Concise Papers

Warm Standby in Hierarchically Structured Process-Control Programs

Ing-Ray Chen and Farokh B. Bastani

Abstract—**We classify standby redundancy design space in process-control programs into the following three categories: cold standby, warm standby, and hot standby. Design parameters of warm standby are identified and the reliability of a system using warm standby is evaluated and compared with that of hot standby. Our analysis indicates that the warm standby scheme is particularly suitable for long-lived unmaintainable systems, especially those operating in harsh environments where burst hardware failures are possible. The feasibility of warm standby is demonstrated with a simulated chemical batch reactor system.**

Index Terms—**Fault tolerance, process-control, standby replacement, replication, knowledge representation, reliability assessment.**

I. Introduction

Process-control programs, such as those for controlling manufacturing systems, can often be organized in a multilevel hierarchical control structure where higher level processes formulate long-term control strategies, e.g., optimizing resource management, whereas lower level processes perform real-time control functions [1], [13]. The long-term nature of the decisions made by non real-time upper level processes means that the system may be able to tolerate temporary loss of such processes, e.g., by using suboptimal strategies. However, the loss of critical real-time processes can disrupt the whole system. This suggests that different fault tolerance techniques should be adopted for upper and lower level processes since their reliability requirements are quite different.

The standby replacement approach [2], [6], [7], [12] is an economical and efficient way of achieving fault tolerance at reasonable cost for processes in the control hierarchy. In one scheme, termed "cold standby," only one copy of each process is active at a time, and each copy is allocated to a processor that is designed to be fail-stop [11], i.e., able to detect if it contains a fault during the normal course of operation by using redundant hardware such as a self-checking circuit [8],[10]. When a processor is faulty, processes which reside on the failed processor are assigned to a spare processor or other functional processors if no spare processors are available. The main disadvantage of this "cold standby" scheme is its long recovery time to load and restart a backup copy. This problem is overcome by using the "hot standby" scheme where two or more copies are allowed to run at the same time on different fail-stop processors, with one copy serving as the primary and the others serving as active backups. When the primary copy fails (due to the failure of the processor on which it resides), a backup copy running on another processor can take over instantaneously without any recovery time delay. However, this "hot standby" approach requires up-to-date copies of a process and may not be cost-effective for upper level processes which do not require instantaneous recovery.

This concise paper develops a warm standby scheme in which the copies of a process may be *partial* copies instead of full copies as in the "hot standby" scheme. In the design space of replication, we envision that a) for cold standby, there is only a single active copy of the process and, hence, there are no other active copies; b) for hot standby, there are multiple active, full copies of a process; c) for warm standby, there are also multiple active copies of a process, some of which are partial copies. Warm standby is suitable for upper level processes because it incurs medium cost and moderate recovery time delay as compared with other standby schemes, although it potentially can also be used for lower level processes that are less time-critical.

The rest of the concise paper is organized as follows. Section II defines the meaning of a warm standby copy in a hierarchically structured system as opposed to a hot standby copy, and identifies the design parameters of the warm standby scheme. Section III presents a reliability analysis of warm standby and a simulation evaluation using a simulated chemical batch reactor. Finally, Section IV concludes the paper and outlines some future research areas.

II. Definition of Warm Standby Copies

We first define our fault model. We assume that if a processor fails then its failure is detected by redundant hardware and it ceases operation. In no cases does a machine behave unexpectedly. This assumption can be satisfied using techniques based on fail-stop processors [11].

As an example of warm standby copies, consider a part of a process-control system where a temperature profile is controlled by a control process according to a prescribed optimal time-temperature curve. This control process monitors the temperature sensor input and calculates the actuator output for effecting temperature changes (with a goal of minimizing the mean square error between the actual temperature profile and the optimal temperature profile). To tolerate possible failure of the control process, a standby copy is created in another computer. The standby copy can be implemented in three ways: a) it has the same view of the control information as that of the primary copy and the frequency of receiving the temperature sensor input is the same as that of the primary copy, b) it only has a partial view of the control information and, hence, the frequency of receiving the sensor input is less than that of the primary copy, and c) it does not have any view of the control information and, hence, the frequency of receiving the sensor input is zero. These three implementations correspond to the hot standby, warm standby, and cold standby schemes, respectively. Fig. 1 illustrates the sensor input temperature profile as perceived by the standby copy using hot, warm, and cold standby schemes, respectively. In effect, the sensor input temperature profile is viewed at different levels of detail, ranging from the most detailed one corresponding to the use of maximum sensor sampling frequency, to the least detailed one corresponding to the use of minimum sensor sampling frequency.

Manuscript received November 27, 1989; revised August 1993. This work was supported in part by the National Science Foundation under Grant CCR-9110816 and the US Nuclear Regulatory Commission under award NRC-04-92-090. Recommended for acceptance by J. Knight.

I.-R. Chen is with the Intitute of Information Engineering, National Cheng Kung University, Tainan, Taiwan. The work was conducted while he was with the Department of Computer and Information Science, University of Mississippi, University, MS 38677 USA; e-mail: irchen@lexus.cs.olemiss.edu.

F. B. Bastani is with the Department of Computer Science, University of Houston, Houston, TX 77204-3475 USA.

IEEE Log Number 9403566.

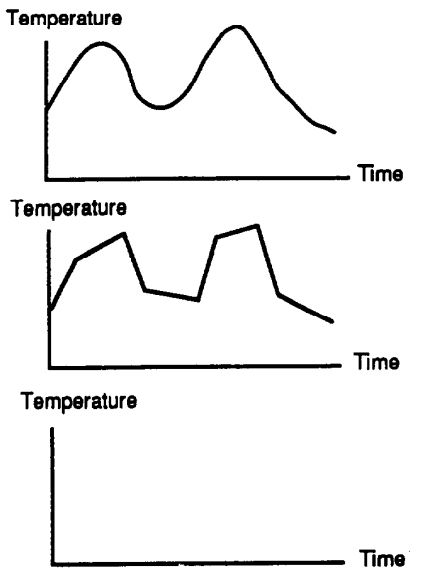

Fig. 1. Different detailed view of a temperature profile.

There are two implications in this example which must be pointed out. First, a warm standby copy, although possessing only a partial view of the sensor input temperature profile, still has a useful and summarized view of the temperature profile, e.g., it may still know what the maximum temperature is, when it was attained, etc. This allows a warm standby to immediately take charge using its summarized information without having to start from scratch as in the cold standby scheme. Second, the amount of processing power to create and maintain a standby copy is proportional to the sampling frequency and, hence, a warm standby copy will not consume as much processing power as a hot standby copy since the sampling rate is lower. This means that for the same hardware cost a higher degree of replication may be achieved by using warm standby instead of hot standby copies. This higher degree of replication for the same hardware cost can provide the system with a better reliability, particularly for long-lived unmaintainable systems, and those operating in harsh environments where burst hardware failures are possible, because now a process has more copies to tolerate multiple processor failures. A reliability analysis will be performed later in Section III to illustrate this point.

In the following, we give a more formal definition of a primary or standby copy of a control process, and its interaction with other control processes in a hierarchically structured system. The definitions are illustrated using the system shown in Fig. 2. It consists of four application processes, a, b, c, and d.

A. Seniority Function

A copy of a process may impose only a partial load on a processor depending on a design parameter called the seniority of that copy. Formally, let A denote the set of processes in the hierarchical control system (e.g., a, b, c, and d in Fig. 2) and let P denote the set of available processors. Then, let π: $A \rightarrow A \bigcup \{\emptyset\}$ be the parent function (e.g., $\pi(b) = \pi(c) = a$ in Fig. 2) for the hierarchical structure;
l: $A \rightarrow [0, \infty)$ be the load function (e.g., instr/sec) of processes in A;
c: $P \rightarrow [0, \infty)$ be the capacity (e.g., instr/sec) of processors in P.
The seniority function allocates copies of processes to processors,

$$\sigma\colon A \times P \rightarrow [0, 1].$$

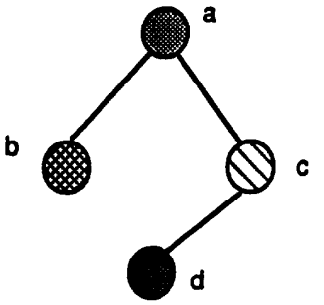

Fig. 2. An abstract hierarchy.

Thus, if a copy of a process a's seniority $\sigma(a, p)$ is 1, then it means that the copy is either a primary copy or a hot standby copy and it runs at its full load, $l(a)$, on processor p, whereas $\sigma(a, p) = 0$ indicates that processor p does not execute a at all. A value between 0 and 1 means that this copy of process a is a warm standby copy and imposes $\sigma(a, p)l(a)$ load on processor p. The allocation must satisfy the following constraint,

$$\forall p \in P : \sum_{a \in A} \sigma(a, p) l(a) \leq c(p).$$

B. Logical Communication Link

In a conventional hierarchical structure, for each parent/child process-pair, we have one parent-to-child logical communication link (for sending control instructions) and one child-to-parent logical communication link (for transmitting status information). In the hierarchical structure with warm standby copies, similar logical communication links are used. However, the logical communication links need not be of the same capacity. Formally, let

$$\text{allocation}(a) = \{x | \sigma(a, x) > 0\},$$
$$\text{primary}(a) = \{x | \forall y, \sigma(a, x) \geq \sigma(a, y)\}.$$

In other words, allocation(a) is the set of processors having at least one copy of a and primary(a) is the set of processors having the most senior copies of a. There are two sets of active logical communication links in the hierarchy.

- A parent-to-child link from x to y iff $\exists a \in A$ such that $x \in$ primary($\pi(a)$) and $y \in$ allocation(a). The capacity of the link is proportional to $\sigma(a, y)$.
- A child-to-parent link from x to y iff $\exists a \in A$ such that $x \in$ primary(a) and $y \in$ allocation($\pi(a)$). The capacity of the link is proportional to $\sigma(\pi(a), y)$.

Notice that the load on the communication subsystem is the same for both the warm standby and hot standby schemes since the source of all information is the set of primary nodes while the destination is the set of allocated nodes. If the receiver is a full copy (one that is allocated to a primary node) then it gets complete information, otherwise it receives only partial information.

III. Evaluation

We first show that the use of warm standby copies instead of just hot standby copies can enhance the system reliability. Design conditions under which the above statement is true are investigated. Then, we present a simulation evaluation of the warm standby scheme using a case study.

A. Reliability of Partial Replication

As pointed out in Section II, since a warm standby (i.e., a partial) copy requires less processing power than a full copy, more standby copies for the same hardware cost can be used to tolerate hardware failures, resulting in a system that is less vulnerable to hardware failure. A direct consequence of this effect is enhanced reliability. At the same time, there is no increase in software complexity since all copies of a process run the same program. Nevertheless, a design tradeoff associated with the use of warm standby copies over just hot standby copies is that there exists a possibility that a warm standby copy may not be able to deal with a control situation when it takes control. The period that is required for a partial copy to advance its seniority to become a full copy when it initially takes charge is called a vulnerable period, which depends on the copy's seniority. In the following, we present an analysis that illustrates conditions under which the warm standby scheme may be favored over the hot standby scheme.

Consider the case of allocating two processes, a and b, to four processors, p_1, p_2, p_3, and p_4, where $\pi(b) = a$ (a is the parent of b and thus both are important system functions and cannot fail at any time) and $c(p_i) = l(a) = l(b)$, $1 \le i \le 4$ (each processor has the processing capability of loading up to one full copy, either a or b). We assume that a processor functions for an exponentially distributed time with rate λ; once it fails it stays down because there is no repair capability in the system. Now, consider the following two ways of achieving fault tolerance by means of standby redundancy.

1) Using only hot standby copies, e.g., allocation(a) = $\{p_1, p_2\}$, allocation(b) = $\{p_3, p_4\}$, $\sigma(a,p_1) = \sigma(a,p_2) = 1$, and $\sigma(b,p_3) = \sigma(b,p_4) = 1$. It consists of a series structure of two subsystems with one consisting of p_1 and p_2, each containing a full copy of a, in a parallel structure, and the other consisting of p_3 and p_4, each containing a full copy of b, also in a parallel structure. The reliability of the system is given by

$$r(t)|_{\text{hot standby}} = 4e^{-2\lambda t} - 4e^{-3\lambda t} + e^{-4\lambda t}.$$

2) Using warm standby copies, e.g., allocation(a) = $\{p_1, p_3, p_4\}$, allocation(b) = $\{p_2, p_3, p_4\}$, $\sigma(a,p_1) = \sigma(b,p_2) = 1$, and $\sigma(a,p_3) = \sigma(b,p_3) = \sigma(a,p_4) = \sigma(b,p_4) = 0.5$. A seniority of 0.5 means that a copy runs only at its one-half load on the processor it is allocated to. The reliability of this system is bounded from above by the reliability of a 2-out-of-4 system. Let x_i be 1 if p_i is alive and let it be 0 if p_i has failed, for $i = 1,2,3,4$. Let $\bar{x}_i$ denote the complement of x_i. Then the structure function for the system [3] is $s(x_1,x_2,x_3,x_4) = x_1x_2x_3x_4 + x_1x_2x_3\bar{x}_4 + x_1x_2\bar{x}_3x_4 + x_1\bar{x}_2x_3x_4 + \bar{x}_1x_2x_3x_4 + x_1x_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1x_2x_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1x_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3x_4$. From this, the upper bound on the reliability of the warm standby system is given by:

$$r(t)|^{\text{upper bound}}_{\text{warm standby}} = 6e^{-2\lambda t} - 8e^{-3\lambda t} + 3e^{-4\lambda t},$$

which is better than the reliability of the hot standby system. However, the lower bound on the reliability is given by

$$r(t)|^{\text{lower bound}}_{\text{warm standby}} = e^{-2\lambda t}.$$

The reason the reliability is less than the upper bound is because of the probability of a faulty control decision while the warm standby is in the process of gathering sufficient information to become the primary controller after the primary fails. We have developed a detailed reliability model [5] that assumes that when a partial copy of process a or process b takes over, it takes an exponentially distributed time (this is the vulnerable period) with rate μ_a or μ_b, respectively, to become a primary copy. Moreover, it models the fact that during this vulnerable period, there is a software failure rate, θ, representing the rate at which a partial copy fails to deal with a control task when it takes over.

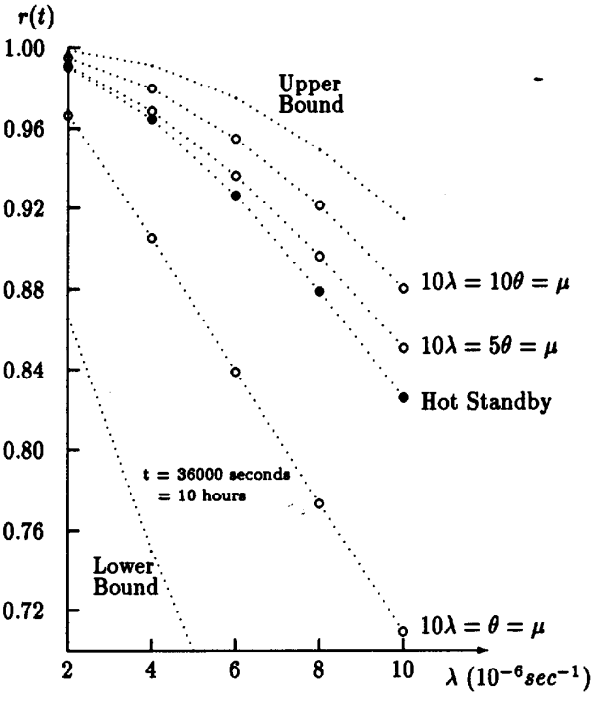

Fig. 3. Reliabilities of hot standby and warm standby with lower and upper bounds.

Detailed calculations [5] show that the reliability of the system using warm standby copies is better than that just using hot standby copies as θ (software failure rate of a partial copy) decreases and as μ (recovery rate of a partial copy) increases (here $\mu_a = \mu_b = \mu$). A general observation is that when $\theta \le \lambda$, the reliability of the warm standby system is always better than that of the hot standby system. Fig. 3 compares the reliability of these two systems with all parameters varying proportionately. When θ is comparable in magnitude to λ, the warm standby system can provide a better reliability than the hot standby system as the underlying hardware becomes more unreliable. An explanation of this is because state transitions that could lead to system failure in the warm standby system are mostly due to θ rather than λ and the probability that a state transition can lead to system failure in the warm standby system is less than that of the hot standby system since there are more states in the warm standby system. Consequently, the reliability of the warm standby system will decline by a lesser extent than that of the hot standby system as λ increases since increasing λ only increases θ by the same order of magnitude. Conversely, when θ is an order of magnitude higher than λ (e.g., $10\lambda = \theta = \mu$), the warm standby system will suffer more from increasing λ since this increases θ by an extra order of magnitude (i.e., 10 times) and the probability of state transitions that can lead to system failure for the warm standby system is greatly increased. In summary, we conclude that the warm

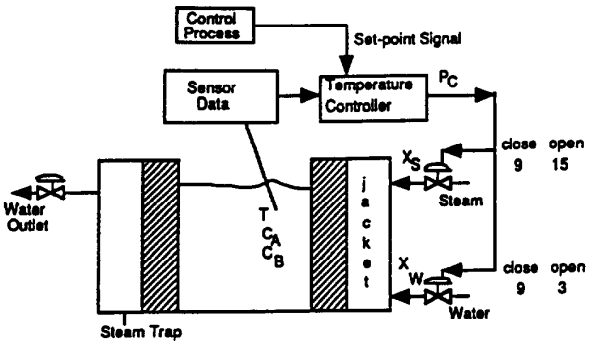

Fig. 4. A batch reactor.

standby scheme is most favorable when θ is of the same order of magnitude as λ and this favorable situation is most likely when the underlying hardware is unreliable and/or the recovery rate (μ) is high.

B. Simulation Evaluation

In this section, we first develop a process-control program for a simulated experimental chemical batch reactor system to illustrate the warm standby technique in practice. Then, we present the simulation results and analyze the effect of various parameters on the reliability of the recovery procedure. In this case study, the physical environment of the batch reactor in which the control processes are embedded is simulated; however, the control processes are completely implemented, instead of being simulated, and operate in real-time. The environment simulator sends sensor data in every Δt interval to the control processes; when it receives control actions in response to a sensor event (e.g., opening a fraction of a steam valve) from the control processes, the simulator updates the state of the environment (e.g., temperature and pressure) to that at $t + \Delta t$ based on the state at t, and advances its simulation clock to $t + \Delta t$.

A Chemical Batch Reactor System: Consider the batch reactor sketched in Fig. 4 where first-order consecutive reactions take place in the reactor as time proceeds. Reactant A (with a corresponding solution concentration C_A) is initially charged into the vessel. Steam is fed into the jacket to bring the reactor up to a temperature at which the consecutive reactions begin. Cooling water is later added to the jacket to remove the exothermic heat of reactions. The product that is desired is component B (with a solution concentration C_B). If the control process lets the reaction go on for too long, too much B will react to form compound C (with a solution concentration C_C) and consequently the yield of B will be low. On the other hand, if the control process stops the reaction too early, too little A will have reacted and the conversion and yield of B will again be low. Therefore, the control process needs to control the batch reaction to follow a specific temperature profile (i.e., time vs temperature profile) in order to optimize the yield. The actual temperature is adjusted by a controller which controls two split-range valves, a steam valve and a water valve. The fraction of the steam valve which is open, X_s, and the fraction of the water valve which is open, X_w, are determined by an output signal, P_c, produced by the temperature actuator. The steam valve is wide open when $P_c = 15$ and is closed when $P_c \leq 9$ while the water valve is closed when $P_c \geq 9$ and wide open when $P_c = 3$. Hence, the control process needs to communicate closely with the temperature controller to properly adjust the temperature in the reactor.

Fig. 5 shows an instance of the optimum temperature and concentration profiles with $T_{\max}$ representing the maximum temperature and

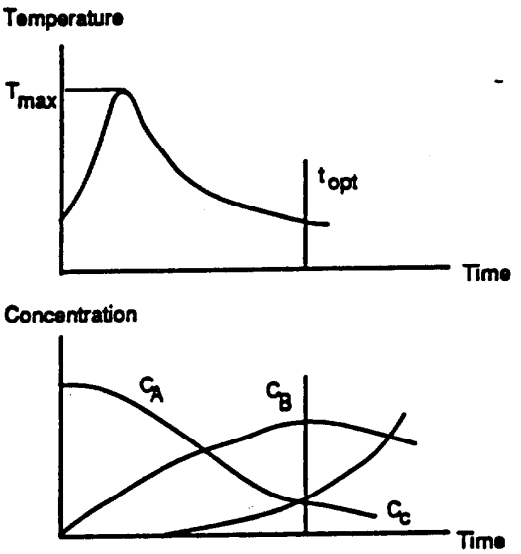

Fig. 5. Batch profiles.

C_j representing the concentration of A, B, or C. If the reaction runs longer than t_{opt}, the yield of B decreases. A complete set of equations that describe the kinetics of the first-order consecutive reactions can be found in [9].

A Hierarchically Structured Control Program: The control process described above can be implemented as a four-level control hierarchy (Fig. 6) as follows:

Level 4: This level controls the inventory of chemicals, the scheduling of batch reactions, the maintenance of production level, etc.

Level 3: This level governs the kinetics of different batch reactions (e.g. formulating optimal temperature profiles).

Level 2: This level consists of processes (i.e., master control processes) each of which controls a batch reaction using an optimal temperature profile provided by a Level 3 process. The responsibilities of a master control process include a) minimizing the mean square error (mse) between the actual and the desired reactor temperature profiles such that $\frac{C_{\text{B,desired}} - C_{\text{B,actual}}}{C_{\text{B,desired}}} \leq 3\%$ for the final concentration of product B, and (b) dynamically formulating desired jacket temperature profiles one segment at a time to be followed by a Level 1 process. The mse is defined as

$$\text{mse} = \frac{1}{t_{\text{batch}}} \sum_{t=1}^{t_{\text{batch}}} |T^2_{\text{desired}}(t) - T^2_{\text{actual}}(t)|$$

where t_{batch} is the total batch reaction time in minutes.

Level 1: This level consists of specialized processes (i.e., jacket temperature control processes) each of which is responsible for regulating jacket temperature changes (by controlling the steam and cooling water valves and flow rates) such that the prescribed jacket temperature profile formulated by a level 2 process is followed.

We focus our attention on a Level 2 process (the master control process) and a Level 1 process (the jacket temperature control process) of the control hierarchy. (In a hierarchical control structure as such, Levels 1 and 2 are normally made fault-tolerant because on-line repair is not practical for lower level real-time controllers.) We assume that a Level 3 process which formulates the desired temperature profile to be followed by the master control process is allocated to some processor in the system and there are four processors, p_1, p_2, p_3 and p_4, to which the master and the jacket processes can be allocated for controlling the batch reaction. Also, we

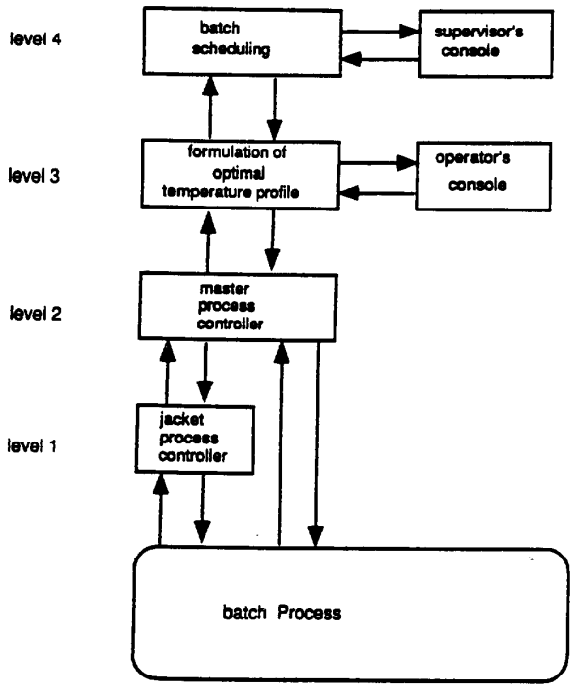

Fig. 6. A control hierarchy for the batch reactor system.

assume that a Level 2 or a Level 1 process will consume a fraction of the processing power of a processor in a ratio that is equivalent to its seniority. For example, if a copy's seniority is 1, then it will consume the full processing power of a processor that it is allocated to.

Seniority Function: A Level 2 master control process, m, is replicated on three processors, p_1, p_3, and p_4 with seniority functions $\sigma(m, p_1)$, $\sigma(m, p_3)$, and $\sigma(m, p_4)$, respectively. $\sigma(m, p_1)$ is always equal to 1.0 and $\sigma(m, p_3) > \sigma(m, p_4)$. Only the copy with $\sigma(m, p_1) = 1.0$ (the primary copy) provides direct control to the batch reactor with the others serving as partial copies. On the other hand, a level 1 jacket temperature control process, j, is replicated on three processors, p_2, p_3 and p_4, with seniority functions $\sigma(j, p_2)$, $\sigma(j, p_3)$, and $\sigma(j, p_4)$, respectively. $\sigma(j, p_2)$ is always equal to 1.0 and $\sigma(j, p_4) > \sigma(j, p_3)$. Again, only the copy with $\sigma(j, p_2) = 1.0$ (the primary copy) provides direct control to the jacket temperature. Recall that a copy consumes a fraction of the processing power of a processor in a ratio that is equivalent to its seniority, so that $\sigma(m, p_3) + \sigma(j, p_3) \leq 1$ and $\sigma(m, p_4) + \sigma(j, p_4) \leq 1$. Furthermore, when a junior copy becomes a primary copy, other partial copies residing in the same processor will be deprived of their processing power. For example, if $\sigma(m, p_3) = 0.6$, $\sigma(j, p_3) = 0.2$, and the junior copy with $\sigma(m, p_3) = 0.6$ advances its seniority to $\sigma(m, p_3) = 1.0$ due to detection of a failure of the primary copy of m, the junior copy with $\sigma(j, p_3) = 0.2$ will be deprived of its processing power from processor p_3. In our implementation, this is achieved by scheduling it to die at the same time when failure of the primary copy of m occurs.

Knowledge Representation: We use the same knowledge representation to describe the desired and actual reactor (or jacket) temperature profiles for all copies of m (or j). Fig. 7 shows the data structure used to describe a temperature profile. There are $60/t_s$ slots which could be filled per minute, where t_s represents the sensor sampling interval in seconds. The degree to which these slots are filled is proportional to a copy's seniority. For example, every nth slot is filled for a copy

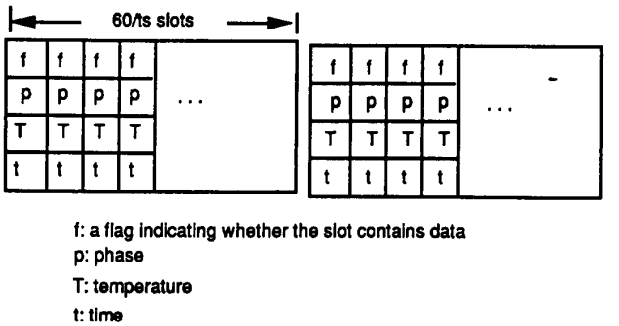

Fig. 7. The data structure for describing A temperature profile.

of a control process with a seniority equal to $1/n$. This data structure allows a temperature profile to be described at different levels of detail, i.e., as the number of filled slots increases, the temperature profile is known to a greater detail. Consequently, a copy with a low seniority will probably have more up-to-date information about less frequently updated information such as phase, rate of change of temperature (slope), etc., and have less up-to-date information about temperature slots. Note that with this implementation, a primary copy of m does not send different information to different copies of j and, hence, broadcast protocols with sampling by copies of j (with a sampling rate proportional to their seniorities) could be used.

Simulating the Batch Reactor Environment: The control environment in which the master and jacket processes are embedded is simulated by the following three processes running on other processors:

1) an environment process which simulates the physical environment of the batch reactor and the sensor subsystem;
2) a channel process which simulates the underlying communication subsystem with varying degrees of channel capacities simulated by changing its input queue lengths (this parameter is called C_{channel}); and
3) a supervisor control process which simulates a Level 3 process.

These three simulated processes, together with the three copies of the master control process and the three copies of the jacket temperature control process, communicate with one another through the channel process.

Fault Recovery Procedure: Process failure detection is implemented by "are you alive" and "I am alive" messages. Specifically, if a process does not respond to an "are you alive" message for more than $N_{\text{broadcast}}$ (a program parameter) consecutive broadcasting intervals (a broadcasting interval = a sensor sampling interval in our implementation), then the process is considered dead. The recovery action taken by a junior copy of m upon detection of a failure of its senior counterpart is a) advancing its seniority from either 0.2 to 0.6, or 0.6 to 1.0, and b) acquiring more detail control information from the supervisor control process (for the desired temperature profile) and the environment process (for the sensor data) in a frequency that is proportional to its new seniority. If the new seniority is 1.0, the junior copy takes charge immediately while gradually acquiring information from the environment. The recovery action taken by a junior copy of j is the same except that the parent process from which it acquires the desired jacket temperature profile is the primary copy of m.

Simulation Results: The parameters of our control program are shown in Table I. Table II compares the cases of single, double, and triple processor failures in terms of the mse (mean-square error) between the actual and optimal temperature profiles and the final

TABLE I
PARAMETERS OF BATCH REACTOR CONTROL PROGRAM

paramter	description
$\sigma(m,p_1)$	seniority of primary master process (always 1)
$\sigma(m,p_3)$	seniority of first junior master process
$\sigma(m,p_4)$	seniority of second junior master process
$\sigma(j,p_2)$	seniority of primary jacket process (always 1)
$\sigma(j,p_4)$	seniority of first junior jacket process
$\sigma(j,p_3)$	seniority of second junior jacket process
$p_{1,failure}$	failure time of processor p_1
$p_{2,failure}$	failure time of processor p_2
$p_{3,failure}$	failure time of processor p_3
$p_{4,failure}$	failure time of processor p_4
$N_{broadcast}$	no. of messages for detecting a process failure
$C_{channel}$	channel capacity of the communication subsystem

TABLE II
MSE AND C_B FOR SINGLE, DOUBLE AND TRIPLE PROCESSOR FAILURES

Processor Failures*	MSE	C_B $(mole/ft^3)$
none	0.674	0.285
p_1	0.972	0.285
p_2	0.813	0.285
(p_1, p_2)	0.940	0.285
(p_1, p_3)	1.051	0.285
(p_2, p_4)	0.921	0.285
(p_1, p_2, p_3)	1.239	0.285
(p_2, p_3, p_4)	529706.890	0.000

*based on $\sigma(m,p_3) = \sigma(j,p_4) = 0.6$, $\sigma(m,p_4) = \sigma(j,p_3) = 0.2$, $C_{channel} = 16$, $N_{broadcast} = 3$, and $t_{failure} = 14.0$.

yield of C_B. The time of processor failure is chosen to be at the most critical moment of the batch reaction, namely, at the time when a phase change occurs (at the 14th minute mark).

From Table II, we see that when all the copies of m or j fail (for example, all copies of j process fail when processors p_2, p_3, and p_4 all fail), the batch reaction goes on unattended and the final yield of C_B becomes quite low (in fact it is equal to zero) because too much B is consumed. On the other hand, for other cases (even for the case of a triple failure, e.g., (p_2, p_3, and p_4)) when one junior copy still survives, the yield of C_B is quite good. This is partly due to the fact that the batch reactor is intrinsically a set-point based reactor system and thus a temporary out-of-sync between the control and environment processes can be tolerated in a short period of time without causing catastrophic damages. However, it also points out the importance of using partial copies to provide fault tolerance – even a partial copy with seniority equal to 0.2 can make a significant difference. Our other experimental results [4] show that in all cases when at least one partial copy survives failures (for both the Level 1 and Level 2 processes), the yield of C_B is good and the mse is never more than $5°F^2$.

Comparison of the simulation results with the case of using full copies is obvious. In the latter case, p_1 and p_2 may be allocated for the master control process, and p_3 and p_4 may be allocated for the jacket control process. Hence, when a critical double failure occurs, namely, (p_1, p_2) or (p_3, p_4), the batch reaction will go on unattended. This is in contrast with the results for warm standby where we observe that when a double failure occurs, the batch reaction is always under proper control. Of particular interest is the simulation result as compared to the case of using *cold* standby processes. In the former case, there is no disruption of continuity of control when a warm standby process takes over whereas in the latter case a cold standby process would require a loading and restart period (e.g., to load the temperature history logged in some stable storage before restart) and during that period the system is left uncontrolled.

IV. SUMMARY

In this concise paper, we have developed a fault-tolerant technique that can be used in a variety of process-control systems. This technique provides good reliability in a cost-effective way by incorporating the concept of warm standby. Our case study shows that a surviving warm standby copy with a seniority as low as 0.2 can make a significant difference in providing continuity of control when failures occur. Our comparative study of the reliability of partial and hot standby techniques suggests that warm standby appears to have its greatest advantage when it is applied to systems whose underlying hardware is unreliable.

There are several research areas which include 1) developing a decentralized management methodology for hierarchically structured process-control programs with warm standby and analyzing the effects of local and global factors which influence the distribution of σ per process (influenced by *local* factors, e.g., importance of a process) and per processor (affected by *global* factors, e.g., load balancing requirements), 2) using a frame-structure-based knowledge representation technique to facilitate self-learning capability of control processes, (e.g., via peer-to-peer communications which can exist among copies of a process), and extending it to cases where the knowledge base represented by the frame structure is large, and 3) comparing the performance of warm standby using unreliable communication protocols (e.g., datagram services) with hot standby using reliable protocols based on timeout and retransmission.

ACKNOWLEDGMENT

The authors wish to thank the five anonymous reviewers for their detailed comments which have significantly improved the quality of this concise paper.

REFERENCES

[1] J. S. Albus, A. J. Barbera, and R. N. Nagel, "Theory and practice of hierarchical control," in *Proc. COMPSAC '81*, Washington, DC, Sept. 1981, pp. 18–39.
[2] A. Avižienis *et al.*, "The STAR (self-testing-and repairing) computer: An investigation into the theory and practice of fault tolerant computing," *IEEE Trans. Comput.*, vol. C-20, pp. 1312–1321, Nov. 1971.
[3] R. E. Barlow and F. Proschan, *Statistical Theory of Reliability and Life Testing*. New York: Holt, Rinehart and Winston, 1975.
[4] I. R. Chen, "An AI-Based architecture of self-stabilizing fault tolerant distributed process-control programs," Ph.D. Thesis, Dept. of Comput. Sci., Univ. of Houston, Dec., 1988.
[5] I. R. Chen and F. B. Bastani, "Reliability of fully and partially replicated systems," *IEEE Trans. Reliability*, vol. 41, no. 2, pp. 175–182, June 1992.
[6] R. Freiburghouse, "Making processing fail-safe," *Mini-Micro Syst.*, May, 1982.
[7] B. W. Johnson and P. M. Julish, "Fault-tolerant computer system for the A129 helicopter," *IEEE Trans. Aero. Elect. Syst.*, vol. 21, no. 2, pp. 220–229, 1985.
[8] B. W. Johnson, *Design and Analysis of Fault Tolerant Systems*. Reading, MA: Addison Wesley, 1989.
[9] W. L. Luyben, *Process Modeling, Simulation, and Control for Chemical Engineers*. New York: McGraw-Hill, 1973.
[10] M. Nicolaidis, "Evaluation of a self-checking version of the MC68000 microprocessor," in *15th Symp. Fault Tolerant Computing*, 1985.
[11] R. D. Schlichting and F. B. Schneider, "Fail-stop processors: An approach to designing fault-tolerant computing systems," *ACM Trans. Comput. Syst.*, vol. 1, no. 3, pp. 222–238, Aug., 1983.
[12] W. Toy, "Fault-tolerant design of local ESS processors," *Proc. IEEE*, vol. 66, Oct. 1978, pp. 1126–1145.
[13] T. J. Williams, "The development of reliability in industrial control systems" *IEEE Micro* vol. 4, no. 6, pp. 66–80, Dec., 1984.