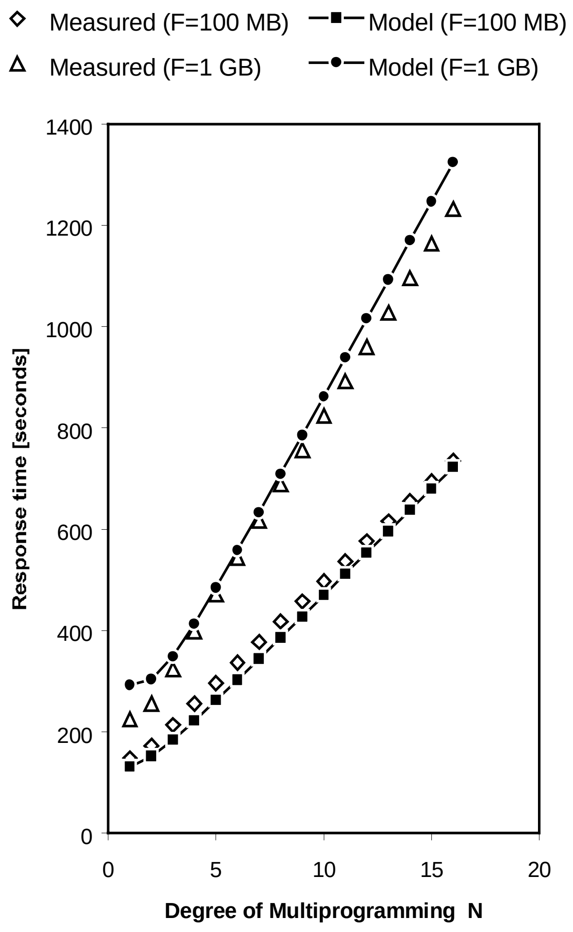

Figure 10. Measured response times and the results of the LDMVA model

Measured response times for DRAN benchmarks accessing a 100 MB file and 1 GB file are presented in Fig. 10. The LDMVA model we used is based on measured disk parameters reported in Section 3 and on disk service time model $S_d(F,n)$ proposed in Section 5. Disk accesses to a file having size *F* occur with the probability *(F-C)/F* and cache accesses with the probability *C/F*. The number of disk visits V_d now depends on the number of processor visits V_p and the size of file. If the number of disks is *k* then $V_d(F) = V_p(F-C)/kF$. Therefore, the disk cache causes that both the disk service time and the number of visits are functions of file size. Processor service time is not constant. For cache accesses it can be expressed as $S_p = t_p^{prog} + t_p^{cache}$ and for disk accesses as $S_p = t_p^{prog} + t_p^{cache} + t_p^{disk}$, where the three components correspond to the processor activity for benchmark program, cache access, and serving the file management system during the disk access. The mean service time is $S_p = t_p^{proge} + t_p^{cache} + t_p^{disk}(F-C)/F$. The calibrated model in Fig. 10 has the mean modeling error of 7.4%.

10. Conclusions

Even in cases of fully controlled simple synthetic workloads the performance modeling of computer systems is not a simple task. Basic popular load independent queuing models (load independent convolution algorithm or MVA) cannot be used for modeling modern computer systems which use caching and disk access optimization.

The use of load dependent MVA models is rather efficient and we proposed models for optimized disk access, cached disk access, and combined optimized and cached disk access. Presented models need a careful calibration procedure. In the majority of cases based on symmetric random disk accesses the modeling errors for optimized disk access were less than 2%. In the more complex cases with optimized disk accesses and caching our models regularly achieve errors below 10%. Experimental verification of our models was successfully performed in both VAX/VMS and PC/NT environments.

References

1. Basket, F., K. Chandy, R. Munz, and F. Palacios, *Open, Closed, and Mixed Networks of Queues with Different Classes of Customers.* Journal of the ACM, Vol. 22, No. 2, pp. 248-260, April 1975.
2. Buzen J.P., *Computational Algorithms for Closed Queueing Networks with Exponential Servers.* Communications of the ACM, Vol. 16, No. 9, pp. 527-531, September 1973.
3. Dujmovic, J.J., *Multiprogramming Efficiency Analysis for Computer Evaluation and Selection Studies.* Proceedings of the 11th International Symposium Computer at the University, 1989.
4. Jain. R., *The Art of Computer System Performance Analysis.* John Wiley, 1991.
5. Menasce, D. , V. Almeida, and L. Dowdy, *Capacity Planning and Performance Modeling.* Prentice Hall, 1994.
6. Nelder J.A. and R. Mead, *A Simplex Method for Function Minimization.* The Computer Journal, Vol. 7, No. 4, pp. 308-313, 1965.
7. Reiser, M. and S.S. Lavenberg: *Mean-Value Analysis of Closed Multichain Queueing Networks.* Journal of the ACM, Vol. 27, No. 2, pp. 313-322, 1980.
8. Silbershatz, A. and P. B. Galvin, *Operating System Concepts.* Addison-Wesley, 1994.