NASA Contractor Report 178035

ICASE REPORT NO. 86-7

NASA-CR-178035
19860012779

ICASE

AN OPTIMAL REPARTITIONING DECISION POLICY

David M. Nicol
Paul F. Reynolds, Jr.

Contract Nos. NAS1-17070 and NAS1-18107

February 1986

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

An Optimal Repartitioning Decision Policy

David M. Nicol
Institute for Computer Applications in Science and Engineering

Paul F. Reynolds, Jr.
University of Virginia

Abstract

A central problem to parallel processing is the determination of an effective partitioning of workload to processors. The effectiveness of any given partition is dependent on the stochastic nature of the workload. We treat the problem of determining when and if the stochastic behavior of the workload has changed enough to warrant the calculation of a new partition. We model the problem as a Markov decision process, and derive an optimal decision policy. Quantification of this policy is usually intractable; we empirically study a heuristic policy which performs nearly optimally. Our results suggest that the detection of change is the predominant issue in this problem.

This research was supported in part by the National Aeronautics and Space Administration under NASA Contracts NAS1-17070 and NAS1-18107 while the first author was in residence at ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, VA 23665. It was also supported in part by the Virginia Center for Innovative Technology, while both authors were in residence at the University of Virginia, Department of Computer Science, Thornton Hall, Charlottesville, VA 22903.

I. Introduction

We consider a multiprocessor system without a dynamic scheduling facility; e.g., a loosely coupled message passing system. We presume that some computation has been partitioned onto the processors, and that this partition cannot be easily changed while the computation is executing. The computation's execution time under this partition is presumed to be dependent on the stochastic behavior of the computation. As long as the stochastic behavior is the same as when the partition was chosen, we suppose that a different partition will not yield an execution time performance gain. If the stochastic behavior does change, a new partition may better exploit the new behavior. The problem is to detect change in the computation's stochastic behavior, determine the performance benefits of implementing a new partition, and weigh those benefits against the overhead costs of calculating and implementing a new partition. We treat the repartitioning decision problem as a Markov decision process. This problem should be distinguished from the partitioning (or task assignment) problem, which addresses *how* to partition a workload among processors. Rather, we are examining the issue of *when* to abandon one partition and adopt another. These two problems are closely related, as the overall performance depends both on how well and how often a workload is partitioned. Previous work in partitioning has focused only on the first problem. Our work considers how often a partitioning algorithm should be applied, as a function of the quality of the partitioning and the behavior of the computation under that partitioning.

Consider a computation which can be described as a sequence of "cycles" which are probabilistically identical, and which are independent of the computation's partitioning. Examples of such computations include iterative numerical methods, real-time systems which execute periodic monitoring tasks, and simulation programs [12]. The partitioning of such computations for parallel processing may take into account many factors, e.g. data dependencies, and system resource requirements. Once a partition is chosen, it may be quite difficult and/or expensive to dynamically move small portions of the computation's workload. Yet, the chosen partition may become quite unsatisfactory if the stochastic nature of the computation changes. We are then presented with the problem of needing to repartition, but are constrained to devising a completely new partition. The related literature falls into two categories, neither of which addresses this particular problem. The work reported in [5], [6], [9], and [25] essentially presumes that jobs arrive at a central dispatcher which assigns jobs to processors. Our problem eschews the job arrival model, and does not allow a dynamic routing mechanism. A more recent body of work including [7], [14], [22], [23], [26] allows decentralized assignment decisions to be made dynamically. Again, our problem presumes that incremental dynamic reassignment is not feasible. Static and dynamic task assignment algorithms are presented in [1], [2], [4], [10], [11], [13], [17], [24]. The dynamic assignment algorithms consider restricted classes of computations; the static partitioning algorithms might be used in conjunction with our repartitioning decision policy if the computations partitioned by such methods change their behavior. Our work is a variation on aspects of the broad treatment of change detection under uncertainty given in [18]. Our model modifies this analysis by using a different decision cost structure, and by assuming a random computation duration. Our work's main contribution is as the first application of statistical analysis and Markov decision theory to the performance issue of *when* to reconfigure a workload distribution in a parallel processing environment.

Suppose we measure the performance of a partitioned cyclic computation by observing the sum of processor utilizations over every cycle. Letting Z_i denote the *ith* utilization measurement, we suppose that the sequence $Z_1, Z_2, \cdots$ forms a weakly stationary stochastic process [20]. Intuitively, this means that every Z_i and Z_j have the same mean, the same variance, and that

their covariance depends only on $|i-j|$. We anticipate that at some unknown future time the stochastic behavior of this computation will change, at which point it may be advantageous to redistribute the computational workload. A change of this sort can be expected if the probabilistic behavior of the inputs driving the system change; such a change could also occur as a result of an intrinsic change in the behavior of the computation. In this paper we consider the problem of detecting and reacting to such a change. We develop a real-time change monitoring decision process which balances the expected costs of redistribution against the expected benefits. Within the constraints imposed by the change detection method, the decision policies we describe are shown to minimize the computation's expected execution time.

In section II we present a statistical means of determining when a computation's stochastic behavior changes. Section III shows how to dynamically maintain the probability of a change having already occurred. Section IV discusses our assumptions about the duration of the computation; section V formulates the repartitioning decision problem as a Markov decision process. Sections VI and VII analyze this model, and characterize a decision policy which minimizes the expected computation execution time. As the calculation of the optimal decision policy is usually intractable, we propose a heuristic policy in section VIII. Section IX reports empirical results which suggest that our heuristic tends to perform very close to optimally. Section X talks about the possibility of multiple changes during a computation, and section XI contains our conclusions.

II. Detecting Change

In this section we discuss a statistical approach to detecting change in the computation's stochastic behavior. The proposed method is not an integral part of our dynamic partitioning solution, it could be replaced with any other statistical change detection technique. Our method does have the advantage of computational simplicity with minimal storage requirements.

The change monitoring process examines the sequence $\{Z_i\}$ of performance measures. Each random variable Z_i is drawn from a common distribution F whose exact characterization is not known. We say that a *change* occurs at cycle j if Z_j's distribution is different from Z_{j-1}'s. We presume that at most one change will occur in the course of the computation. The probabilistic nature of this problem requires us to statistically determine when a change occurs. However, F is general, and the performance observations are correlated. For statistical tractability, we presume that the sequence $\{Z_i\}$ is first transformed into a sequence $\{X_i\}$ of approximately normal and independent observations. This is accomplished using the *batch means* [8] transformation, where a sequence of d observations is replaced by its mean value. Thus the transformed observation X_i is defined by

$$X_i = \frac{1}{d}\sum_{j=0}^{d-1} Z_{d\cdot i+j}.$$

If $Z_{d\cdot i+j}$ has mean μ and variance σ^2 for $i = 1, \cdots, d-1$, then X_i is approximately normal with mean μ and variance σ^2/d.

Our change monitor examines the observations $\{X_i\}$ as they become available to determine if and when their underlying normal distribution changes. Solutions to the so-called model identification problem [3] in statistics can be used to detect this change. In our situation, model identification decides whether two groups of independent normal observations are drawn from a single normal distribution.

Using a model identification approach, we create a test cluster of c adjacent normal observations. The *ith* cluster C_i is defined to be the collection

$$C_i = \{X_{c\cdot i}, \cdots, X_{c\cdot(i+1)-1}\}.$$

We assume the existence of a base cluster B of size c derived from initial observations taken before a change could have occurred. The model identification test determines whether observations in B and C_i are identically distributed. Positive indication of distributional difference is evidence for the change having already occurred. The AIC model identification approach described in [3] attempts to describe the data set $B \cup C_i$ with a probabilistic model having as few parameters as possible while still fitting the data. We consider two competing models of $B \cup C_i$. One model states that $B \cup C_i$ is a set of $2 \cdot c$ observations drawn from a normal distribution $N(\mu_J, \sigma_J^2)$. This model has two parameters, μ_J and σ_J^2. The competing four parameter model states that B is a set of normal $N(\mu_B, \sigma_B^2)$ observations and that C_i is a set of normal $N(\mu_C, \sigma_C^2)$ observations, $\mu_B \neq \mu_C$ and $\sigma_B^2 \neq \sigma_C^2$. For each model we calculate the statistic

$$AIC = -2 \cdot l(P) + 2 \cdot P$$

where P is the number of model parameters, and $l(P)$ is the maximized log-likelihood function [12] using P parameters. The model achieving the minimum AIC statistic is taken as the most parsimonious.

The statistic AIC_J for the two parameter model is given by

$$AIC_J = c \cdot ln(\hat{\sigma}_J^2) + 4$$

and the competing model's statistic is given by

$$AIC_s = \frac{c}{2} \cdot ln(\hat{\sigma}_B^2 \cdot \hat{\sigma}_C^2) + 8$$

where $\hat{\sigma}_J^2$, $\hat{\sigma}_B^2$, $\hat{\sigma}_C^2$ are the sample variances for the sets $B \cup C_i$, B, and C_i, respectively. One advantage of the AIC model selection criterion is its simplicity. The significance level of the statistic is effectively chosen by its derivation, and no statistical tables need to be stored. Furthermore, the calculations take linear time in the size of the clusters.

The model identification criterion may fail to correctly determine whether a cluster C_i represents the existence of a change. In a Bayesian framework, the AIC test gives us some indication of change, with uncertainty. If we have a prior probability of change, we can calculate a posterior probability of change as a function of the test result. This calculation requires knowledge of the statistical test's accuracy. We let α denote the probability that the model selection statistic falsely indicates a change (type I error); we let β denote the probability that the statistic fails to detect a change (type II error).

III. Calculating the Probability of Change

We now consider the calculation of the probability of change. A measurement of system utilization is taken every cycle. The mechanics of the batch means transformation produce a transformed observation every d cycles; a new test cluster C is thus available every $c \cdot d$ cycles. The beginning of cycle $n \cdot c \cdot d$ is called the *nth decision step* or *time* n, and is the *nth* epoch at which a test cluster is evaluated for change. The $c \cdot d$ cycles between decision steps are known as a *decision interval.* We define p_n to be the probability that a change has occurred by decision step n. p_n is calculated as a function of p_{n-1}, and the result of the statistical change test performed at time n. This function's description requires the evaluation of $p^*(p_{n-1})$, the probability that a

change **will have occurred** by time n, given only the value of p_{n-1}. $p^*(p_{n-1})$ can be calculated at time $n-1$, but must presume some prior knowledge of the distribution of the time of change. Supposing that such foreknowledge is not precisely known, it is both reasonable and convenient to assume that the failure rate of the time of change distribution is some constant ϕ. By conditioning on the time of change, we have

$$p^*(p_{n-1}) = p_{n-1} + \phi \cdot (1 - p_{n-1}) \tag{1}$$

$$= (1 - \phi) \cdot p_{n-1} + \phi.$$

$p^*(p_{n-1})$ is the probability of change by time n, *before* a change test is performed at time n. p_n depends both on $p^*(p_{n-1})$ and the result of the change test performed at decision step n. The posterior probability p_n is calculated using Bayes' Theorem [21] . If the test at decision step n indicates change, p_n is given by $p^c(p_{n-1})$:

$$p_n = p^c(p_{n-1}) = \frac{p^*(p_{n-1}) \cdot (1 - \beta)}{p^*(p_{n-1}) \cdot (1 - \beta) + (1 - p^*(p_{n-1})) \cdot \alpha}. \tag{2}$$

Given a negative indication of change, p_n is defined by $p^{\overline{c}}(p_{n-1})$:

$$p_n = p^{\overline{c}}(p_{n-1}) = \frac{p^*(p_{n-1}) \cdot \beta}{p^*(p_{n-1}) \cdot \beta + (1 - p^*(p_{n-1})) \cdot (1 - \alpha)}. \tag{3}$$

As the test clusters become available for statistical testing, equations (1)-(3) allow us to maintain the probability that the change has already occurred.

At each decision step we will decide whether to repartition the computational load. This decision should be based on the costs and benefits of repartitioning. Presuming we receive no substantial performance benefits if a change has **not** occurred, we see that the probabilities $\{p_n\}$ should play a pivotal role in determining whether (and when) repartitioning is worthwhile. Sections V and VI confirm this intuition.

IV. Number of Cycles

We presume that the computation will require a random number N decision steps, irrespective of its partitioning. We assume that N is bounded above by some constant M. Our analysis also assumes that the distribution of N has an increasing failure rate function. That is, the failure rate probability

$$\frac{Prob\{N = n\}}{Prob\{N \geqslant n\}}$$

is an increasing function of n. An intuitive explanation of this requirement is that the longer the computation continues, the more likely it is that the computation will stop with the next decision step.

We use an equivalent statement of increasing failure rate probabilities. To express this equivalency, we first define the $\geqslant_L$ (stochastically larger) relation between random variables. If X and Y are random variables, we say that $X \geqslant_L Y$ if for all increasing functions g, $E[g(X)] \geqslant E[g(Y)]$. Let N_n denote the random variable N conditioned on $N \geqslant n$. [20] shows that assuming N has an increasing failure rate function is equivalent to assuming that $N_n \geqslant_L N_{n+1}$ for all $n \geqslant 0$.

Finally, we let $\hat{N}_n$ denote the expected value of N, given that $N \geqslant n$.

V. A Markov Decision Process

Our notation concerning Markov decision processes is taken largely from [19]. Consider a stochastic process whose state we observe at each of a sequence of times $t = 0, 1, \cdots$. Let I be the set of all possible states. At each time j, the state of the process is discerned to be some $s \in I$. Then a decision is made, choosing some action a from a finite set A; the choice of action a while in state s incurs a cost $c(s,a)$. $c(s,a)$ may be random; we assume that $E[c(s,a)]$ is finite for all states s and actions a. The decision process then passes into another state. The probability $p_{sq}(a)$ of passing into state q from s is dependent on the action a chosen in state s. The expected total cost of a decision policy is the expected sum of the costs incurred at each decision step. An *optimal* decision policy minimizes the expected total cost.

We restrict our attention to the class of *stationary* decision policies, those policies which are deterministic functions of the discerned state. A useful theorem concerning optimal stationary decision policies is given by [19].

THEOREM 1 : Let $V(s)$ be the expected total cost of the process which starts in state s, and which is governed by the optimal stationary policy. Then,

$$V(s) = \min_{a \in A} \left\{ c(s,a) + \sum_{q \in I} p_{sq}(a) \cdot V(q) \right\}. \tag{4}$$

□

The function $V(s)$ is known as the *optimal cost function.* From state s, the optimal stationary decision is the choice of action which minimizes the right hand side of equation (4).

We now formulate the repartitioning decision problem as a Markov decision process. Table I summarizes the notation used by this decision process formulation. The decision process time steps will be precisely the decision steps. The decision costs are functions(in part) of e_o and e_r, the expected execution times of a decision interval after a change, with the original and with the new partition, respectively. We presume that $e_o > e_r$. A state of the process has the form $<p,n>$, reflecting the decision step n, and posterior probability of change, $p = p_n$. The decision process chooses to *test*, or to *retain*. The retain decision causes the current partition to be kept for the next decision interval. Our cost structure is concerned only with execution times after a change. Therefore, if the anticipated change has not occurred, the retain decision exacts cost 0. Otherwise, the retain decision causes the next decision interval execution to have a mean execution time of e_o. The expected cost of the retain decision in state $<p,n>$ is $p \cdot e_o$.

The decision process may decide to *test*. This decision causes the running system to halt; a new partition is calculated, and is tested against the old partition on recent workload profiles. We let D_d denote the delay caused by calculating and testing a new partition. Partition comparison can be placed in a decision theoretic context, as described in [15]. If the new partition is found superior, the change is considered to have occurred, and the decision process is considered to be *stopped*. A stopped process incurs an additional time cost D_r quantifying the time required to implement the new partition. A stopped process also incurs an execution cost e_r for every remaining decision interval in the computation. At time n, the expected value of this execution cost is equal to $e_r \cdot (\hat{N}_n - n + 1)$. If the change has occurred, the test decision incurs an expected cost $D_d + D_r + e_r \cdot (\hat{N}_n - n + 1)$. A premature test decision incurs only the calculation delay D_d. In this case the probability of change by time n is taken to be zero, and the decision process

Notation	Definition
n	Decision Step Number
N	Random number of decision steps
M	Upper Bound on N
$\hat{N}_n$	N given $N \geq n$
N_n	$E[\hat{N}_n]$
e_o	Decision Interval Post-Change Execution Time, Original Partition
e_r	Decision Interval Post-Change Execution Time, New Partition
D_d	Delay to Calculate and Test New Partition
D_r	Delay to Implement New Partition
α	Change Test Type I Error
β	Change Test Type II Error
ϕ	Time of Change Failure Rate Probability
$p^*(p)$	Pre-Observation Probability of Change At Next Decision Step
$p^c(p)$	Posterior Probability of Change After Positive Change Observation
$p^{\bar{c}}(p)$	Posterior Probability of Change After Negative Change Observation
$q^c(p)$	Probability of Observing Change Next Observation
$q^{\bar{c}}(p)$	Probability of Not Observing Change Next Observation
$V(<p,n>)$	Optimal Cost Function
$E_v[<p,n>]$	Expected Future Value of $V(<\cdot,n+1>)$ from $<p,n>$
$R(<p,n>)$	Optimal Future Costs Given Retain at $<p,n>$
$T(<p,n>)$	Optimal Future Costs Given Test at $<p,n>$

Table I

continues. The expected cost of choosing to test in state $<p,n>$ is thus given by

$$D_d + p\cdot\left(e_r\cdot(\hat{N}_n - n + 1) + D_r \right) .$$

We now consider the state transition probabilities. The state following a retain decision from $<p,n>$ depends on the result of the change detection test to be performed at time $n + 1$. From state $<p,n>$, the probability of observing a change at time $n + 1$ is given by $q^c(p)$, found by conditioning on the time of change:

$$q^c(p) = p^*(p)\cdot(1 - \beta) + (1 - p^*(p))\cdot\alpha. \tag{5}$$

From $<p,n>$ the probability $q^{\bar{c}}(p)$ of *not* observing a change at time $n + 1$ is just $1 - q^c(p)$. Given a retain decision in $<p,n>$, the decision process passes into state $<p^c(p),n+1>$ with probability $q^c(p)$; it passes into state $<p^{\bar{c}}(p),n+1>$ with probability $q^{\bar{c}}(p)$.

The optimal cost function's state transition component is concisely represented by the following function. Let

$$E_v(<p,n>) = q^c(p)\cdot V(<p^c(p),n+1>) + q^{\bar{c}}(p)\cdot V(<p^{\bar{c}}(p),n+1>). \tag{6}$$

$E_v(<p,n>)$ is interpreted as the minimized expected future costs at time $n+1$, as seen from state $<p,n>$ after a retain decision.

The decision to retain in $<p,n>$ incurs an expected cost $p \cdot e_o$; the minimal expected *future* costs are given by $E_v(<p,n>)$. The expected future cost of the policy which retains in $<p,n>$, and thereafter uses the optimal stationary policy is thus

$$R(<p,n>) = p \cdot e_o + E_v(<p,n>). \tag{7}$$

Similarly, the decision to test in $<p,n>$ incurs an expected cost $D_d + p \cdot (e_r \cdot (\hat{N}_n - n + 1) + D_r)$. No other costs are incurred if the process stops. If instead the process rejects the new partition, the probability of change is taken to be zero, the state transition probabilities into time $n+1$ are identical to those after a retain decision from state $<0,n>$. Thus the minimal expected future costs in this case are simply $E_v(<0,n>)$. The expected future cost of the policy which retains in $<p,n>$ and thereafter uses the optimal stationary policy is thus

$$T(<p,n>) = D_d + p \cdot \Big(e_r \cdot (\hat{N}_n - n + 1) + D_r \Big) + (1 - p) \cdot E_v(<0,n>). \tag{8}$$

In terms of equations (7) and (8), Theorem 1 states that

$$V(<p,n>) = \min \Big\{ T(<p,n>), R(<p,n>) \Big\}, \tag{9}$$

so that the optimal stationary decision in state $<p,n>$ is to retain if and only if $R(<p,n>) \leqslant T(<p,n>)$.

Equations (6)-(9) illustrate the recursive relationship satisfied by the optimal cost function. Since the number of decision steps is bounded above by M, we can define $V(<p,M+1>) = 0$ for all $p \in [0,1]$, and can solve for $V(<p,n>)$ when $0 \leqslant n \leqslant M$. We next show that the solution of $V(<p,n>)$ is nicely characterized without explicit quantification.

VI. Properties of $V(<p,n>)$

We will demonstrate that the optimal stationary decision policy is given by a sequence $\pi_0, \pi_1, \cdots$ of thresholds from the interval [0, 1]. The optimal decision in state $<p,n>$ is to test if and only if $p > \pi_n$. This structure is revealed by analysis of $T(<p,n>)$ and $R(<p,n>)$ for fixed n as a functions of p. We will show that for every n, there exists π_n such that whenever $p \leqslant \pi_n$ then $R(<p,n>) \leqslant T(<p,n>)$, and whenever $p > \pi_n$ then $R(<p,n>) > T(<p,n>)$. Our vehicle for this result is the demonstration that for fixed n, $T(<p,n>)$ is linear in p, and $R(<p,n>)$ is concave in p. We analyze the values of these functions at their endpoints and argue that $T(<p,n>)$ and $R(<p,n>)$ can intersect at most once, at $p = \pi_n$.

Some of our analysis conditions on the value of N. We use the notation $f(<p,n> | N=m)$ to denote the value of function f at state $<p,n>$ given that $N = m$.

Our first observation is that for any fixed n, $T(<p,n>)$ is a linear function of p. This is apparent from equation (8), as the value $E_v(<0,n>)$ is independent of p. Thus

LEMMA 1 : For fixed n, $T(<p,n>)$ is a linear function of p.

□

We next observe that for fixed n, $R(<p,n>)$ is a piece-wise linear continuous concave (*plcc*) function of p. This result follows primarily from the following lemma reported in [18] and stated in terms of our notation:

LEMMA 2 : Suppose that $N = m$. If $V(<p,n+1>|\, N=m)$ is a *plcc* function of p, then $E_v(<p,n>|\, N=m)$ is a *plcc* function of p.

□

We use this lemma to establish another.

LEMMA 3 : For every fixed $n \geqslant 0$, $V(<p,n>)$ and $R(<p,n>)$ are *plcc* functions of p.

PROOF: We first condition on $N = m$ for some m. We then inductively show that $V(<p,n>|\, N=m)$ and $R(<p,n>|\, N=m)$ are *plcc* functions of p. For the base case we consider $n = m$. For any $p \in [0,1]$,

$$R(<p,m>|\, N=m) = p \cdot e_0 + E_v(<p,m>|\, N=m)$$

$$= p \cdot e_0$$

since $V(<p,m+1>|\, N=m) = 0$ for all p. Thus $R(<p,m>|\, N=m)$ is *plcc* in p. We also observe that $T(<p,m>|\, N=m)$ is *plcc* since it is linear. The class of *plcc* functions is closed under the pointwise minimum operation; $V(<p,m>|\, N=m)$ must also be *plcc*, establishing the induction base.

For the induction hypothesis we suppose that both $R(<p,n+1>|\, N=m)$ and $V(<p,n+1>|\, N=m)$ are *plcc* functions of p for some $n \leqslant m - 1$. Lemma 2, and the closure of *plcc* functions under addition and pointwise minimum again ensure that $R(<p,n>)$ and $V(<p,n>)$ are *plcc* functions of p, completing the induction.

To complete the proof, we note that the class of *plcc* functions is also closed under scalar multiplication, and observe that

$$V(<p,n>) = \sum_{m=0}^{M} Prob\{N = m\} \cdot V(<p,n>|\, N=m)$$

and

$$R(<p,n>) = \sum_{m=0}^{M} Prob\{N = m\} \cdot R(<p,n>|\, N=m)$$

□

We next analyze the values of $T(<p,n>)$ and $R(<p,n>)$ at $p = 1$. We show that $V(<1,n>)$ is a well-behaved function of n.

LEMMA 4 : Either

(i) $V(<1,n>) = T(<1,n>)$ for all n for which $Prob\{N = n\} \neq 0$; or

(ii) $V(<1,n>) = R(<1,n>)$ for all n for which $Prob\{N = n\} \neq 0$; or

(iii) There exists an n_0 (possibly ∞) such that for all $n < n_0$, $V(<1,n>) = T(<1,n>)$, and for all $n \geqslant n_0$ for which $Prob\{N = n\} \neq 0$, $V(<1,n>) = R(<1,n>)$.

PROOF: We condition on $N = m$, for any $0 \leqslant m \leqslant M$. Let K be the largest integer such that $(e_0 - e_1) \cdot K \leqslant D_d + D_r$. Simple algebra (omitted here) establishes the inductive proof that for all n such that $m - K < n \leqslant m$,

$$V(<1,n>|\, N=m) = R(<1,n>|\, N=m) = e_o \cdot (m - n + 1);$$

and that for $0 \leqslant n \leqslant m - K$,

$$V(<1,n>|\, N=m) = T(<1,n>|\, N=m) = D_d + D_r + e_r \cdot (m - n + 1)$$

and

$$R(<1,n>|\, N=m) = D_d + D_r + e_o + e_r \cdot (m - n).$$

Define $d(n|\, N=m)$ to be the conditional difference $T(<1,n>|\, N=m) - R(<1,n>|\, N=m)$, and $d(n)$ to be the unconditional difference $T(<1,n>) - R(<1,n>)$. From the equations above, we see that as a function of m,

$$d(n|\, N=m) = \begin{cases} D_d + D_r - (e_o - e_r) \cdot (m - n + 1) & \text{for } m < n+K \\ (e_r - e_o) & \text{for } n + K \leqslant m \end{cases}.$$

It follows from the definition of K that $d(n|\, N=m)$ is a decreasing function of m. The unconditional difference $d(n)$ is obtained by taking the expectation of $d(n|\, N=m)$ with respect to the residual distribution N_n of N given $N \geqslant n$. Recalling our remarks in section 4, we have effectively assumed that $N_n \geqslant_L N_{n+1}$ for all $n \geqslant 0$, implying that $E[g(N_n)] \leqslant E[g(N_{n+1})]$ for all *decreasing* functions g. In particular, $d(n) \leqslant d(n+1)$, showing that the difference $T(<1,n>) - R(<1,n>)$ is an *increasing* function of n. Then case (i) occurs if $d(n)$ is negative for all n, case (ii) occurs if $d(n)$ is positive for all n, and case (iii) occurs if $d(n)$ changes sign at $n = n_0$.

□

We summarize the known behavior of $V(<p,n>)$ as a function of p. $R(<p,n>)$ is a *plcc* function of p, and $T(<p,n>)$ is linear in p. From this we infer that if $T(<1,n>) \leqslant R(<1,n>)$, then the functional curves of $T(<p,n>)$ and $R(<p,n>)$ can intersect at most once for $p \in [0,1]$ (otherwise the concavity of $R(<p,n>)$ is violated). The last lemma shows that either $T(<1,n>) \geqslant R(<1,n>)$ for all $n \geqslant 0$, or that $T(<1,n>) \leqslant R(<1,n>)$ only if n is less than some threshold n_0 (potentially ∞). Furthermore, it is easily seen that

$$T(<0,n>) - R(<0,n>) = D_d > 0$$

for all n. These observations collectively establish the structure of the optimal stationary decision policy for small enough n: if $n \leqslant n_0$ let π_n be the unique solution to the equation $T(<p,n>) = R(<p,n>)$. The existence of this solution is ensured by continuity, and the fact that T exceeds R at $p = 0$ while R exceeds T at $p = 1$. The optimal policy is to retain in all states $<p,n>$ such that $p \leqslant \pi_n$, and to test in states $<p,n>$ such that $p > \pi_n$. Figure I illustrates this argument, showing plots of $R(<p,n>)$ and $T(<p,n>)$ as functions of p when $n \leqslant n_0$.

We complete the analysis of $V(<p,n>)$'s behavior by supposing $n > n_0$, so that $R(<1,n>) < T(<1,n>)$. The following lemma demonstrates that when $n > n_0$, $R(<p,n>)$ is *linear* in p. Since T is linear and exceeds R at $p = 0$ and $p = 1$ for such n, the functional curves for $T(<p,n>)$ and $R(<p,n>)$ cannot intersect.

LEMMA 5 : If $n > n_0$, then $R(<p,n>)$ is linear in p, and $V(<p,n>) = R(<p,n>)$ for all $p \in [0,1]$.

PROOF: We proceed by induction. M is the largest integer such that $Prob\{N = M\} \neq 0$, so that $V(<p,M+1>) = 0$ for all p and

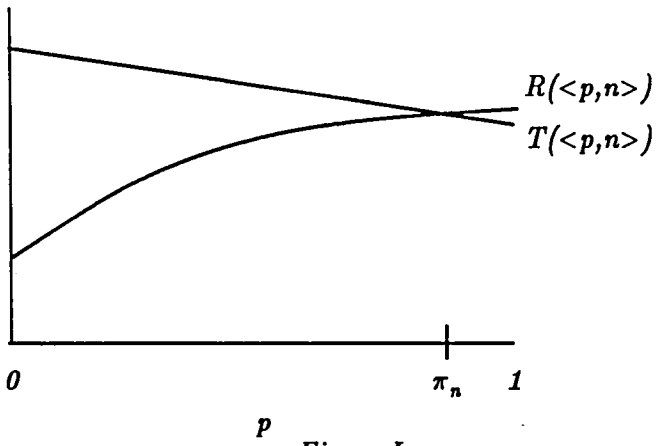

Figure I

$$V(<p,M>) = \min\begin{cases} D_d + p\cdot(e_r + D_r) \\ p\cdot e_o \end{cases}.$$

Presuming that $n_0 < M$, we have $V(<p,M>) = R(<p,M>) = p\cdot e_o$, which is linear in p.

For the induction hypothesis, we suppose there is an $n > n_0$ such that $V(<p,n+1>) = R(<p,n+1>)$ for all $p \in [0,1]$, and that $R(<p,n+1>)$ is linear in p. Equation (7) implies that

$$R(<p,n>) = p\cdot e_o + q^c(p)\cdot V(<p^c(p),n+1>) + q^{\overline{c}}(p)\cdot V(<p^{\overline{c}}(p),n+1>),$$

and the induction hypothesis states that

$$V(<p,n+1>) = A\cdot p + B$$

for some A and B. Equations (2), (3), and (5) show that that $p^c(p) = \frac{p^*(p)\cdot(1-\beta)}{q^c(p)}$ and that $p^{\overline{c}}(p) = \frac{p^*(p)\cdot\beta}{q^{\overline{c}}(p)}$; it follows that

$$R(<p,n>) = p\cdot e_o + A\cdot p^*(p) + B$$

$$= p\cdot\Big(e_o + A\cdot(1-\phi)\Big) + \Big(A\cdot\phi + B\Big).$$

Then $R(<p,n>)$ is linear in p; since $T(<p,n>)$ exceeds $R(<p,n>)$ at both $p = 0$ and $p = 1$, it follows directly that $T(<p,n>)$ exceeds $R(<p,n>)$ for all $p \in [0, 1]$. Thus $V(<p,n>) = R(<p,n>)$, completing the induction.

□

The discussions developed in this section prove our main analytic result.

THEOREM 2 : For every n, there exists a $\pi_n \in [0,1]$ such that the optimal stationary decision in state $<p,n>$ is to retain if $p \leqslant \pi_n$, and to test if $p > \pi_n$.

□

VII. Minimization of Expected Execution Time

We have constructed a decision process and exposed the structure of its optimal stationary policy. We next show that employment of the optimal decision policy minimizes the computation's expected execution time.

Our model formulation does not impose any execution costs until a change actually occurs. The optimal decision policy minimizes the expected sum of all overhead delays and all post-change execution delays. The overall expected finishing time is equal to the expected sum of all execution and overhead delays. However, the pre-change execution delays are independent of any decision policy. By minimizing the expected overhead and post-change execution delays, we minimize the expected finishing time.

The optimality of our derived policy is conditioned on constant model parameters. Modification of these parameters may change the optimal decision policy without changing the computation in any functional sense. For example, we might decrease the batch means set size d or the *AIC* cluster size b to decrease the decision interval length. This modification would increase the responsiveness of the decision model to change, at the cost of increased error probabilities α and β. This change in no way affects the functional behavior of the computation. Likewise, our policy depends on both the quality of partitions created by the partitioning algorithm, and that algorithm's running time. Our decision policy would be a valuable tool in exploring the tradeoffs between partition quality and the run-time required to achieve that quality.

VIII. A Repartitioning Heuristic

In this section we note that solving for the precise optimal thresholds π_n is not computationally feasible, and examine a simple heuristic which nearly minimizes the expected computation execution time. We furthermore observe that the timely detection of change is the most important component of our repartitioning decision heuristic.

In theory, the optimal cost equations (9) can be solved recursively. M was defined to be the largest integer such that $Prob\{N = M\} \neq 0$. The recursive solution begins with

$$V(<p,M>) = \min\{p \cdot e_o,\ D_d + p \cdot (e_r + D_r)\},$$

and then solves for decreasing n using equations (6)-(9). However, $V(<p,n>)$ is piecewise linear with the number of pieces tending to double at each step of the recursion. An exact solution is not computationally feasible for any large M. Approximation techniques might be employed, but even then the solution method could require more computation than its results justify. Furthermore, our decision model impractically presumes that the values of e_o and e_r are known a priori. We describe a heuristic which is based on the optimal decision policy structure. Empirical tests indicate that the heuristic yields total policy costs which are extremely close to the optimal policy's cost.

The optimal decision policy's structure suggests that a heuristic be focused on the probability of change. We have shown already how this probability can be dynamically maintained. The first task for our heuristic is to determine when to estimate e_o and e_r. We want the heuristic to be responsive to a change, and yet we don't want premature estimations of these execution time means. Before a change occurs, most change detection tests will report no change, and the posterior probability of change is calculated using the function $p^{\overline{c}}(p)$. In [15] we show that $p^{\overline{c}}(p)$ is a contraction mapping[16], implying that so long as no change has occurred, the probability of change will be close to the fixed point solution $q = p^{\overline{c}}(q)$. If p_n is significantly greater than this q we can be reasonably sure that a change did occur. Following this reasoning, we choose an initial

threshold p_e so that e_r and e_o are estimated whenever $p_n > p_e$. p_e is chosen so that starting with a prior probability equal to $p^{\overline{c}}(p)$'s fixed point solution, three successive positive indications of change are needed to exceed p_e. After estimating e_o and e_r, the value of n_0 is determined (in $O(M)$ time). If the current time step n_e exceeds n_0, the original partition is retained for the rest of the computation. Otherwise, a high probability threshold $\rho = .8$ is chosen. We approximate π_n for time steps n between n_e and n_0 linearly by

$$\rho_n = \rho + (1 - \rho)\cdot\frac{(n - n_e)}{(n_0 - n_e)}.$$

The heuristic policy then mimics the optimal policy, treating each ρ_n as though it were π_n. A premature choice of the test decision causes the heuristic to abandon the $\{\rho_n\}$, and wait again for the probability of change to exceed p_e before recalculating the $\{\rho_n\}$.

IX. An Empirical Study

We performed an empirical study comparing this heuristic's performance with the optimal decision policy. The exact optimal decision policy cannot be calculated except for small M. For large M we used a computationally expensive approximation to the optimal policy. At every step in the recursive solution, this approximation retained at most 1024 linear segments of an approximation to $V(<p,n+1>)$. These segments were used to calculate the approximately 2048 linear segments of (the approximation to) $V(<p,n>)$. The 1023 segments closest to $p = 1$ were retained; the remainder of $V(<p,n>)$ was approximated by a single line segment extending from $p = 0$ to the leftmost of the retained segments. This approximation was designed to closely model the behavior of optimal cost function in the region of the optimal policy threshold.

Our study varied two parameters: N and $G = e_o - e_r$. For simplicity, N was considered to be constant. Given values for N and G, the other model parameters had the following values:

e_o	200	ϕ	1/N
e_r	200-G	α	0.2
D_d	100	β	0.05
D_r	100		

Table II

The rational behind this quantification of e_o, e_r, D_d, and D_r was that we expect these values to have the same order of magnitude. ϕ was chosen with the philosophy that if we know very little about a potential change, a reasonable guess is that it is (approximately) as likely as not that a change will occur. α and β model our experience with using the AIC statistic. Our selection of N and G as free parameters followed from our intuition (borne out by computational experience) that optimal behavior is influenced most by these two parameters.

We allowed G to take on the values 5, 50, and 100, thereby spanning two orders of magnitude. N was assigned the values 10, 50, 100, and 1000. We first measured the relative difference $\%_n$ in finishing time between the computation under the approximated optimal policy and under the policy which always retains, i.e., does nothing. Relative to the total finishing time, $\%_n$ is the maximal percentage gain we can hope to achieve. We then calculated $\%_H$, the percentage of this maximal gain achieved by our heuristic. Table *III* contains the result of these calculations; each measurement is plus or minus 0.5% with a confidence of at least 95%.

N	G	$\%_n$	$\%_H$	G	$\%_n$	$\%_H$	G	$\%_n$	$\%_H$
10	5	0.0		50	4.7	55.2	100	19.2	75.3
50	5	0.48	54.8	50	11.3	93.4	100	32.4	95.5
100	5	0.5	82.9	50	12.2	95.1	100	33.9	97.1
1000	5	0.94	98.3	50	12.5	99.5	100	34.3	99.5

Table III

We can draw a number of conclusions from Table *III*. Obviously, when the gain G achievable by repartitioning is very small, there is very little difference between using the optimal policy, and the policy which always retains. If G is substantially larger, we can expect significant improvement in completion time by using a good decision policy. Furthermore, our proposed heuristic achieves most of the possible repartitioning gain. In fact, the relative difference in finishing time between our heuristic and the optimal decision policy is usually a fraction of one percent. The length of the computation also has a significant effect on our performance figures. As N grows, the performance of our heuristic tends to the optimal performance. It therefore appears that when G is precisely known, then our heuristic can be expected to yield nearly optimal performance.

The experiments summarized by Table *III* assumed that the heuristic could accurately assess e_o and e_r. Since these quantities are difficult to predict, we tested the heuristic's performance when it miss-estimated G. Miss-estimation of G affects our heuristic by altering the time step threshold n_0, which in turn alters the $\{\rho_n\}$. We tested the heuristic using the parameter values illustrated in Table *III*. For each pair of fixed G and N, we caused the heuristic policy to miss-estimate G by factors of 10^{-3}, 10^{-2}, 10^{-1}, 10, 10^2 , and 10^3. Table *IV* lists the resulting $\%_H$

G	N	10^{-3}	10^{-2}	10^{-1}	10	10^2	10^3
50	10	-7.1	-16.7	-19.4	44.5	48.3	44.1
100	10	5.2	-3.5	0	77.0	77.1	74.3
5	50	35.2	35.1	34.6	-67.1	-97.7	-108.2
50	50	10.0	23.1	33.8	93.9	94.2	92.4
100	50	14.1	27.0	82.0	95.6	96.4	95.1
5	100	11.4	21.7	22.3	49.6	42.9	42.6
50	100	53.1	50.5	86.9	95.2	96.4	96.1
100	100	55.7	53.2	95.7	97.2	97.5	97.6
5	1000	92.7	92.6	95.3	98.4	98.3	98.4
50	1000	94.8	98.0	99.6	99.7	99.7	99.7
100	1000	95.2	99.2	99.6	99.7	99.7	99.7

Table IV

as a function of G, N, and the miss-estimation factor.

Study of Table *IV* leads to several observations. Table *IV* clearly shows that the performance of the heuristic becomes insensitive to miss-estimation as N increases. This phenomenon parallels the observation from Table *III* that the performance of the heuristic increases with N. Both observations follow from the fact the the expected total gain from a new partition increases in N, while the expected costs of not being optimally responsive after a change are constant. It is also clearly more harmful to underestimate G than it is to overestimate it. When N is small, underestimation leads to the conclusion that $n_0 < n_e$, so that no new partition is adopted. The

heuristic's costs are somewhat larger in this case than the true "always retain" policy, due to the costs of estimating G. As N grows, it becomes more likely that, even though G is underestimated, the change occurs soon enough so that a new partition is adopted with a high enough probability of change. By overestimating G, it becomes possible to adopt a new partition when the benefits of doing so do not outweigh the costs D_d and D_r. This unhappy situation is realized only when both N and G are small. The most important conclusion we can draw from Table *IV* is that for most reasonable values of G and N, gross miss-estimation of G does not seriously affect our heuristic's performance. This implies that our heuristic's most critical feature is its ability to detect change.

X. Multiple Changes

Our decision model presumes that at most one change will occur during the computation. We were dissuaded from incorporating multiple changes as we felt that the additional complexity might not lead to a better understanding of the problem. This opinion is supported by the intractability of finding the optimal policy assuming a *single* change, and the high performance of our simple heuristic. Nevertheless, we should consider how multiple changes might be handled.

Only minor modifications are required to adapt our heuristic to the possibility of multiple changes. These modifications are invoked after a change occurs. Unlike the single change policy, the decision process does not stop after adopting a new partition. It remains active, looking for the next change. To do this, it must first collect a new base group B of observations which characterize the post-change behavior. The test cluster which triggered the new partition might be used as this B. The decision policy then continues just as before. The emperical results which imply that change detection is the most critical element of a repartitioning decision also suggest that this approach to multiple changes should work well.

XI. Conclusions

A good partitioning of a computation across multiple processors must make certain assumptions about the computation's running behavior. If that behavior were to radically change in the middle of the computation[1], those assumptions could be invalidated, so that the partition is no longer effective in reducing the running time of the computation. We have considered the problem of when to reject an old partition, and adopt a new one. We proposed the use of proven statistical techniques to detect change in a computation's stochastic behavior; we modeled the repartitioning decision problem as a Markov decision process. This decision process takes into account the critical costs and benefits of repartitioning. We then characterized the decision policy which minimizes the expected computation finishing time. While this policy can be intuitively described, it is not easily quantified. We thus proposed and studied a heuristic decision policy which is modeled after the optimal policy. This heuristic performs remarkably well over a wide range of parameter values. Furthermore, it is quite insensitive to miss-estimation of the repartitioning gain. This observation leads us to conclude that the ability to detect change is the most important feature of a repartitioning policy. If we can detect change, and if we can expect to achieve more than marginal gain from repartitioning after a change, then we can expect our proposed policy to achieve most of the possible repartitioning gain.

---

1. We are currently considering different models of stochastic behavior in "Load Balancing Computations with Non-Stationary Behavior", D.M. Nicol and J.H. Saltz, ICASE Report in preparation, 1986.

References

[1] M.J. Berger and S. Bokhari, "The Partitioning of Non-Uniform Problems",*ICASE Report No. 85-55*, November 1985.

[2] S. Bokhari, "Partitioning Problems in Parallel, Pipelined, and Distributed Computing", *ICASE Report No. 85-54*, November 1985. (NASA CR-178023.)

[3] H. Bozedogan and S. Sclove, "Multi-Sample Cluster Analysis Using Akaike's Information Criterion", *Annals of the Institute of Statistical Mathematics 36,1* (1983).

[4] W.W. Chu, L.J. Holloway, M. Lan, and K. Efe, "Task Allocation in Distributed Data Processing", *Computer, 13, 11*, November 1980, 57-69.

[5] Y. Chow and W. Kowhler, "Models for Dyanamic Load Balancing in a Heterogeneous Multiple Processor System", *IEEE Trans. on Computers, C-28, 5*, (May 1979), 354-361.

[6] T. C. Chou and J. A. Abraham, "Load Balancing in Distributed Systems", *IEEE Trans. on Software Eng., 8*, 4 (July 1982), 401-412.

[7] D.L. Eager, E.D. Lazowska, J. Zahorjan, "A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing (Extended Abstract)", *Proceedings of the 1985 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, August 1985, 1-3.

[8] G. S. Fishman, "Grouping Observations in Digital Simulation", *Management Science 25*, (1978), 510-521.

[9] G. J. Foschini, "On Heavy Traffic Diffusion Analysis and Dynamic Routing in Packet Switched Networks", in *Computer Performance*, K. M. Chandy and M. Reiser Eds. New York: North-Holland, 1977.

[10] D. Gusfield, "Parametric Combinatorial Computing and a Problem of Program Module Distribution", *Journal of the ACM, 30, 9*, July 1983, 551-563.

[11] D.I. Moldovan and J.A.B. Fortes, "Partitioning and Mapping Algorithms into Fixed Size Systolic Arrays", *IEEE Trans. on Computers, C-35, 1*, (January 1986), 1-12.

[12] J. G. Kalbfleish, *Probability and Statistical Inference II*, Springer-Verlag, 1979.

[13] P.R. Ma, E.Y.S. Lee, and M. Tsuchiya, "A Task Allocation Model for Distributed Computing Systems", *IEEE Trans. on Computers, C-31, 1*, January 1982, 41-47.

[14] L. M. Ni, C. Xu, and T.B. Gendreau, "A Distributed Drafting Algorithm for Load Balancing", *IEEE Trans. on Software Engineering, SE-11, 10*, October 1985, 1153-1161.

[15] D. M. Nicol and P. F. Reynolds, Jr., "The Automated Partitioning of Simulations for Parallel Execution", *University of Virginia Department of Computer Science Tech Report TR-85-15*, August 1985.

[16] G. M. Phillips and P. J. Taylor, *Theory and Applications of Numerical Analysis*, Academic Press, 1973.

[17] C.C. Price, U.W. Pooch, "Search Techniques for a Nonlinear Multiprocessor Scheduling Problem", *Naval Research Logistics Quarterly, 29, 2*, June 1982, 213-233.

[18] A. Rapoport, W. E. Stein, and G. J. Burkheimer, *Response Models for Detection of Change*, D. Reidel Publishing Company, Boston, 1979.

[19] S. Ross, *Applied Probability Models with Optimization Applications*, Holden-Day, San Francisco, 1970.

[20] S. Ross, *Stochastic Processes*, Wiley and Sons, New York, 1983.

[21] S. A. Schmitt, *An Elementary Introduction to Bayesian Statistics*, Addison-Wesley, 1969.

[22] J. A. Stankovic, "An Application of Bayesian Decision Theory to Decentralized Control of Job Scheduling", *IEEE Trans. on Computers, C-34*, 2 (Feb 1985), 117-130.

[23] J. A. Stakovic, K. Ramamritham and S. Cheng, "Evaluation of a Flexible Task Scheduling Algorithm for Distributed Hard Real-Time Systems", *IEEE Trans. on Computers, C-34, 12 (December 1985), 1130-1143.*

[24] H.S. Stone, "Critical Load Factors in Distributed Computer Systems", *IEEE Trans. on Software Engineering, SE-4, 3* (May 1978), 254-258.

[25] D. Towsley, "Queueing Network Models with State-Dependent Routing", *Journal of the ACM,* 27, 2 (April 1980) 323-337.

[26] A. N. Tantawi and D. Towsley, "Optimal Static Load Balancing", *Journal of the ACM, 32*, 2 (April 1985), 445-465.

Standard Bibliographic Page

1. Report No. NASA CR-178035 ICASE Report No. 86-7	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle An Optimal Repartitioning Decision Policy		5. Report Date February 1986 6. Performing Organization Code
7. Author(s) David M. Nicol and Paul F. Reynolds, Jr.		8. Performing Organization Report No. 86-7
9. Performing Organization Name and Address Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225		10. Work Unit No. 11. Contract or Grant No. NAS1-17070, NAS1-18107
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		13. Type of Report and Period Covered Contractor Report 14. Sponsoring Agency Code 505-31-83-01

15. Supplementary Notes

Langley Technical Monitor:
J. C. South

Submitted to IEEE Trans. on Software Engineering

Final Report

16. Abstract

A central problem to parallel processing is the determination of an effective partitioning of workload to processors. The effectiveness of any given partition is dependent on the stochastic nature of the workload. We treat the problem of determining when and if the stochastic behavior of the workload has changed enough to warrant the calculation of a new partition. We model the problem as a Markov decision process, and derive an optimal decision policy. Quantification of this policy is usually intractable; we empirically study a heuristic policy which performs nearly optimally. Our results suggest that the detection of change is the predominant issue in this problem.

17. Key Words (Suggested by Authors(s))	18. Distribution Statement
Parallel Processing, Partitioning, load balancing, decision process	66 - Systems Analysis Unclassified - unlimited

19. Security Classif.(of this report)	20. Security Classif.(of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	18	A02

For sale by the National Technical Information Service, Springfield, Virginia 22161