THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY[1]

NEW PERFORMANCE MEASURES FOR DESIGN AND EVALUATION OF REAL-TIME MULTIPROCESSORS

Kang G. Shin and C. M. Krishna

CRL-TR-33-83

NOVEMBER 1983

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

[1]This work was supported in part by National Aeronautics and Space Administration Grant No. NAG 1-196. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the view of NASA.

ABSTRACT

Conventional performance measures such as throughput, reliability, and availability are not by themselves alone suitable for analyzing and designing real-time multiprocessors. (The real-time system we focus on here consists of a *computer controller* and *controlled processes*). Most of these measures treat the computer as a monolith in terms of the services it delivers, thus failing to accurately describe the nature of diversified real-time application tasks.

As a remedy for this problem, this report presents some new performance measures based on computer(multiprocessor) controller response time that are (i) *congruent* to the real-time applications, (ii) able to offer an *objective comparison* of rival computer systems, and (iii) experimentally *measurable/determinable*. These measures, unlike others, provide the real-time multiprocessor controller with *a natural link* to controlled processes.

In order to demonstrate their utility and power, these measures are first determined for an example controlled process on the basis of two control performance functionals. They are then used for an important real-time multiprocessor design application -- the number-power tradeoff. Also, some potential applications of these measures are identified.

Index Terms --Performance measures, real-time multiprocessors, controllers and controlled processes, response time, cost functions, dynamic failure, hard deadlines, number-power tradeoff.

1. INTRODUCTION

Like every other medium of expression, performance measures for computers influence *what* is expressed almost as much as *how* it is expressed. Analyses of computer systems therefore depend greatly for their precision and relevance upon the performance measures used.

In this report, we consider computer systems used in real-time control of critical processes. A critical process is one whose failure can lead to loss of life, or other catastrophe. Such processes include aircraft, spacecraft, manufacturing processes, and power generation and distribution. They cover a wide spectrum, but have the following in common: in all of them computers are used in control, in all of them the job load of the control computer is precisely characterized, the characteristics of the operating environment are known *a priori*, performance criteria for the controlled processes are well-defined, their complexity is growing rapidly, and some of these processes are likely to be so complex in the near future that manual over-ride of the controlling computer is likely to be of no use: no human operator could possibly react quickly enough to stimuli from the controlled process. Nowhere is this more true than in the aerospace field, where the drive for improved fuel efficiency is expected by the beginning of the next century to lead to civilian aircraft which are intrinsically unstable, and which require "active controls" to keep them from crashing.

The requirements such applications impose upon the computer are stringent. They must be extremely robust, and capable of low response times. Since it is their performance within the context of the environmental demands that is crucial, the characteristics of the operating environment must be incorporated into any model of computer behavior. To express the capabilities of a real-time computer in such a context, perfor-

mance measures must quantify such characteristics as graceful degradation, throughput, reliability, availability, and the operating environment, in the form of a unified metric.

Performance measures that at least partially meet these requirements have been suggested by the following authors. Beaudry [1] considers measures emanating from the volume of computation from a computer system over a given period of operation. Mine and Hatayama [2] consider *job-related reliability,* by which they mean the probability that the system will successfully complete a certain job. Huslende [3] attempts to be as general as possible, and presents what amounts to a re-statement of Markov modeling with traditional measures. Chou and Abraham [4] present performance-availability models, Castillo and Sieworek [5,6] performance-reliability models. Osaki and Nishio [7] consider the "reliability of information", by which they mean the expected percentage of wrong outputs per unit time in steady state.

All these measures consider the computer system in isolation, i.e. without explicit regard to the requirements of the operating environment. For this reason, they are quite unsuitable for use in assessing real-time control computers.

Among all existing performance measures, Meyer's *performability* [8] seems to meet, though in an abstract form, the real-time requirements discussed above. His measure explicitly links the application with the computer by listing "accomplishment levels", which are expressions of how well the computer has performed within the context of the application. His work focuses on the development of a framework for modeling and performance evaluation, rather than on a method for deriving the performance measures themselves. No guidelines are given for appropriately specifying the accomplishment levels: what is provided is a set of mathematical tools for their computation, once they have been defined. Some recent work by Meyer [9] continues this trend, developing the theory

of stochastic Petri nets.

By contrast we focus, in this report, on presenting a methodology for objectively characterizing real-time computer performance. If one wished to translate our work into the terms of Meyer's performability, we show how to derive a set of uncountably many accomplishment levels that are completely objective and capable of definitive estimation and/or measurement. It is this that makes our work complementary to that of Meyer.

This report is organized as follows. In Section 2, we introduce our performance measures. Then, these measures are derived for example systems, thereby showing the link they form between a controlled process and its controlling computer. In Section 3, we trade the number of processors against their individual power. This is used as a vehicle to describe an application of these measures. Further applications are suggested in Section 4, with which this report concludes.

2. THE PERFORMANCE MEASURES

2.1. Terminology and Notation

A real-time computer executes pre-defined control jobs repeatedly, upon environmental or other stimuli. *System response time* is defined as the time between initiation/triggering of a control job and the actuator and/or display output that results. This quantity is the sum of *controller response time* and *actuation time*. Environmental or other occurrences trigger the tasks, a unique *version* being created as a result. This is said to be an *extant version* as long as it continues to execute in the system. Versions of task i are denoted by V_{ij}, which represents the j-th execution of task i. Denote the response time of a no-longer-extant version V_{ij} by RESP(V_{ij}). The response time of an extant version is undefined. The *extant time* of a version V_{ij} triggered at time τ_{ij} when

the

system is in state n_{ij} is given by $\Xi(V_{ij},\tau_{ij},n_{ij},t) = \min(t-\tau_{ij},\ RESP(V_{ij}))$.[1]

A controller task is said to be *critical* if it has an associated *hard deadline* [10], which if exceeded by any of its versions, results in catastrophic or *dynamic* failure. Hard deadlines do not exist for *non-critical* tasks.

Ordinarily, repair to the real-time computer is not allowed while the computer is in operation. In this connection, we define the *mission lifetime* as the duration of operation between successive stages of service. We let the mission lifetime be a random variable with probability distribution function $L(t)$. At the beginning of a mission (i.e. immediately after service), a system is assumed to be free of faults.

2.2. Definition of the Performance Measures

Our performance measures are all based on the extant and response times. For critical tasks i, with hard deadlines t_{di}, the cost function is defined by:

$$C_i(\Xi) = \begin{cases} g_i(\Xi) & \text{if } 0 \leq \Xi \leq t_{di} \\ \infty & \text{if } \Xi > t_{di} \end{cases} \tag{1}$$

where g_i is called the *finite cost function* of task i, and is only defined in the interval $[0,\ t_{di}]$, and we omit for notational convenience the arguments of Ξ, extant time.

For non-critical tasks, the same definition for the cost function can be used, with the associated hard deadline set at infinity.

Let $q_i(t)$ denote the number of times task i is initiated in the interval $[0,t)$. Then, the *cumulative cost function* for the task i is defined as

[1] This does NOT imply that no state changes occur during the course of a task execution. $RESP(V_{ij})$ is an implicit function of n_{ij}.

$$\Gamma_i(t) = \sum_{j=1}^{q_i(t)} C_i(\Xi(V_{ij}, \tau_{ij}, n_{ij}, t)) \tag{2}$$

and the *system cost function* is defined as

$$S(t) = \sum_{i=1}^{r} \Gamma_i(t) \tag{3}$$

where r is the number of tasks in the system. Both Γ_i and S are clearly defective random variables. Our performance measures are then given by:

$$\textit{Cost Index, } K(\chi) \equiv \int_0^\infty \textit{Prob}\{S(t) \leq \chi\}\, dL(t) \tag{4}$$

$$\textit{Probability of dynamic failure, } p_{dyn} \equiv \int_0^\infty \textit{Prob}\{S(t) = \infty\}\, dL(t) \tag{5}$$

$$\textit{Mean Cost, } M \equiv \int_0^\infty E\{S(t) \mid \textit{no hard deadlines are missed}\}\, dL(t) \tag{6}$$

$$\textit{Variance Cost, } V \equiv \int_0^\infty \textit{Var}\{S(t) \mid \textit{no hard deadlines are missed}\}\, dL(t) \tag{7}$$

where $E\{\bullet|\bullet\}$ and $Var\{\bullet|\bullet\}$ represent conditional expectation and variance, respectively. The probability of dynamic failure subsumes the traditional probability of failure (called here for distinction the *probability of static failure*) since the latter can be viewed as the probability that the expected system response time is infinity. Clearly, in the case of non-critical tasks, the probabilities of static and of dynamic failure are equal.

The following auxiliary measures are useful when one focuses on the contribution to the cost of individual tasks.

$$\textit{Cost Index for Task } i, K_i(\chi) \equiv \int_0^\infty \textit{Prob}\{\Gamma_i(t) \leq \chi\}\, dL(t) \tag{4a}$$

$$\textit{Mean Cost for task } i,\ M_i \equiv \int_0^\infty E\{\Gamma_i(t) \mid \textit{no hard deadlines are missed}\}\, dL(t) \qquad (5a)$$

$$\textit{Variance Cost for task } i,\ V_i \equiv \int_0^\infty Var\{\Gamma_i(t) \mid \textit{no hard deadlines are missed}\}\, dL(t) \qquad (7a)$$

The computation of these measures can sometimes be complicated by the fact that the mission might end while one or more versions are still extant. In most instances, however, the mission lifetimes are very much longer than individual task execution times[2] and the number of times tasks are executed to completion before the mission ends is also very large. For this reason, it is usually an acceptable approximation to compute the costs assuming that all jobs that enter the system during the mission complete executing before the mission ends (as long as they do not miss any hard deadlines).

In what follows, we consider how to determine these performance measures, beginning with the determination of the hard deadline.

2.3. Obtaining the Hard Deadline

The dynamics and the nature of the operating environment of the critical process are both known *a priori*. This follows from the critical nature of the process -- for example, the dynamics and operating environment of aircraft have both been studied carefully -- and advances in the theory and design of controlled processes.

The process can most conveniently be expressed by a state-space[3] model. Let $\mathbf{x}\in\mathbf{R}^n$ denote the process state, $\mathbf{u}\in\mathbf{R}^m$ the input vector, and t the time. The input vector is made up of two sub-vectors, $\mathbf{u}_c\in\mathbf{R}^{m_c}$ and $\mathbf{u}_e\in\mathbf{R}^{m_e}$. $\mathbf{u}_c$ denotes the input delivered at the

[2]For example, it could be several hours for aircraft and several days or even months for spacecraft.

[3]The term "state" here has the control-theoretic meaning, and is *not* the same as the one frequently used in computer performance analysis.

command of the computer, and $\mathbf{u}_e$ the input generated and then applied by the operating environment. Also, let $\mathbf{y} \in \mathbf{R}^p$ denote the measurement or output vector. Then, the control system can be represented by a pair of vector equations; a state vector equation,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t),t) \tag{8}$$

and an output equation,

$$\mathbf{y}(t) = \boldsymbol{\eta}(\mathbf{x}(t),\mathbf{u}(t),t) \tag{9}$$

Catastrophic failure can follow if the process leaves the "safe" region of the state space. For example, a boiler may explode if its temperature becomes too high. This is formally expressed by defining an *allowed state space,* $\mathbf{X}_a(t)$, which defines the "safe" region of operation.

The task of the controller or real-time computer is to derive the optimal control, $\mathbf{u}_c(t)$, as a function of the perceived process state. Since the response time, denoted by ω, is positive, we have $\mathbf{u}_c(t) = \mathbf{h}(\mathbf{x}(t-\omega),\mathbf{u}_c(t-\omega),t)$ where $\mathbf{h}$ expresses the control algorithm for the task in question. Then, the hard deadline associated with this task is given by the maximum value of ω that may be permitted if the process is to remain in $\mathbf{X}_a$ with probability one. The hard deadline, defined in this general way, is a function both of the process state at the moment of task initiation, and of time. It is a random variable if the environment is stochastic. Consider the following example to see how the hard deadline can be determined.

Example 1: A body of mass m is constrained to move in one dimension. Its state-vector consists of three components: position (x_1), velocity (x_2), and acceleration (x_3). The allowed state-space is defined by $\mathbf{X}_a = \{\mathbf{x} \mid |x_1| \leq b,\ x_2 < \infty,\ x_3 < \infty\}$ where $b > 0$ is a constant. The body is subject to impact from either direction with equal probability. Each

impact changes the velocity of the body by $k>0$ units, in the appropriate direction. The change of velocity takes place in a negligible duration. The body has devices that can exert thrust of magnitude H in either direction. This thrust is imposed only after the controller has recognized an impact and has determined how to react to it. It takes a negligible amount of time to switch the thrust on or off in either direction. The controller's job is to bring the body to $\mathbf{x}=0$. The controller operates in open-loop; when it recognizes an impact, it computes the thrusts as a function of time, following which the control response is assumed to be instantaneous. The problem now is to compute the hard deadline associated with this task.

The hard deadline is only a function of the state and $\mathbf{X}_a$. In computing it, we do not need to take into account the possibility of a second impact before the controller has finished responding to the first, since the state of the process contains all necessary information.

The allowed state space is static and simply connected, so that if when the body is brought to rest for the first time following the impact it is in $\mathbf{X}_a$, it must have been within $\mathbf{X}_a$ throughout the period following the impact, assuming only that it was within $\mathbf{X}_a$ at the moment of impact. Therefore, we have only to compute the position of the body when it first comes to rest (after the impact) as a function of the response time, ξ, and set the hard deadline equal to the largest ξ for which the body comes to rest within $\mathbf{X}_a$. Let the initial state of the body be $\mathbf{x}_i^T=[x_{1i}, x_{2i}, x_{3i}]$. Since the impact duration and the switch-off or on time for the thrust are assumed to be zero, we can always take $x_{3i}=0$. Define

$$t_1 = \left| \frac{m}{H}[x_{2i}+k] \right|, \qquad t_2 = \left| \frac{m}{H}[x_{2i}-k] \right|$$

By an elementary derivation, we arrive at the following.

Case 1, $x_{2i}<-k$:

$$t_d(\mathbf{x}_i)=\min\left\{\frac{b+x_{1i}+(x_{2i}+k)t_1+\frac{Ht_1^2}{2m}}{-(x_{2i}+k)},\ \frac{b+x_{1i}+(x_{2i}-k)t_2+\frac{Ht_2^2}{2m}}{k-x_{2i}}\right\} \tag{10}$$

For future convenience, denote the right hand side of the above by $t^{(1)}$.

Case 2, $|x_{2i}|<k$:

$$t_d(\mathbf{x}_i)=\min\left\{\frac{b-x_{1i}-(x_{2i}+k)t_1+\frac{Ht_1^2}{2m}}{x_{2i}+k},\ \frac{b+x_{1i}+(x_{2i}-k)t_2+\frac{Ht_2^2}{2m}}{k-x_{2i}}\right\} \tag{11}$$

Denote the right hand side of the above equation by $t^{(2)}$.

Case 3, $x_{2i}>k$:

$$t_d(\mathbf{x}_i)=\min\left\{\frac{b-x_{1i}-(x_{2i}+k)t_1+\frac{Ht_1^2}{2m}}{x_{2i}+k},\ \frac{b-x_{1i}-(x_{2i}-k)t_2+\frac{Ht_2^2}{2m}}{x_{2i}-k}\right\} \tag{12}$$

Denote the right hand side of the above equation by $t^{(3)}$.

If the velocity imparted to the body upon an impact is not constant at k, but is a random variable (which would be more realistic) the magnitude of which has probability distribution function F_{impact}, then the hard deadline will be a random variable, whose distribution F_d is a function of the state at the moment of impact. The following can be written down by inspection:

Case 1, $x_{2i} < 0$:

$$t_d(\mathbf{x}_i) = \begin{cases} t^{(2)} & \textit{with probability } F_{impact}(|x_{2i}|) \\ t^{(1)} & \textit{with probability } 1 - F_{impact}(|x_{2i}|) \end{cases} \tag{13}$$

Case 2, $x_{2i}>0$*:*

$$t_d(\mathbf{x}_i) = \begin{cases} t^{(2)} & \textit{with probability } 1 - F_{impact}(|x_{2i}|) \\ t^{(3)} & \textit{with probability } F_{impact}(|x_{2i}|) \end{cases} \tag{14}$$

We could similarly treat the case when the allowed state-space is stochastic.

The above example is meant only to illustrate the hard deadlines and should not lull the reader into a false sense of security. Obtaining closed-form expressions for the deadlines of any but the most trivial systems and static allowed space is usually extremely difficult, if not impossible. For example, if we relax the assumption that the controller acts in open-loop, the equations of motion become too difficult to solve exactly in closed form.

It is generally necessary to resort to numerical methods to obtain deadlines for real-life systems. Since most of the state-spaces one uses in practice have uncountably many points, we must define hard deadlines as functions of sets of states, not of the states themselves if the entire allowed state-space is to be covered. Subdividing the state-space into these subsets while keeping errors low is not always easy. For an example of subdivision where the application is the control of aircraft elevator deflections, see [11].

A further remark is in order here. The hard deadlines are not dependent upon the performance functional (time, energy, etc.) that the controller is attempting to optimize, since the paramount duty of the controller is to keep the system within the allowed state-space, and only secondarily to optimize the performance functional.

2.4. Obtaining the Finite Cost Functions

Performance functionals have been known for a long time in control theory as optimization criteria and measures of controlled process performance. We exploit this fact to derive the control-computer cost functions, by linking directly the performance of the controller to the value of the controlled process performance functional that results.

Performance functionals in control theory are functionals of system state and input and express the cost of running the process over some interval $[t_0, t_f]$. The performance functionals can be stated as:

$$\Theta(\mathbf{x}_0, t_0, t_f) = \int_{t_0}^{t_f} E\left[f_0(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}_0, t) \mid \mathbf{y}(\tau), t_0 \leq \tau < t \right] dt \tag{15}$$

where $\mathbf{x}(t_0) = \mathbf{x}_0$, and f_0 is the instantaneous performance functional at time t. Since the controller response time affects the state trajectory of the controlled process, it affects the performance functional as well. If we use the expected contribution to the performance functional, $\Theta(\mathbf{x}_0, t_0, t_f)$, of the control delivered as a result of executing task i with response time ξ, we can derive the finite cost function as:

$$g_i(\mathbf{x},\xi) = \begin{cases} \Omega(\mathbf{x},\xi) - \Omega(\mathbf{x},0) & \text{for } 0 \leq \xi \leq t_{di} \\ 0 & \textit{otherwise} \end{cases} \tag{16}$$

where $\Omega(\mathbf{x},\xi)$ denotes the contribution to $\Theta(\mathbf{x}_0, t_0, t_f)$ of a task with response time ξ, and initiated when the process state was $\mathbf{x}$. By doing so, we can directly couple the response time of the controller to the fuel, energy, or other commodity by the controlled process.

Notice that while we use response time to compute the cost function, the finite costs were originally defined as functions of the extant time. The latter is the case since we wish costs to accrue as the execution proceeds, so that the system cost function is continuous. This ensures if two systems are compared under an identical load with the

first faster than the second with regard to a particular task, that the faster system will never exhibit a mean cost greater than the slower system as long as both have approximately the same probability of dynamic failure.

The possibility of correlation of successive tasks can complicate calculations considerably. To see this, take the system in the example above. If a second impact comes in *before* the system has finished reacting to the first, then, the energy or time to be expended will not, in general, be the sum of the energies or the times that would have to be expended if the second impact had arrived *after* the system had finished reacting to the first (i.e. had arrived at x=0). Assuming that successive tasks are decoupled leads to a certain measure of "double-counting" of the energy or time spent. The same remark would apply to fuel, force, or any other performance functional used for the controlled process.

Due to this double-counting, assuming that successive tasks are decoupled leads to an upper bound to the energy, time, or other quantity expended. If we find an upper bound acceptable, we can simplify our computations greatly. If exact figures are called for, a detailed and complicated model has to be worked out in which each instance of inter-task coupling is itemized and its probability of occurrence computed. Whether or not this is worth the effort depends entirely on the requirements of the analysis.

There is also an irritating anomaly. Since the mean costs are defined by an expectation that is conditioned on not failing in the mission lifetime, it is possible to construct pathological examples where a system with a probability of dynamic failure of, say, 0.5 over a given lifetime, will exhibit a *lower* mean cost over that lifetime than another that has a p_{dyn} of 10^{-10}: we shall see examples of them in Section 3. Such cases are, however, generally no more than an academic curiosity.

Example 2: Consider again the controlled process described in Example 1. This time, we set out to compute the finite cost function associated with the task under review.

As before, assume that the state of the body at the moment of impact is given by $\mathbf{x}^T=[x_{1i},x_{2i},x_{3i}]$. We make the assumption that a function that provides an upper bound of the cost expended is sufficient so that it is not necessary to consider the correlative effect of successive jobs. We provide cost functions relating to two different control policies.

Case A: Assume that the duty of the controller is to bring the body back to $\mathbf{x}=0$ within as short a time period as possible. (Note that $\mathbf{x}=0$ means that all three components -- position, velocity, and acceleration -- are zero). The cost function is the time taken. This is the well-known minimum-time problem in optimal control theory [13]. If, after the impact, the body is moving away from $x_1=0$, it must be stopped, and brought back using bang-bang control. If it is moving toward $x_1=0$, depending on the velocity after impact and the response time of the controller, the body is either first accelerated toward $x_1=0$ and then decelerated, or first brought to a stop on the other side of $x_1=0$ and then brought back to the origin using bang-bang control. The derivation of the time taken is elementary, if tedious, and is excluded. See the Appendix for expressions of the finite cost function under such a control policy. The case when the velocity imparted upon impact is not constant, but a random variable, can be handled as in Example 1.

Case B: Suppose the controller is to minimize the energy expended while, after every impact, keeping the body within the allowed state-space. Then, the control policy is simply to bring the system to rest anywhere inside $\mathbf{X}_a$, and the cost function is in terms of energy. If the controller computer responds to an impact within the hard deadline, it

can by definition, keep the system from failing. As may easily be verified, the energy expended in doing this is the energy required to bring the body to rest, which is equal to the energy of the body immediately after impact. Therefore, as long as the allowed state-space is not violated, the energy expended will remain the same no matter what the response time (assuming that the response time is within the hard deadlines derived in the preceding section). So, the finite cost function over the entire allowed state-space is here identically zero, which signifies that, as long as the hard deadlines are honored, it makes no difference to the overhead *under this control policy,* as to what the response time may be.

This example has served to emphasize the intimate relation between control policy and controller (finite) cost function. The *same* system, with the *same* constraints on the allowed state-space, has different cost functions based on what the duty of the controller is. It reflects our goal of having the cost functions express the control overhead *in the context of the application.*

Once again, it should be noted that the simplicity of the above expository examples does not usually exist in real-life systems. Real-life analyses are much more difficult, and the same comments as applied to p_{dyn} above apply to the mean cost, too. There is also one additional complication. The controller is to optimize the performance functional subject to the condition that the system must not leave the allowed state-space, if this is at all possible. Such a difficulty did not arise in this simple example, but it can sometimes prove difficult to design optimal control policies under this requirement. This, however, is a problem for the designer of the controlled process, not of the controlling computer.

Also, see [11] for a computation of the cost function in a realistic case.

2.5. Remark on Finite Cost Functions

It is not necessary that the process performance functional that is used to derive the cost function (e.g. energy, fuel, time, etc.) be the same as the process performance functional that the system is trying to optimize. For example, the system may be given the task of optimizing fuel, and the cost function may be measuring the extra energy consumed as a result of controller delay. However care should be taken to ensure that the two functionals (the one used to optimize the process, and the one used to express the cost with) do not conflict. For the functionals not to conflict, the optimal control actions (i.e. the controller decisions) taken on the basis of one functional should be identical to the optimal control actions that would have been taken on the basis of the other. For example, if the fuel consumed were linearly related to the energy expended, the cost function could be expressed in terms of energy, while the controller was trying to minimize the fuel used.

To see why conflicts must not be allowed, assume in the system of the above example, that the job of the controller is to minimize the time taken in bringing the body back to the origin, while the cost function is in terms of energy. Take the instance in which the speed of the body after the impact is a slow motion toward $x_1=0$. The controller should in such a case apply full thrust throughout the motion, first speeding the body up toward $x_1=0$, and then slowing it down to reach $x=0$ in minimum time. However, since the cost function is in terms of energy, not time, it is easy to see that the shorter the time period over which the controller exerts thrust, the smaller is the value of the cost measured. Thus, over a certain range of states, the cost function would actu-

ally *decrease* with an increase in response time. This is not only counter-intuitive, but also results in inefficient operation. Task priorities, scheduling policies, etc., for the control computer are meant to be derived to optimize the mean cost as expressed by the cost functions. If inconsistencies such as the above arise, the operating system of the controller would tend to oppose the goals inherent in its own applications software, resulting in an unsatisfactory overall computer-controlled system.

In what follows, we consider how to apply our performance measures to the design of real-time multiprocessors.

3. THE NUMBER-POWER TRADEOFF AND ITS APPLICATION TO COMPUTER DESIGN

The number-power tradeoff problem can be stated as follows [12]. It appears intuitively obvious if there are no device failures, that a system with a *single* processor with mean service time (exponentially distributed) of $1/\mu$ is more efficient than an N-processor ($N>1$) system with each processor providing exponential service at rate μ/N. When failure is allowed for in the model, the above assertion is no longer obvious, and may not even be true in specific instances. So, we ask the question, "Given that the total processing power (number of processors $\times$ service rate per processor, called here the *number-power product*) is fixed at μ, what is the optimal number, N, of processors, that the system should start out with for a specific mission lifetime?" We extend an adaptation of this problem to demonstrate the use of our performance measures to real-time computers.

Two observations are in order here. Firstly, we tacitly assume that processors with *any* prescribed power are available. This is not true, although a wide variety of processors *is* available. Secondly, such a tradeoff depends for its resolution upon the cost

functions and hard deadlines introduced above. It is this second point we pursue here. Specifically, we set out to determine for an example control computer, the configurations that meet specifications of reliability, and the sensitivity of reliability and the mean cost, to changes in the number-power product under different operating conditions of hard deadline and mission lifetime. Implicit in all this will be the tradeoff between device redundancy and device speed.

3.1. System Description and Analysis

We use the multiprocessor system in Figure 1 to demonstrate the idea. Assume that there is a single job class that enters the system as a Poisson process with rate λ, and which requires an exponentially distributed amount of service with mean $1/\mu$. Then the system at any given time is an M/M/c queue (if the small dispatch time is ignored), where c is the number of processors functioning at that time. The distribution function for the response time for an M/M/c queue is well known [14]. It is given by:

$$F_{MM\,c}(t) = \frac{\{\lambda - c\mu + \mu W_c(0)\}(1 - e^{-\mu t}) + \mu\{1 - W_c(0)\}\{1 - e^{-(c\mu-\lambda)t}\}}{\lambda - (c-1)\mu} \tag{17}$$

where $W_c(0)$ is the probability that, when there are c processors functional, at least one functional processor is free. This is given by:

$$W_c(0) = 1 - \frac{c(\lambda/\mu)^c}{c!(c-\lambda/\mu)}\left[\sum_{n=0}^{c-1}\frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n + \frac{1}{c!}\left(\frac{\lambda}{\mu}\right)^c\left(\frac{c\mu}{c\mu-\lambda}\right)\right]^{-1} \tag{18}$$

This distribution function is not defined unless $c\mu > \lambda$.

Assume that the hard deadline has a probability distribution function F_d, and that the finite cost function for the task is denoted by g (as in Eq. (16)). Let the processors fail according to an exponential law with rate μ_p. Also, let n be the smallest integer for

which $n\mu > \lambda$ -- clearly, if there are fewer than n processors functioning, the utilization exceeds unity, and failure takes place with certainty.

The system fails if a hard deadline is violated, or if there are fewer than n processors functioning.[4] When the system is in a state $i > n$, the rate at which failure can happen is equal to the product of the task input rate and the probability that the response time of the system exceeds the hard deadline. If we assume that steady state is achieved between each state transition (i.e. between processor failures), then the probability distribution function of the response time at state i is always given by $F_{MM\,i}$. Such an assumption is valid, since the Mean Time Between Failures for components used in such systems typically ranges from 1,000 to 10,000 hours. The probability of dynamic failure can therefore be computed using the Markov model in Figure 2. Denoting the probability of being in state i by π_i, the quantity $\lambda[1 - F_{MM\,j}(t)]$ by $\alpha(j,t)$, the failure state by *fail* and the number of processors at start-up by N, the following balance equations can be written

$$\dot{\pi}_N(t) = -[N\mu_p + \int_0^\infty \alpha(N,\xi)\, dF_d(\xi)\,]\pi_N(t), \qquad \pi_N(0)=1 \tag{19a}$$

$$\dot{\pi}_i(t) = -[\, i\mu_p + \int_0^\infty \alpha(i,\xi)\, dF_d(\xi)\,]\pi_i(t) + (i+1)\mu_p\pi_{i+1}(t), \quad \pi_i(0)=0, \text{ for } n \le i < c \tag{19b}$$

$$\dot{\pi}_{fail}(t) = \sum_{i=n}^{N} \{\int_0^\infty \alpha(i,\xi)\, dF_d(\xi)\}\pi_i(t) + n\mu_p\pi_n(t), \quad \pi_{fail}(0)=0. \tag{19c}$$

Clearly, $p_{dyn}(t) = \pi_{fail}(t)$ so that a solution of the above equations yields the probability of dynamic failure. Implicit in these calculations is the assumption that the hard deadline is very much smaller than the mission lifetime or the sojourn time in the various

[4] We do not consider here the failure of the interconnection net, or of the dispatcher. Taking account of these is easy, but would obscure the analysis somewhat.

states. This is invariably the case in practice.

To compute the mean finite cost over the mission lifetime, we first have to evaluate the distribution function of the response time, conditioned on the event that no hard deadline is violated. This distribution function for a c-processor system, denoted by $F^{finite}_{MM\,c}$, is given by:

$$F^{finite}_{MM\,c}(\xi) = \int_0^\infty \frac{F_{MM\,c}(\xi)}{F_{MM\,c}(\tau)}\, dF_d(\tau) \tag{20}$$

where the function is only defined for arguments less than the associated hard deadline. Then, the mean cost is defined by:

$$M = \sum_{c=n}^{N} \int_0^\infty \int_0^\infty \int_0^\xi \lambda\, \pi_c(t)\, g(\tau)\, dF^{finite}_{MM\,c}(\tau)\, dF_d(\xi)\, dL(t) \tag{21}$$

The above expressions are used in the following section to obtain values for the probability of dynamic failure, and the mean cost as a function of the mission lifetime.

3.2. Numerical Results and Discussion

In what follows, it is assumed that the job arrival rate, λ=100, and that the processor failure rate is μ_p=10^{-4}. All time units are in hours.

Figure 3 is the probability of dynamic failure when the task is non-critical, i.e. the deadline is at infinity. In such cases, the probability of dynamic failure reduces to being the probability of *static failure,* namely the probability of failure of hardware components to the point when the system utilization exceeds 100 %. This is because catastrophic failure does not occur in this case until the system is overloaded. This explains the monotonic nature of the plot.

Figure 4 shows the probability of dynamic failure when the task is critical with the deadlines for the respective curves noted in the figure. The number-power product is

2200. The curves that result for the failure plot form an inverted bell. The portions of the failure curve where the slope is positive can be explained as follows. When the number of processors increases, the response time distribution is skewed to the right. This in turn increases the probability of failing to meet the hard deadline to a greater extent than the static failure probability is reduced by the addition of further redundancy. The positive slope is the result of this tendency. When the hard deadline is smaller, the premium on speed is increased, and as a result, the trough of the curves moves to the left.

In the region corresponding to the fewest processors, there tends to be a tradeoff between dynamic failure probability and the number of processors, leading to a negative slope for the failure curves. When there are few processors, the fault-tolerance is less and the probability of static failure is therefore greater. Since the total processing power of the processor set is fixed, the few processors each have greater power, and the mean waiting time is low. This accounts for the very small nature of the non-static component of the failure probability. As the speeds of the individual processors are decreased, but their numbers increased commensurately, the static failure probability drops, but the probability of missing the hard deadline increases. In the area of the curve where the slope of the probability of failure is negative, the benefits accrued from adding redundancy outweigh the negative impact of the lowered individual speeds that result; elsewhere the reverse is the case. Notice that the curve corresponding to a deadline of 0.01 has no region of negative slope. This means that 2200 is a number-power product that is too small with respect to that hard deadline.

In Figure 5, the dependence of the probability of dynamic failure on the number-power product for a mission lifetime of 10 hours is considered. Each point on the curves

represents the configuration yielding the lowest possible failure probability for the number-power product represented. The label of each point on this plot is the number of processors in the configuration for which this lowest failure probability is achieved. As the product increases, the optimal configuration tends to contain more processors: this also is due to the lowering of the non-static component of the dynamic failure probability when the product is increased.

Naturally, the curves are monotonically non-increasing. They serve to show the marginal gain in maximum achievable reliability that is to be had on increasing the number-power product at each point for the class of systems under consideration. Notice the "elbows" in the plot. These occur when the minimum failure probability configuration changes, and are the result of a tradeoff between the static and non-static components of the failure probability. The p_{dyn} drops exponentially with an increase in the product as long as the static component is a small fraction of p_{dyn}. When the non-static component drops to sufficiently below the static component value, the optimal configuration changes, and the static component once again becomes negligible compared to the non-static component. This race continues indefinitely and is portrayed in Figure 6. The discrete nature of the processors causes the elbows: if the number of processors were a continuous quantity, they would not appear.

The probability of dynamic failure is used as a pass-fail test for control computers. Plots such as Figure 5 can be used in this connection. As an example, let the mission lifetime be 10 hours, and the specified probability of dynamic failure equal to 10^{-7} over that period. Let the system parameters be those of the model in this section. Then, corresponding to each of the four deadlines considered, we can obtain graphically from Figure 5, the minimum number-power product that is required to satisfy the p_{dyn} specifi-

cations. These products are listed in Table 1. Any system that has a smaller product than this must be rejected, no matter what its other credentials may be.

When this stage of the evaluation is complete, one has a set of acceptable configurations. Only after this point does the mean cost come into consideration. The mean costs associated with each of the points in Figure 5 is graphed in Figure 7, where the finite cost function, g, has been taken as equal to the response time for demonstrative purpose. The curves take the form of a sawtooth wave, with each upward transition occurring when the optimal configuration increases by one. Clearly, the greater the power of each processor, the smaller is the mean cost.

In Figure 8 (A, B, and C), we show the effects on p_{dyn} of changing mission lifetime for various values of the hard deadline, t_d. In the light of the preceding discussion, these plots should be largely self-explanatory. It is worth pointing out, however, that as the lifetime increases, the optimal configuration contains a larger number of processors. The trough (around the optimal point) becomes shallower as one increases the mission lifetime, until finally, it disappears to be replaced by a shallow trough one unit to the right. As the lifetime increases still further, the new trough deepens, then begins to become shallow. Whether or not the cycle continues depends upon the hard deadline: it will continue so long as the number-power product is sufficiently large to cope with the hard deadline at the lifetimes used; the plot will rise monotonically to a failure probability of one if this is not the case (cf. Figure 3).

In Figure 9 (A, B, and C), we show the associated Mean Costs per unit time of operation using the same cost function as was used in Figure 7. In all three curves, we may note the anomaly mentioned in Section 2: as the lifetime increases, and as the number of processors increases, there is a region over which the mean costs per hour

actually drop. It is most pronounced in Figure 9A, where the probability of dynamic failure is close to unity under almost all configurations. This anomaly, of course, is due to the fact that the mean costs are computed on a response-time distribution that is conditioned on the system's not failing. Thus, on comparing Figures 9A, 9B, and 9C, we see that the system operating in the longest hard deadline, and therefore having greater reliability exhibits a *higher* mean cost per hour in some configurations than its identical counterparts that operate under more difficult conditions. If this causes undue irritation, the anomaly can be made to vanish by redefining the finite cost function for a task i to be:

$$f_i(t) = \begin{cases} g_i(t) & \text{if } t \leq t_d \\ g_i(t_d) & \text{if } t > t_d \end{cases} \tag{22}$$

introducing the following functions:

$$\alpha_i(t) = \sum_{j=1}^{q_i(t)} g_i(\Xi(V_{ij}, \tau_{ij}, n_{ij}, t)) \tag{23}$$

$$\beta(t) = \sum_{j=1}^{r} \Gamma_j(t) \tag{24}$$

and defining the mean and variance costs to be:

$$\textit{Mean Approximate Cost}\,(MAC) \equiv \int_0^\infty E\{\beta(t)\}\, dL(t) \tag{25}$$

$$\textit{Variance Approximate Cost}\,(VAC) \equiv \int_0^\infty Var\{\beta(t)\}\, dL(t) \tag{26}$$

It is easy to see that $MAC > \textit{Mean Cost}$ always, and that the approximate costs approach the accurate costs when the probability of dynamic failure is small. Indeed, the anomaly does not appear until the probability of dynamic failure is significant. Since the applications under consideration are all critical processes, p_{dyn} is always small for the

accepted configurations, and the configurations that exhibit this anomaly will be rejected by the p_{dyn} pass-fail test, and their mean costs need never be computed.

3.3. Extension

The number-power tradeoff can easily be extended to make it very useful in the process of design. The reader will have noticed that in the cost functions with which we measure the goodness of controller performance, no account is taken of controller hardware cost. All that the cost functions express is the control overhead incurred in actually *running* the process. Indeed, we may regard the mean costs as average *operating* overheads. It is not easy directly to incorporate the hardware cost into the cost functions themselves. Instead, one may consider the set of hardware configurations available for a particular hardware cost outlay. Then, *constant-cost* plots can be drawn, showing the range of performance (in terms of probability of dynamic failure and average operating costs) that is available for any particular hardware cost outlay. From similar curves, one may arrive at the minimum finite average operating cost associated with a particular hardware cost given that specifications for the probability of dynamic failure are met.

This approach can easily be illustrated with the number-power tradeoff considered here. When the hardware cost of a processor is proportional to its processing speed, the curves in Figure 4 become dynamic failure curves for a particular hardware cost. Curves such as Figure 8 can be used to identify the configurations that meet requirements for the probability of dynamic failure for given mission lifetimes, and the sensitivity of p_{dyn} to changes in mission lifetime.

Also, one can study any other tradeoffs that may exist between the hardware cost or the number-power product and the minimum mean cost per lifetime associated with

such a cost or product.

The computer studied here is simple; however, it can be extended in some useful directions relatively easily. It is easy to take care of the case when the hardware cost or the finite cost function is a more complicated function of the processing speed. More complicated multiprocessors require a more involved analysis, but the basic ideas should now be clear.

4. CONCLUSION

In this report, we have proposed performance measures for computers used in the control of critical processes. The principal differences between our measures and those of others are summarized in Table 2.

Studying the behavior of the controlled system as a function of the computer response time provides a means for the effective design of computer controllers in the context of controlled processes. As we saw in the examples in Section 2, this includes such things as control policy. This means that while the cost function is defined explicitly in terms of the controller response time, all facets of the controlled process are implicitly included in the calculations.

Due to the objectivity of the cost functions, they can be used with some confidence for the design of real-time control computers (architecture and operating system design) as optimization criteria. The probability of dynamic failure is to be used as a pass-fail criterion with comparison of rival systems on the basis of the mean cost limited to systems exhibiting an acceptably low probability of dynamic failure. The inclusion of hardware or life-cycle costs into the analysis is also possible, as indicated in Section 3.3.

In addition to the applications treated in Section 3, the cost functions are useful as criterion functions in the following areas:

(1) Optimal placement of checkpoints for backward error recovery.

(2) Optimal task allocation and reallocation strategies.

(3) Optimal control of queues at shared resources.

(4) Optimal routing policies at interconnection networks.

(5) Measuring sensitivity of reliability and operating overhead on the redundancy and bandwidth of the interconnection links.

(6) Optimal event-handling and time scheduling.

The design procedures used at present for control computers are ad-hoc, principally because of the lack of adequately objective means for the characterization of controller performance. The expression, through a scalar metric, of the performance of the controller in the context of the process it is controlling, is important in making the controller design and evaluation process systematic. All facets of the controller-controlled process relationship are taken into account in the performance measures here presented: while the measures are explicitly functions of the response time of the controller, they are implicitly functions of the characteristics of the controlled process. It is this fusion of controller and controlled process characteristics that is novel, and that distinguishes the work presented in this report from those of others. The chief utility of these measures is also derived from this accounting of the synergistic coupling between controller and controlled process.

ACKNOWLEDGMENT

The authors are grateful to Yann-Hang Lee, Ricky W. Bulter, Milton Holt, and the

anonymous referees for their constructive comments on the material presented in this report.

References

[1] M. D. Beaudry, "Performance-Related Reliability Measures for Computing Sustems," *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 540-547, June 1978.

[2] H. Mine and K. Hatayama, "Performance Evaluation of a Fault-Tolerant Computing System," *Proc. 1979 Int'l Symp. Fault-Tolerant Comput.*, Madison, WI, pp. 59-62, June 1979.

[3] R. Huslende, "A Combined Evaluation of Performance and Reliability for Degradable Systems," *Proc. SIGMETRICS Conf. on Meas. and Model. of Comp. Sys.*, pp. 157 - 164, September 1981.

[4] T. C. K. Chou and J. A. Abraham, "Performance/Availability Model of Shared Resource Multiprocessors," *IEEE Trans. Reliability*, Vol. R-29, No. 1, pp. 70-76, April 1980.

[5] X. Castillo and D. P. Sieworek, "A Performance Reliability Model for Computing Systems," *Proc. 1980 Int'l Symp. Fault-Tolerant Comput.*, pp. 187-192, October 1980.

[6] X. Castillo and D. P. Sieworek, "Workload, Performance, and Reliability of Digital Computing Systems," *Proc. 1981 Int'l Symp. Fault-Tolerant Comput.*, pp. 84-89, June 1981.

[7] S. Osaki and T. Nishio, *Reliability Evaluation of Some Fault-Tolernat Computer Architectures,*" Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1980.

[8] J. F. Meyer, "On Evaluating the Performability of Degrading Computer Systems," *IEEE Trans. Comput.*, Vol. C-29, No. 6, pp. 501-509, June 1980.

[9] J. F. Meyer, "Performability Modelling of Distributed Real-Time Systems," *Int'l Workshop on Applied Mathematics and Performance/Reliability Models of Computer/Communications Systems*, University of Pisa, Pisa, Italy, pp. 345-356, September 1983.

[10] G. K. Manacher, "Production and Stabilization of Real-Time Task Schedules," *J. ACM*, Vol. 14, No. 3, pp. 439-465, July 1967.

[11] K. G. Shin, C. M. Krishna, and Y.-H. Lee, "The Application to the Aircraft Landing Problem of a Unified Method for Characterizing Real-Time Systems," *Proc. IEEE Real-Time Systems Symposium*, Arlington, VA, December 1983.

[12] P. J. B. King and I. Mitrani, "The Effect of Breakdown on the Performance of Multiprocessor Systems," in F. J. Kylstra, ed., *Performance '81,* pp. 201-211, North-Holland, Amsterdam, 1981.

[13] D. E. Kirk, *Optimal Control Theory,* Prentice Hall, Englewood Cliffs, NJ, 1970.

[14] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory,* John Wiley, New York, 1974.

APPENDIX: EXPRESSIONS FOR FINITE COST FUNCTION

Finite cost functions for Example 2 in Section 2.4 can be expressed by an **if-then-else** construct as follows:

if $x_{1i}x_{2i}>0$ **then**

if $|x_{2i}|>k$ **then**

$$g(\mathbf{x}_i,\xi) = \frac{1}{2}[t_1(\mathbf{x}_i,k,\xi) + t_1(\mathbf{x}_i,-k,\xi)]$$

else

$$g(\mathbf{x}_i,\xi) = \frac{1}{2}[t_1(\mathbf{x}_i, \mathrm{sgn}(x_{2i})k, \xi) + t_2(\mathbf{x}_i, -\mathrm{sgn}(x_{2i})k, \xi)]$$

else

if $|x_{2i}|>k$ **then**

$$g(\mathbf{x}_i,\xi) = \frac{1}{2}[t_2(\mathbf{x}_i,k,\xi) + t_2(\mathbf{x}_i,-k,\xi)]$$

else

$$g(\mathbf{x}_i,\xi) = \frac{1}{2}[t_1(\mathbf{x}_i, -\mathrm{sgn}(x_{2i})k, \xi) + t_2(\mathbf{x}_i, \mathrm{sgn}(x_{2i})k, \xi)]$$

end if;

where $a = H/m$, $y(x_{2i},k)=x_{2i}+k$, $\mathbf{x}_i=(x_{1i},x_{2i})^T$,

$$r(\mathbf{x}_i,k,\xi) = \frac{2a|x_{1i}| - y^2(x_{2i},k) - 2a|y(x_{2i},k)|\xi}{4ay(x_{2i},k)}$$

$$t_1(\mathbf{x}_i,k,\xi) = \xi + \frac{|y(x_{2i},k)(1+\sqrt{2})|}{a}$$

$$t_2(x_i,k,\xi) = \begin{cases} \xi+\tau(x_i,k,\xi)+\dfrac{|x_{1i}|-|y(x_{2i},k)|(\xi+\tau(x_i,k,\xi))}{a}-\dfrac{\tau^2(x_i,k,\xi)}{2} & \text{if } \xi \le \dfrac{2a|x_{1i}|-y^2(x_{2i},k)}{2a|y(x_{2i},k)|} \\ \xi+\dfrac{y^2(x_{2i},k)}{2a}+2\sqrt{\dfrac{y^2(x_{2i},k)}{a^2}-\dfrac{|x_{1i}|-|y(x_{2i},k)|\xi}{a}} & \text{otherwise} \end{cases}$$

Hard Deadline	Number-Power Product
0.010	7165
0.025	2126
0.050	1496
0.075	1180

Required $p_{dyn} = 10^{-7}$
Mission Lifetime = 10 hours

Table 1. Minimum Number-Power Product for Various Hard Deadlines.

Other Measures	Our Measures
Wide applicability to almost all applications of fault-tolerant, gracefully degrading systems.	Limited applicability. Aims specifically at real-time, especially at control, applications.
Measures express performance in rather gross terms.	Measures express performance in rather exact terms.
Performance measured with respect to the computer system alone.	Performance measures specifically designed to reflect the overhead of the computer on the application.

Table 2. Comparison of Traditional and New Methods of Characterizing Performance

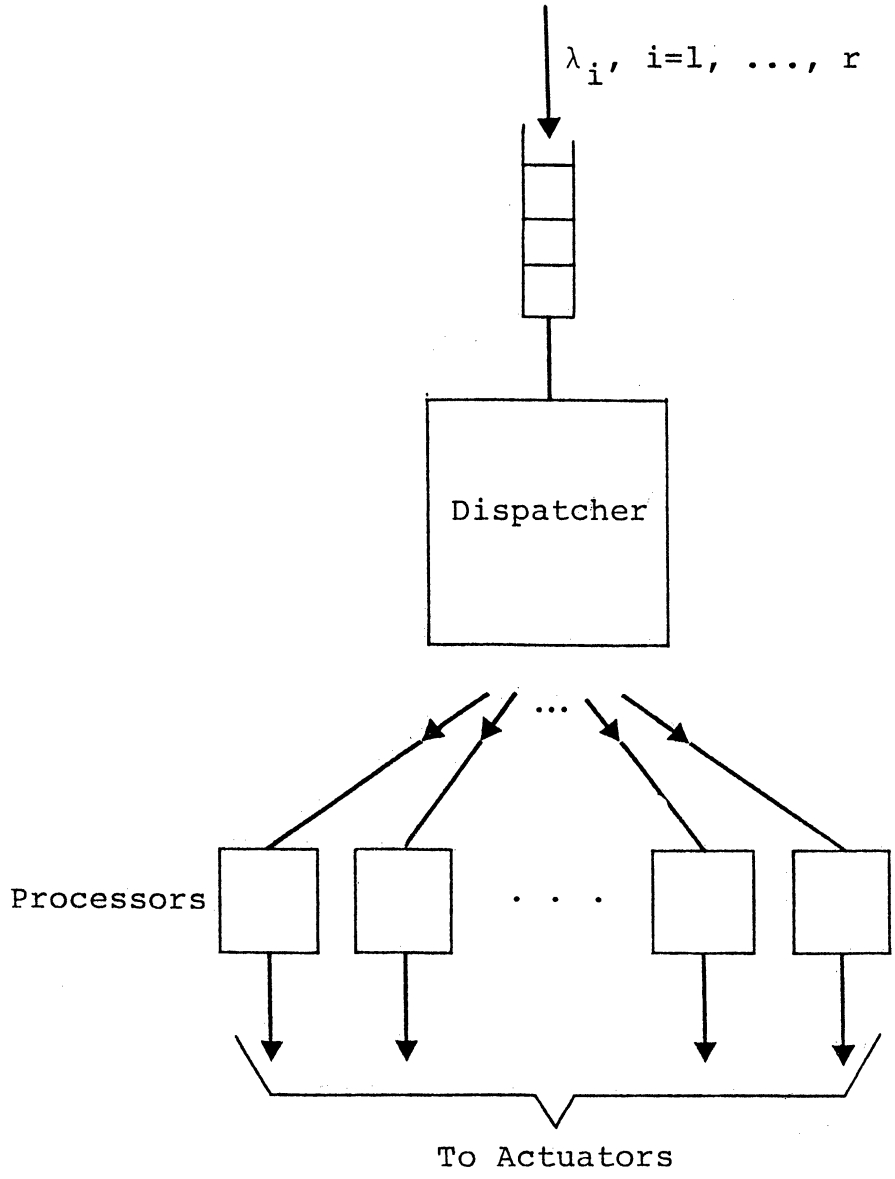


Figure 1. A real-time multiprocessor.

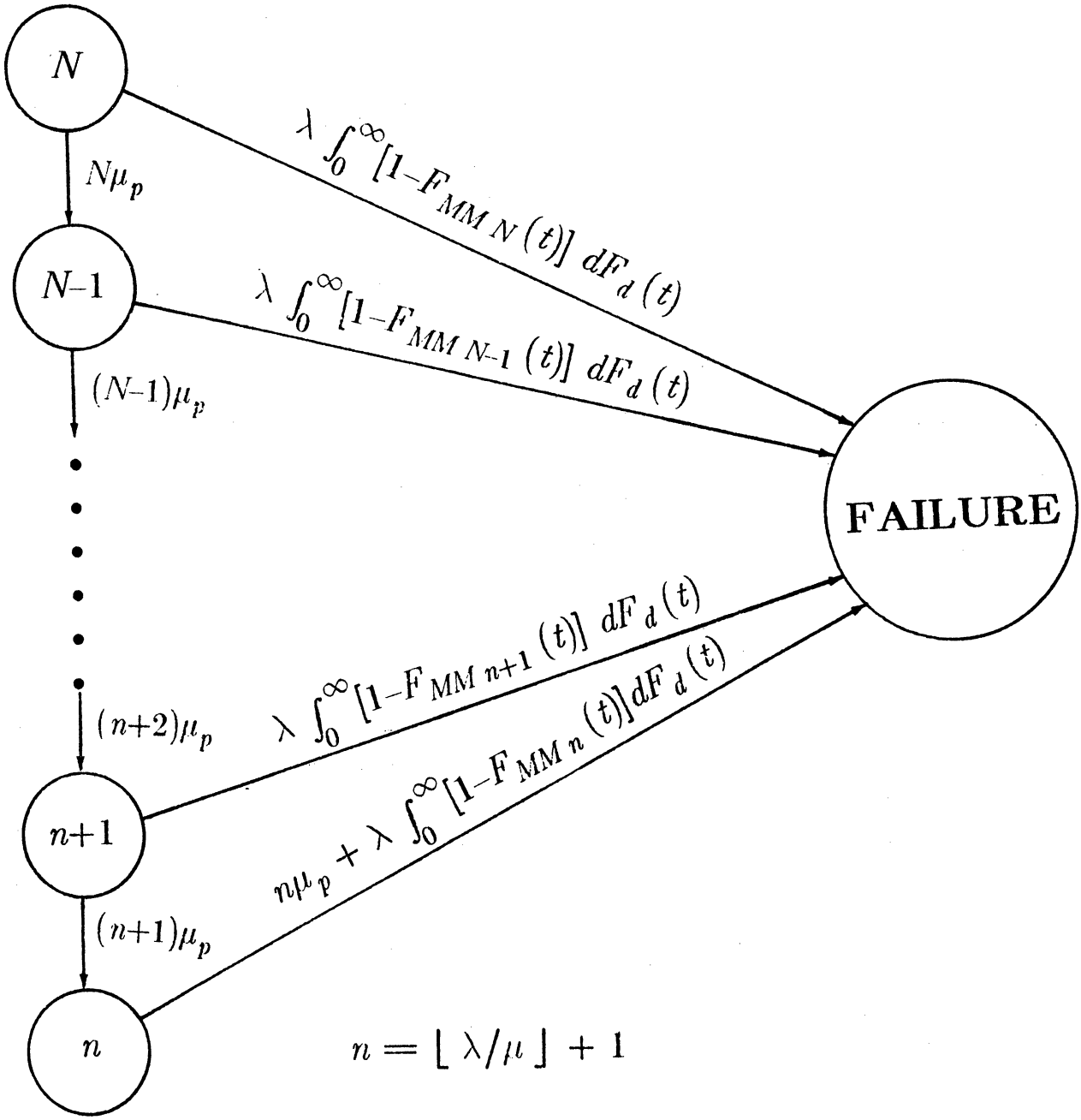


Figure 2. Markov model for number-power tradeoff.

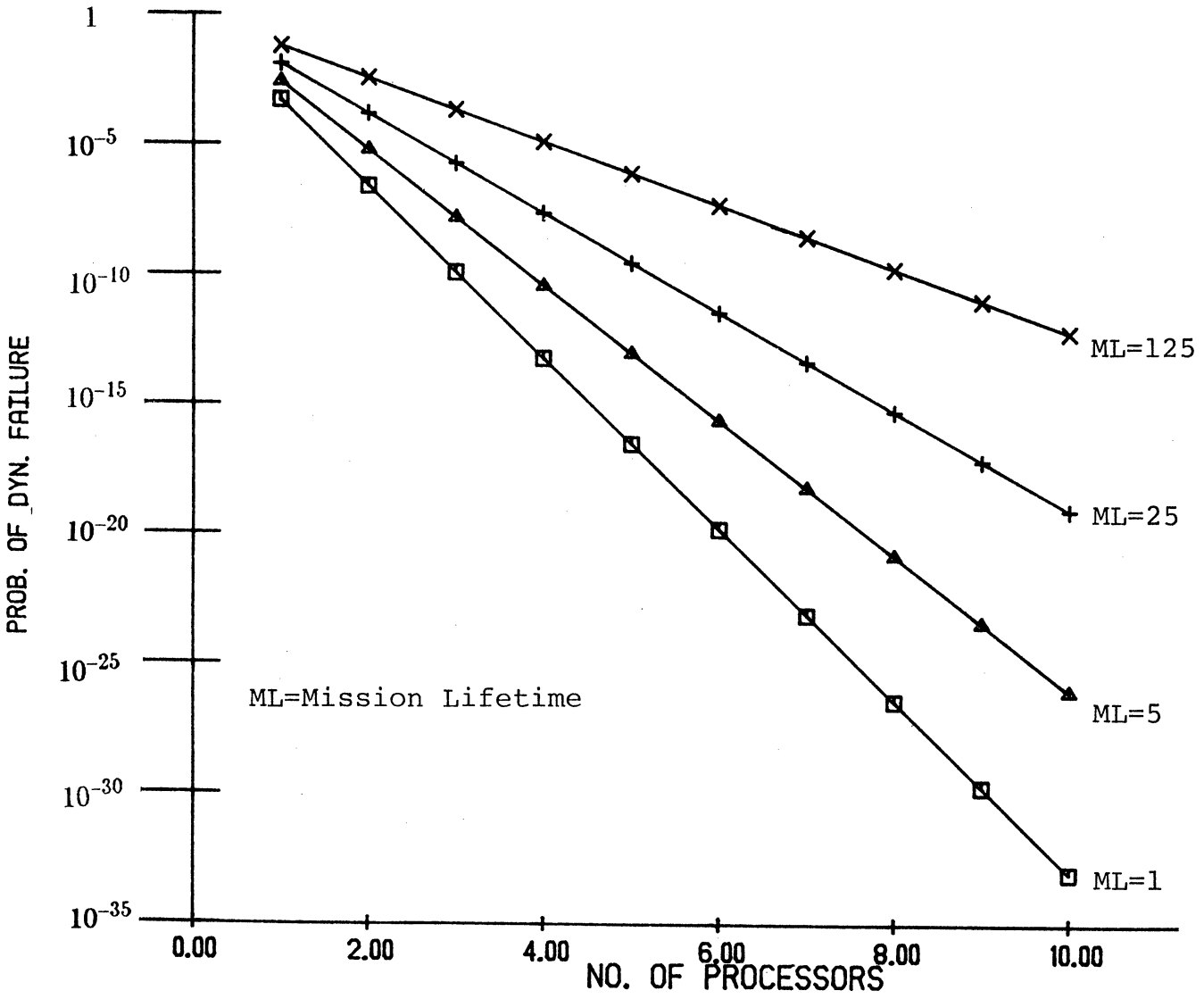


Figure 3. Dependence of probability of dynamic failure on processor number when tasks are non-critical.

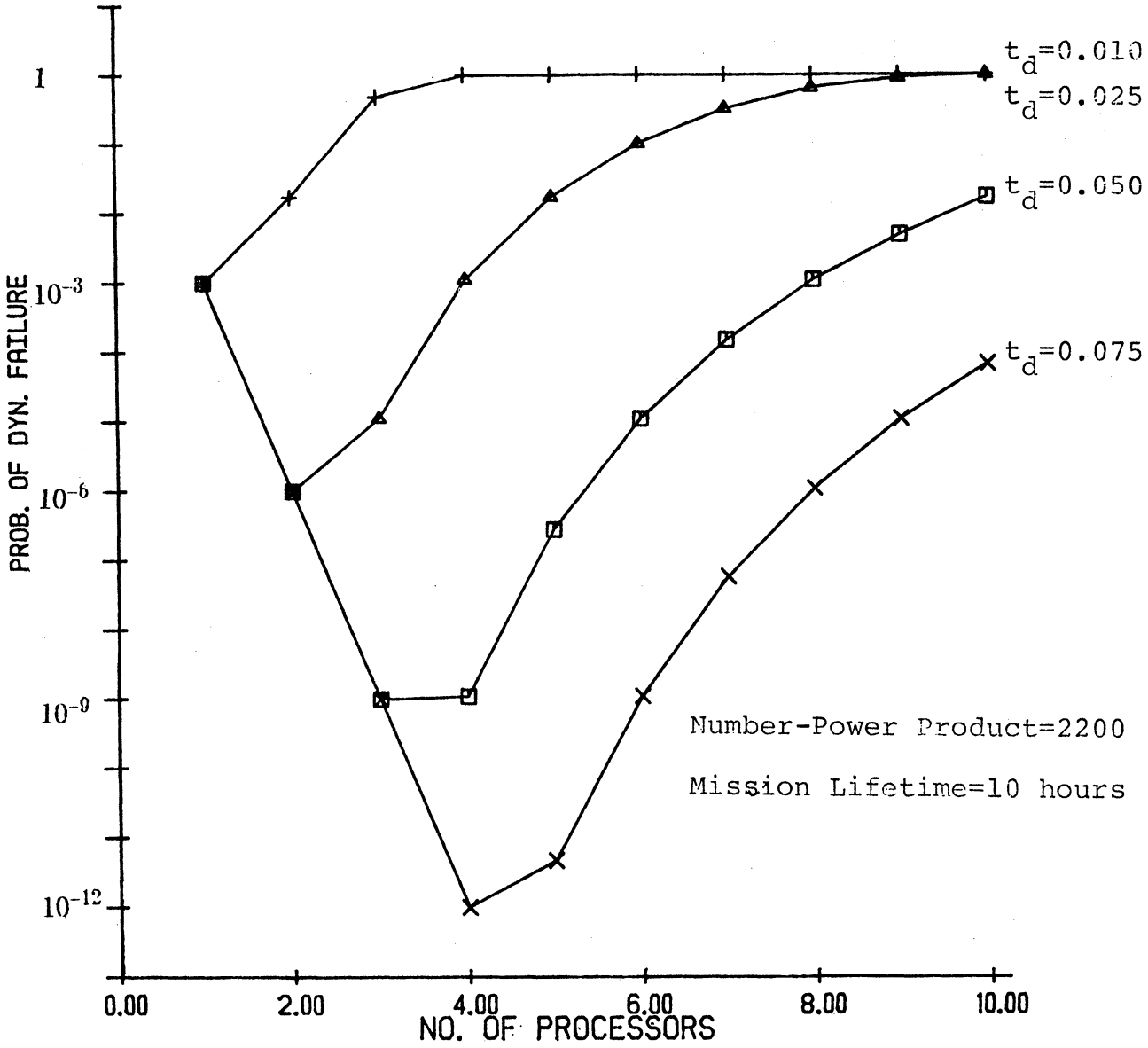


Figure 4. Dependence of probability of dynamic failure on hard deadlines and number of processors.

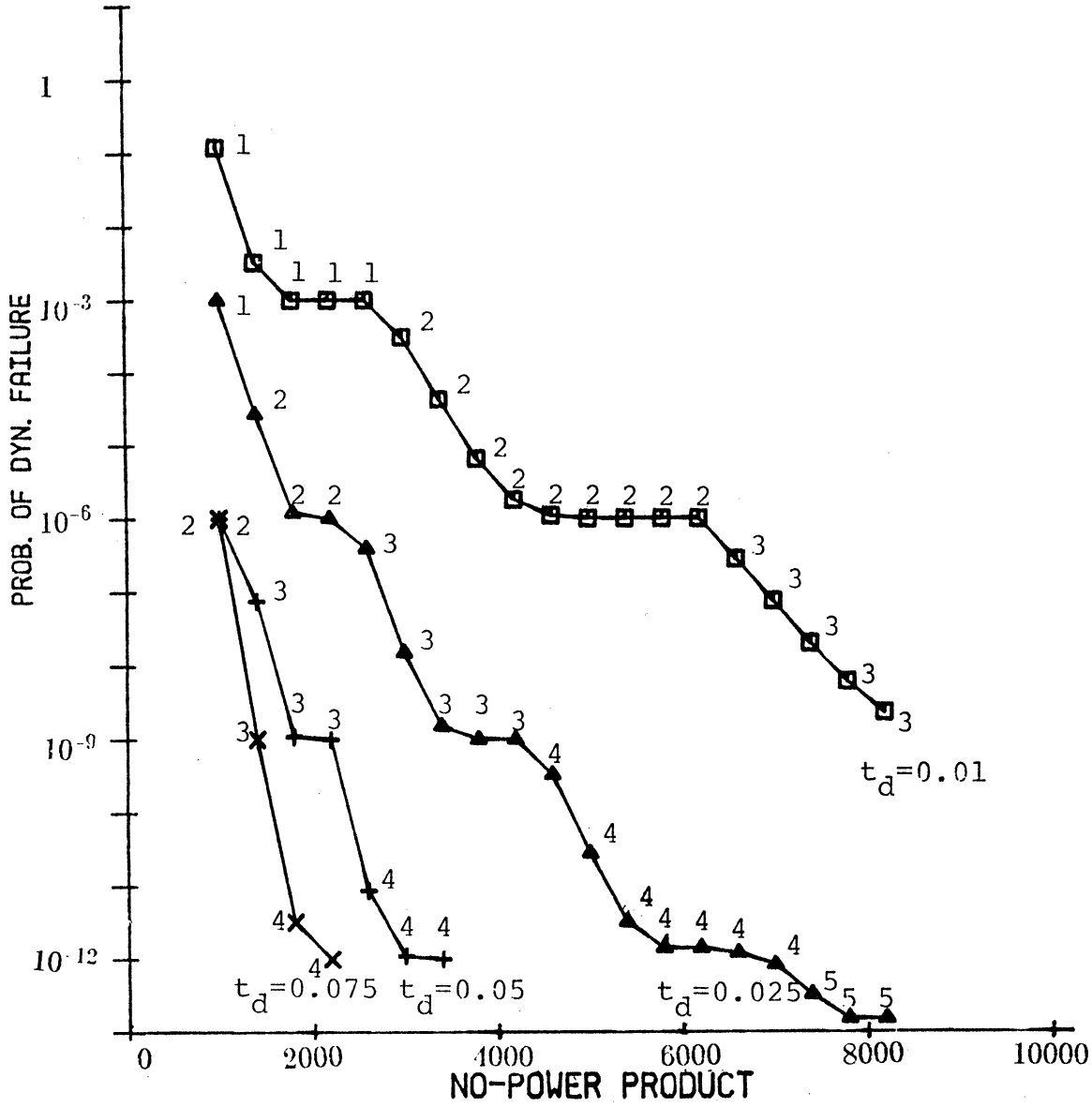


Figure 5. Minimum achievable probability of dynamic failure.

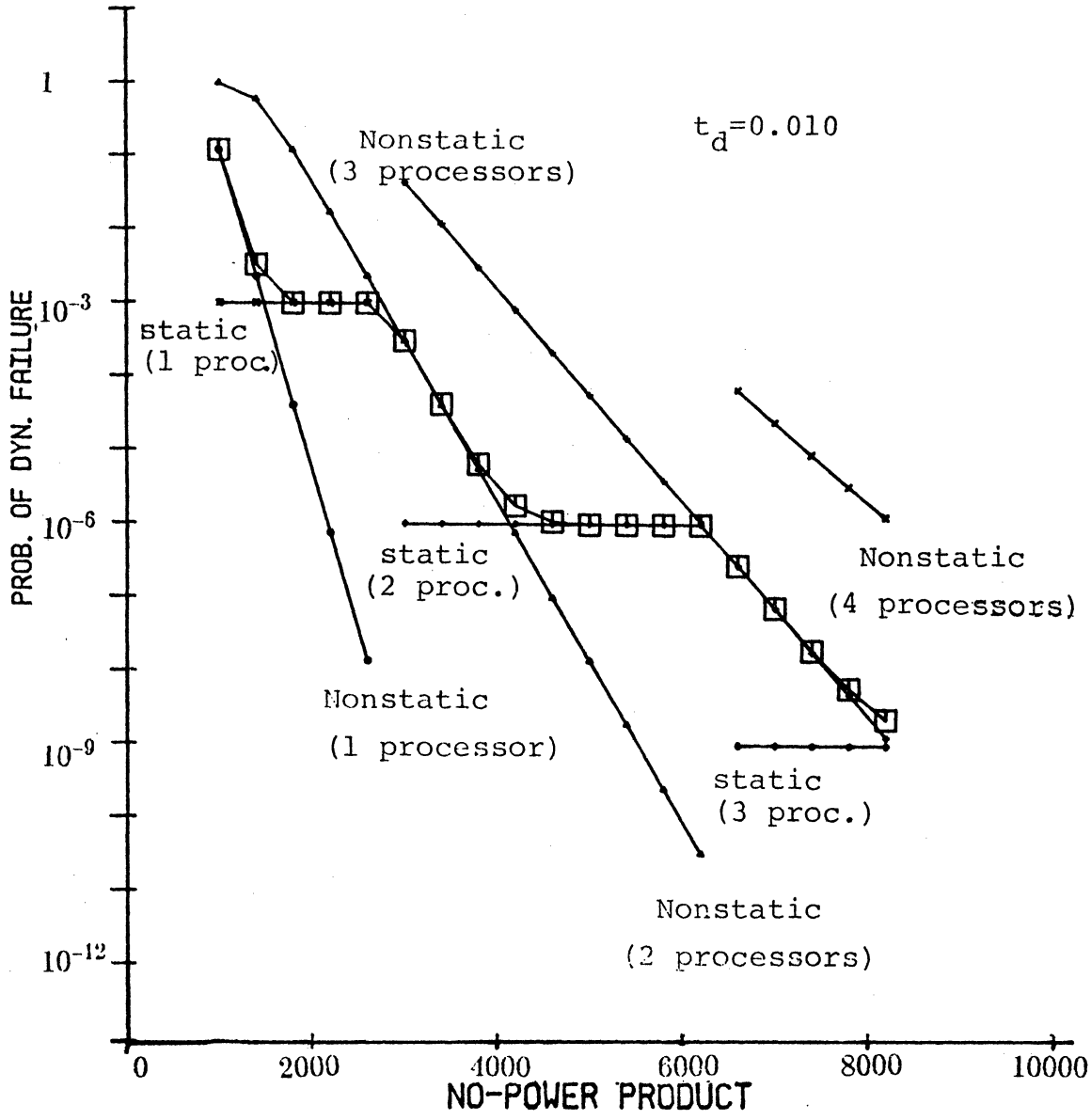


Figure 6. Race between static and non-static components of probability of dynamic failure.

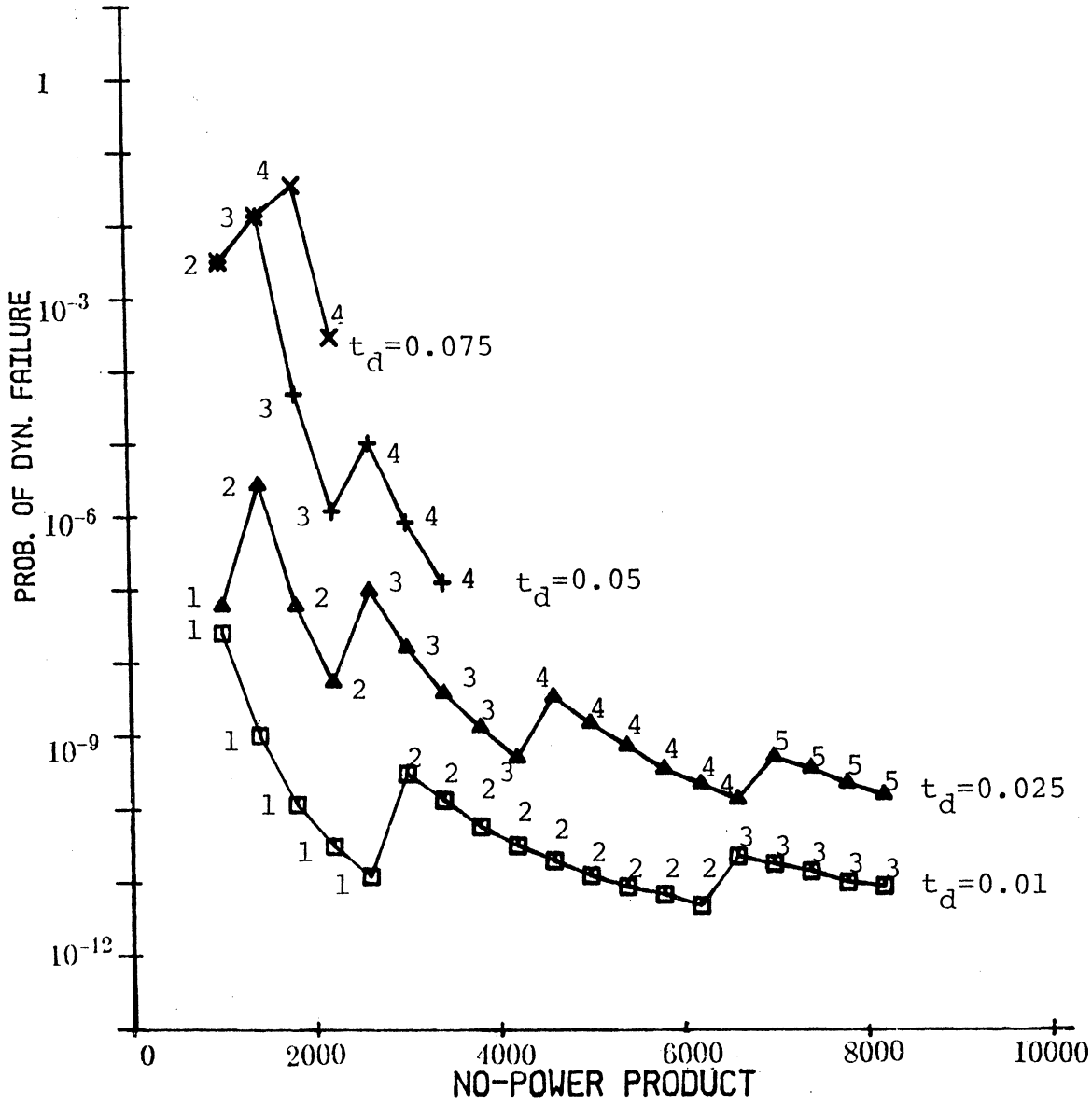


Figure 7. Mean cost for configurations of Figure 5.

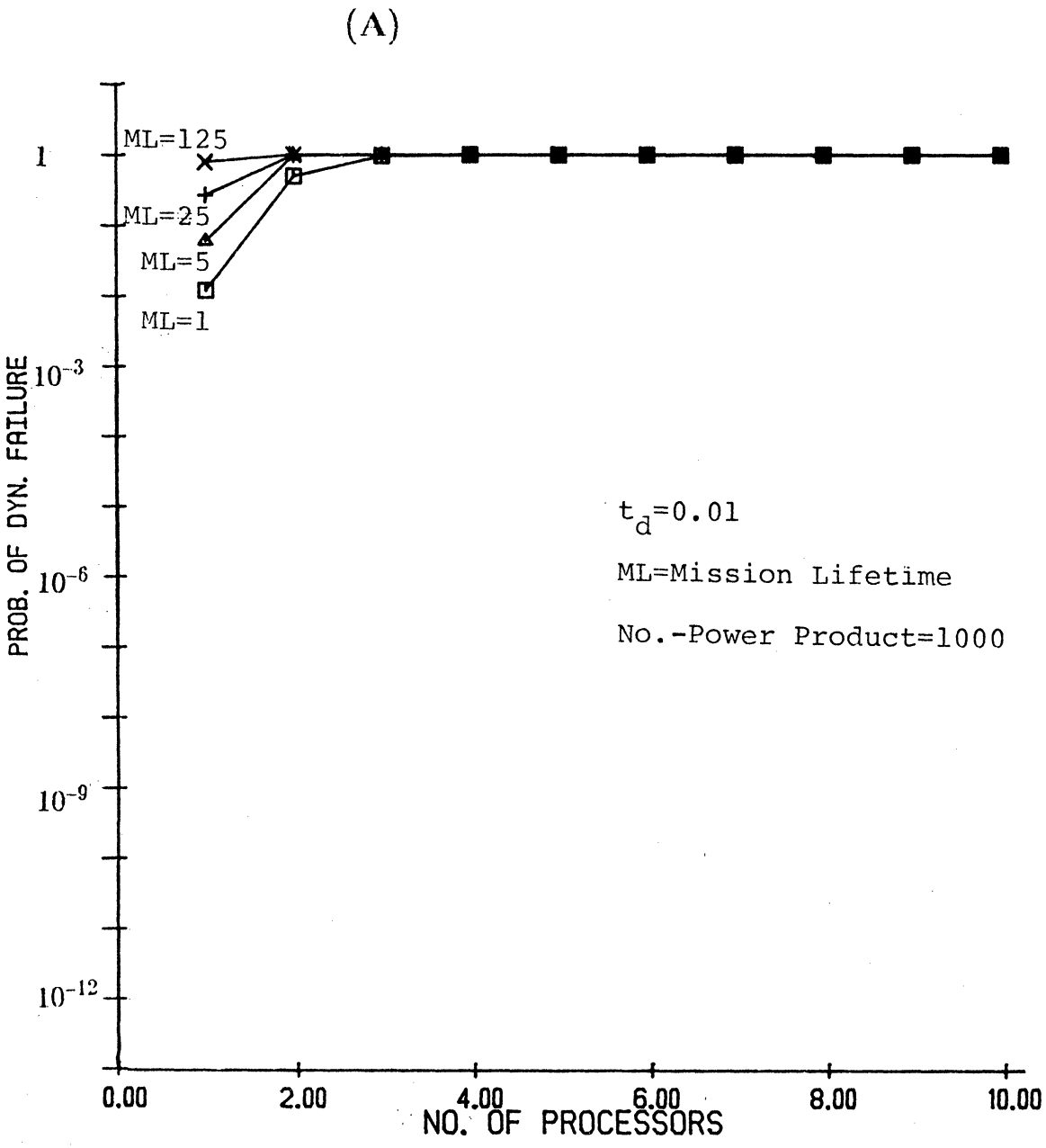
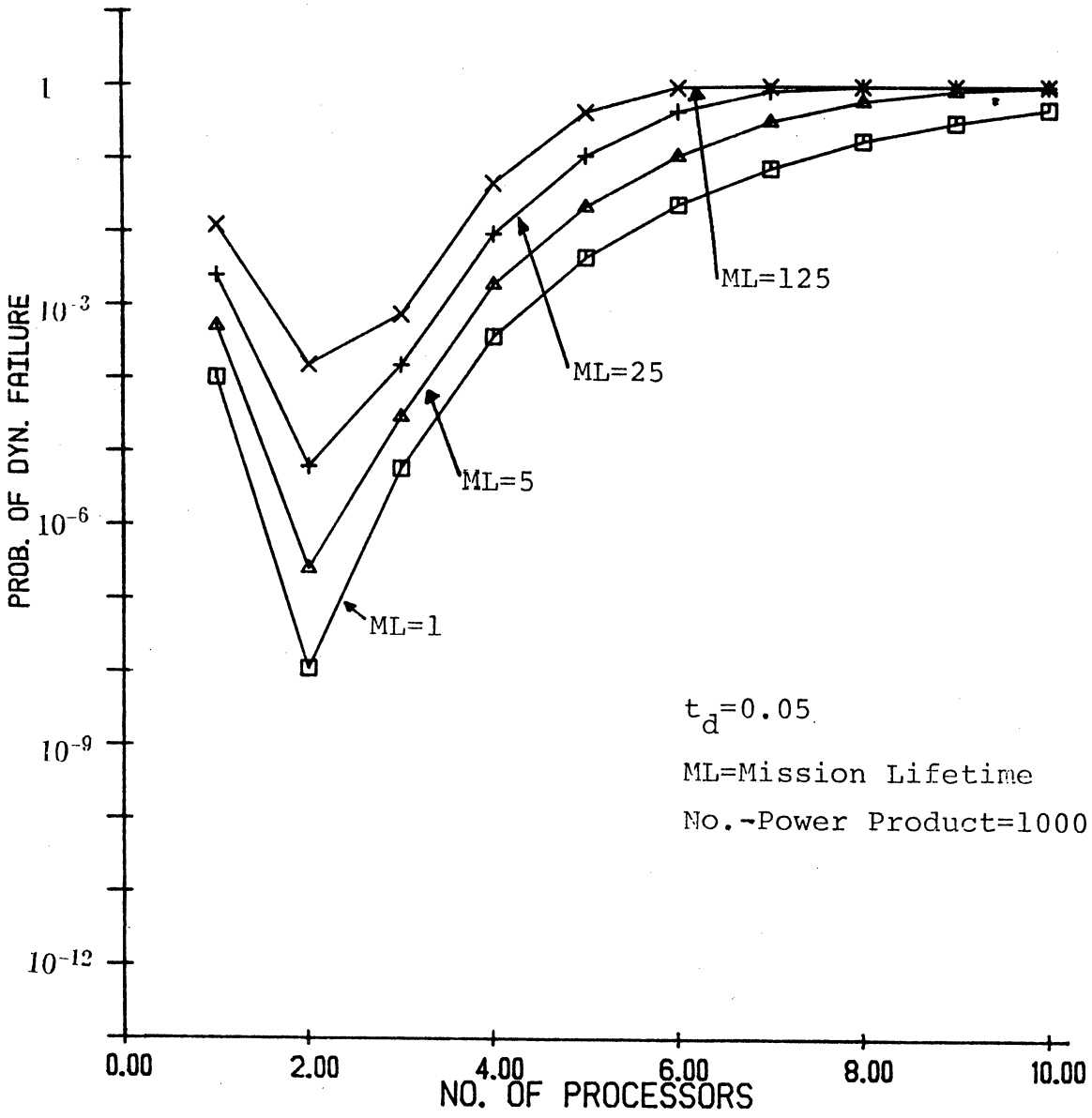
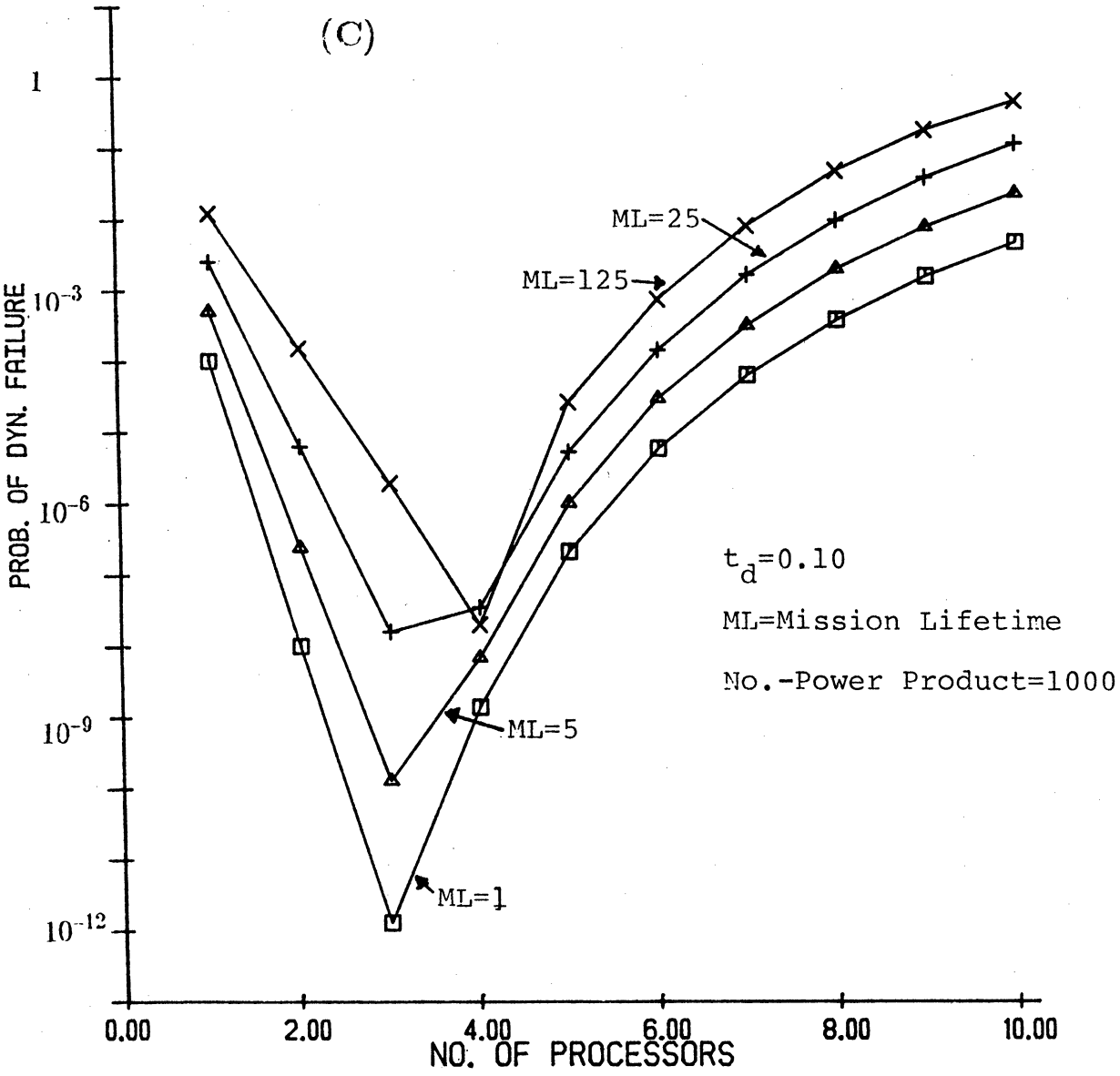


Figure 8. Probability of dynamic failure for a constant number-power product.

(B)


1
10^{-3}
10^{-6}
10^{-9}
10^{-12}
PROB. OF DYN. FAILURE
ML=125
ML=25
ML=5
ML=1
t_d=0.05
ML=Mission Lifetime
No.-Power Product=1000
0.00
2.00
4.00
6.00
8.00
10.00
NO. OF PROCESSORS


(C)
1
10^{-3}
10^{-6}
10^{-9}
10^{-12}
PROB. OF DYN. FAILURE
ML=25
ML=125
ML=5
ML=1
t_d=0.10
ML=Mission Lifetime
No.-Power Product=1000
0.00
2.00
4.00
6.00
8.00
10.00
NO. OF PROCESSORS

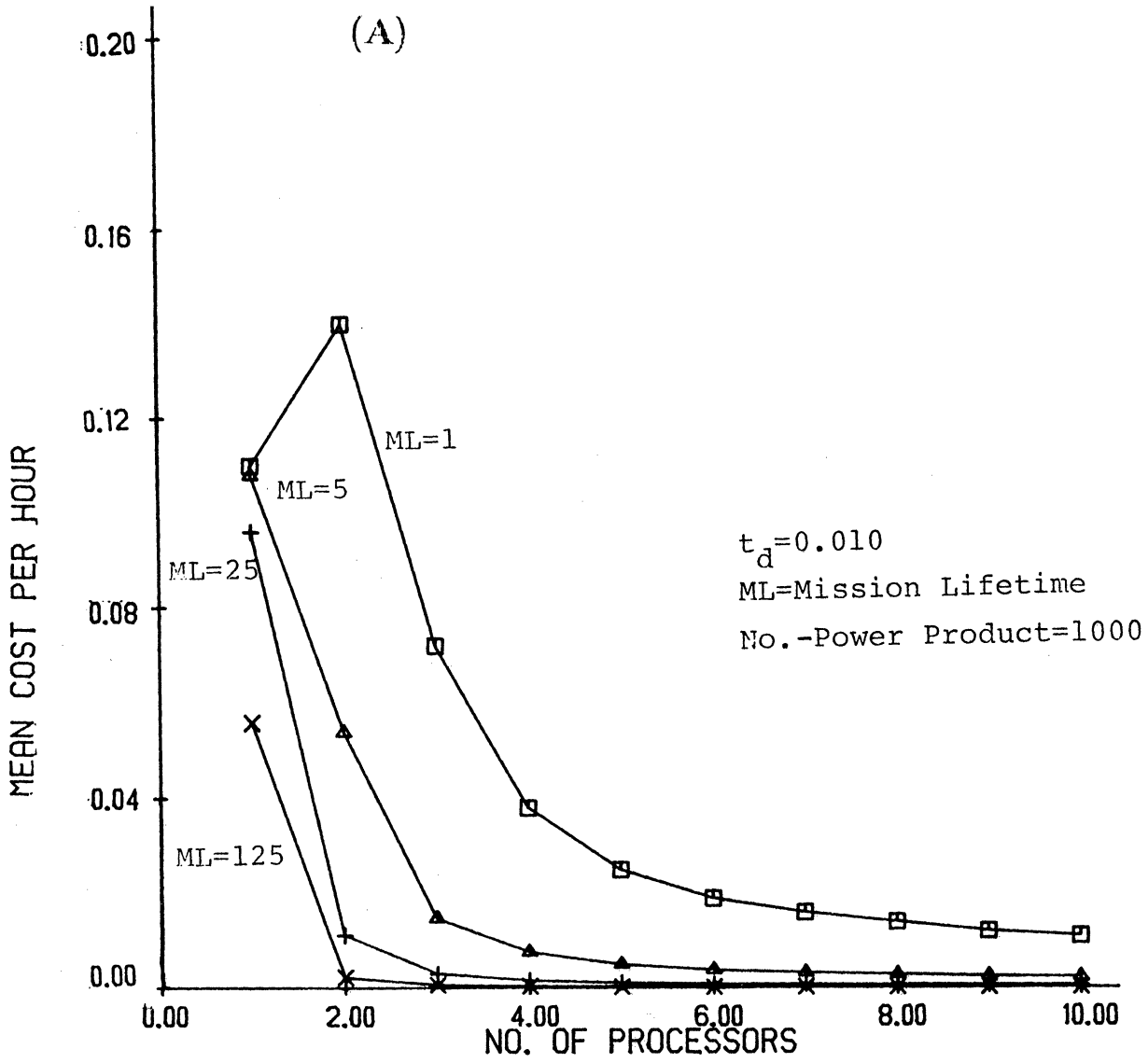
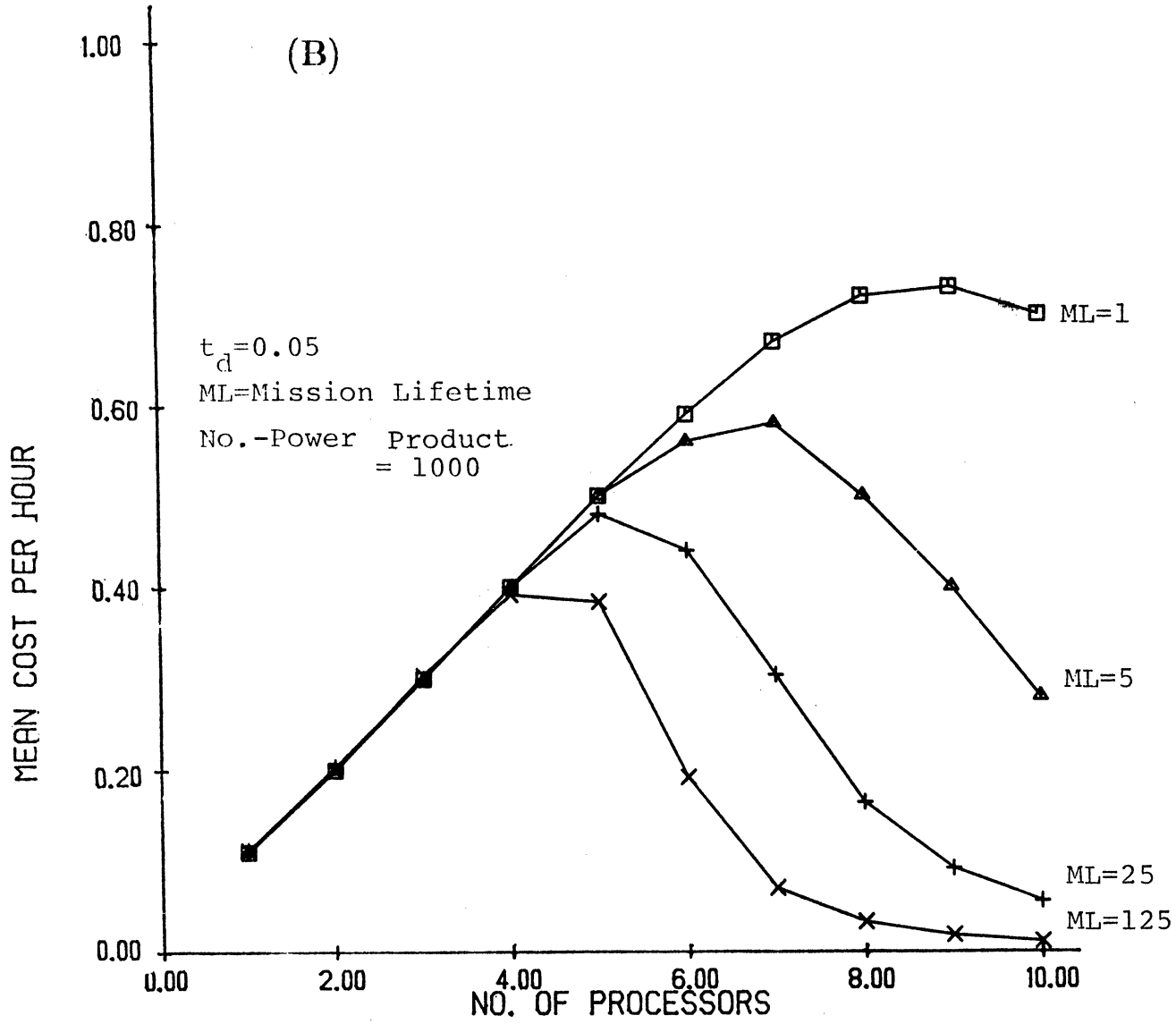


Figure 0. Mean cost for a constant number-power product.


(B)
1.00
0.80
0.60
0.40
0.20
0.00
MEAN COST PER HOUR
t_d=0.05
ML=Mission Lifetime
No.-Power Product
= 1000
ML=1
ML=5
ML=25
ML=125
0.00
2.00
4.00
6.00
8.00
10.00
NO. OF PROCESSORS