On Real-Time Quasi-Durable Checkpointing

J. Huang, P.-J. Wan, V. Thomas
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418, USA
{huang,pwan,thomas}@htc.honeywell.com

Abstract

This study investigates real-time checkpointing techniques in the context of distributed process control applications where checkpointing and recovery operations must meet timing constraints, such as process deadline and plant state validity. We introduce the notion of quasi-durability, which allows one to make tradeoffs between storage device reliability and the process control and recovery timing constraints. Based on this notion, we study three protocols for real-time quasi-durable checkpointing and recovery. For each protocol, we analyze its recoverability and provide the sufficient and necessary conditions for a set of device to be feasible for checkpointing and recovery.

1. Overview

In distributed process control systems, control processes may fail due to a variety of reasons including platform failures, transient faults caused by the operating environment, and software defects. The control system must be able to recover from failures by either performing on-line recovery or "cold restart". The on-line recovery uses a temporal plant state checkpointed during the normal operations, whereas the cold restart has to restart the system from its cold state. An on-line recovery is preferable over a cold restart because the cold state may have a larger deviation from the current plant state than the temporal state. A large state deviation may result in a *system bump* where the control system over-reacts to the controlled environment and takes a longer time to converge to a stable state.

The implication in providing such an on-line failure recovery capability is that the control system must be able to maintain the temporal plant state up-to-date and, at the same time, to meet control application timing constraints, such as periods of control processes. In a process control environment, a plant state is a function of time. So is the state, called *internal state*, maintained by the system for the purpose of failure recovery. The internal state is valid for recovery only if it is sufficiently close to the current plant state. In addition, control processes are real-time, i.e., they must perform their sensing, calculation, and actuating operations within each time period of a specified execution rate. As application requirements, neither an out-of-date internal state nor a tardy process control operation is acceptable.

This paper presents an approach, called *real-time quasi-durable checkpointing*, to meet such requirements [1]. The checkpointing and recovery are said to be *real-time* if their operations must meet process control deadlines and the time-variant validity of plant state information. A checkpointing is said to be *quasi-durable* if the internal state information maintained by the system is vulnerable to storage failure. The real-time quasi-durable checkpointing approach considers tradeoffs between the timeliness of checkpointing and the durability of stored state information. In addition, since the internal plant state maintained for recovery is time-variant, decaying as time goes, there exists a tradeoff between the storage reliability and the validity of the internal state.

Based on these tradeoffs, we introduce a new concept, *recoverability*. It is defined as the expected validity of the internal state used for recovery in case of process failure. The recoverability depends on the set of devices used to maintain the plant state information, the checkpointing order and the recovery order of the set of devices.

In this study, we investigate three protocols for real-time quasi-durable checkpointing and recovery. We analyze the time bound and recoverability of the three protocols with respect to the timing constraints of checkpointing deadline and state validity interval. Our goal is to enable control system designers to determine (1) the worst case checkpointing and recovery times of their systems by using one of the three protocols best suitable to their system; (2) the recoverability of a control process by plugging in appropriate parameter values for the storage devices as well as the checkpointing and recovery timing constraints.

0-8186-7614-0/96 $5.00 © 1996 IEEE

Checkpointing techniques have been widely used in various application domains [1]. Improving the performance of checkpointing has always been a system design issue. For example, the idea of using main memory for checkpointing was investigated in the context of database systems. However, the timing constraint imposed on checkpointing was not an issue. There was a body of work on real-time data management, focusing on real-time concurrency control but not real-time recovery. The effects of checkpointing overhead and latency was studied recently without considering deterministic performance behavior. The uniqueness of our work is that it addresses the issues of time-constrained checkpointing and recovery for real-time process control applications.

2. System Model

Reference Architecture

To be able to accommodate different types of distributed control system environments [1], we consider a two-tier system model, as illustrated in Figure 2-1. To focus on checkpointing and recovery issues, the model mainly consists of storage devices and interconnecting communication links. The control processes are located in the lower tier. The lower tier also contains a number of memory storage devices. The upper tier contains a number of disk storage devices. The control process, the memory and disk storage devices can be arbitrarily distributed. The two tiers are connected by a vertical network link. We use D_vert to denote the length of the network in terms of data transmission time. The value of D_vert can also be arbitrary. If it becomes zero, then the reference architecture actually becomes a flat one.

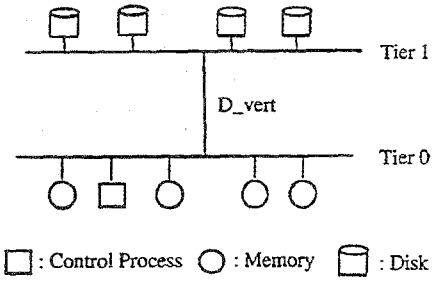

Figure 1. Reference Architecture

We assume that process communication between the devices is *asynchronous*, i.e., the sender of a message does not need to wait for an acknowledgment from the receiver.

Storage Device

A storage device can be a volatile memory or a SCSI disk. A number of storage devices exist across the control system. They are available to maintaining the plant state information for failure recovery. Each storage device has the following three parameters (r, χ, τ):

- r—Device reliability. The failure of a storage device occurs when the device is not available to the control system. We consider a fail-stop mode where a device stops functioning once it fails.
- χ—Device transmission time. It takes χ units of time for the storage device to access the network and inject the internal state message into the network.
- τ—Device access time. It takes τ_i units of time to access the storage device from the network.

The controller itself also contains a storage device, which is usually a volatile memory. The storage device in the controller also has the above three parameters.

State Validity

In a process control environment, a plant state is a function of time. For the purpose of failure recovery, each controller maintains its state information on storage device(s). This state information, called *internal state*, is valid for recovery only if it is sufficiently close to the current plant state. The real-time constraints for the internal state to be valid can be modeled by the following parameter.

- Y—the internal state decay interval, i.e., the units of time during which the internal state is valid for recovery use.

As a starting point, we consider a situation where the validity of the internal state decays linearly within a decay interval, as shown in Figure 2.

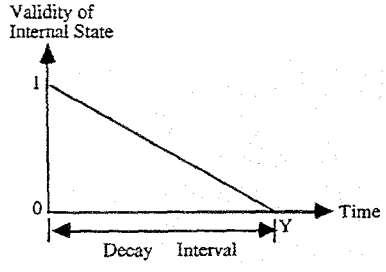

Figure 2. Plant State Validity Model

In particular, suppose that the internal state are refreshed at time 0, then at any time t, the validity is given by

$$v(t) = \begin{cases} 1 - \frac{t}{Y} & \text{if } 0 \le t \le Y, \\ 0 & \text{if } t \ge Y. \end{cases}$$

At any time, the internal state is said to be fresh if it has been refreshed within the current refresh period, otherwise it is said to be stale.

Real-Time Processes

We consider three kinds of real-time processes of the controller. One is the real-time control process that carries

out time-critical control system functions, such as advanced PID algorithms. This process must meet certain timing constraints. An example is the execution period of a PID algorithm. During each period, the control process performs sensing, PID calculation, and actuating. If these operations miss the period, the control system may malfunction.

Another kind of process is *checkpointing* which periodically refreshes internal states stored in storage devices to ensure the internal state is consistent with the plant state of the controlled system. We consider three constraints imposed on the checkpointing process:

- C—Period of the checkpointing process. It is also called *refresh period.*
- D—Deadline of the checkpointing process during each period. Missing a deadline may affect either the validity of the checkpointed internal state or the timing behavior of the control process.
- M—Maximum number of devices allowed to be used for checkpointing. Typically, M is small. We consider a range of 1 to 5. This consideration is due to two reasons: one is to avoid the communication congestion; the other is that each device has a high reliability.

The third kind of process is a *recovery* process, which is invoked in case of a control process failure. It is an operation that restores the control process using the internal state information stored in a storage device. The real-time constraint of the recovery process is that the internal state must be restored before it becomes obsolete.

Failure Model

We consider two types of failures—control process failure and storage device failure. The control process failure refers to transient faults where control processes temporarily malfunction due to, for example, execution exception and power outage. The storage device failure occurs when a device is not available to the control system. We consider a fail-stop mode where a device stops functioning once it fails. In this study, we do not consider communication failures, assuming the underlying communication protocols / subsystems provide fault-tolerance support.

Recoverability Model

For any checkpointing and recovery mechanism, a set of devices is *feasible* for checkpointing and recovery if

- the checkpointing time for the controller to refresh the internal states of the set of devices is less than D,
- whenever the control process fails, the internal state of any device in the set must remain valid when it is used by the controller for recovery.

Given a feasible storage device set, we define *recoverability* of a control process as the expected validity that a control plant can resume its operation from its valid internal state after a control process fails at time t.

Notation

b_{ij}—the propagation time of messages between the access points of device i and device j

$d1(i, j)$—the one-way distance from storage device i to storage device j, which is the sum of the time for storage device to access the network χ, the propagation time in the network between the storage device i and j, b_{ij}, and the time to access the storage device j from the network τ_j.

$d2(i, j)$— the two-way distance between storage device i and storage device j: $d2(i, j) = d1(i, j) + d1(j, i)$.

$d2(i)$—the two-way distance between the control process and the storage device i.

3. Real-Time Quasi-Durable Checkpointing Approach

Given the system model, we propose a real-time quasi-durable checkpointing approach for on-line recovery of process failure. It is real-time in the sense that its operations meet the timing constraints of control processes, checkpointing processes, and recovery processes. It is quasi-durable since it uses storage devices that are subject to a failure, i.e., the internal state information stored in the storage device is vulnerable to failure of the device. With this approach, we study three protocols for checkpointing and recovery and present the results of protocol timing and recoverability analysis. For details, the reader is referred to [1].

Star Protocol

We first consider a star protocol, where the control process communicates with individual storage devices in a point-to-point manner, as illustrated in Figure 3. Let S be a sequence of storage devices for checkpointing and recovery. For the checkpointing process, the controller first sends the plant state message to the first device in S. After transmitting the plant state message to the first device in S, it sends the plant state message to the second storage device S, and so on. After sending the plant state messages to all the devices, it finishes its one round of checkpointing operation.

The recovery process works as follows. In case of control process failure, the controller will send the recovery request to all devices in S one-by-one. If a storage device have a valid internal state, it will send its internal state back to the controller. The controller will use the first arrived internal state for recovery.

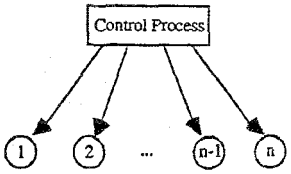

Figure 3. Star Protocol

Theorem 1 Let $S=\{1,2,\ldots,n\}$ be any sequence of storage devices. Then S is feasible for checkpointing and recovery in the star protocol if and only if the following conditions (1) and (2) are satisfied. Furthermore, if S is feasible, the worst recoverability for S is given by (3)

$$n\chi < D \tag{1}$$

$$C+(2i-1)\chi + d2(i) < Y, \quad \forall\, 1 \le i \le n \tag{2}$$

$$R^{\min} = \min\{R_i \mid 0 \le i \le n-1\} \tag{3}$$

where

$$R_i = \sum_{j=1}^{i-1} p_j\left(1-\frac{(i+j-1)\chi + d2(j)}{Y}\right) + \sum_{j=i}^{n} p_j\left(1-\frac{C+(i+j-1)\chi + d2(j)}{Y}\right)$$

$$p_i = r_i \prod_{j=0}^{i-1}(1-r_j) \tag{4}$$

Line Protocol

In the line protocol, the checkpointing process is performed as illustrated in Figure 4. The controller first sends the plant state to storage device 1 to refresh its internal state, and after the transmission of the plant state message, the controller will resume its normal process. The storage device 1 will then send its refreshed internal state message to the second storage. After it finishes the transmission of its internal state message, it will resume its normal state information. The second storage device will similarly refresh the internal state of storage device 3, and so on.

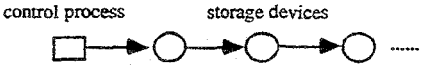

Figure 4. Line Protocol

The recovery process in the line protocol can be illustrated in Figure 5. When a control process failure happens, the control process will send the recovery request to the first device, if the first device has valid internal state, then it will send its internal state to the control process; otherwise it will forward the request to the second storage device. If the second storage device has valid internal state, then it will send its internal state to the control process directly; otherwise, it will forward this message to the third storage device, and so on.

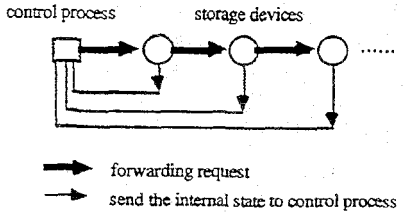

Figure 5 Recovery Process in the Line Protocol

Theorem 2 Let $S=\{1,2,\ldots,n\}$ be any sequence of storage devices. Then S is feasible for checkpointing and recovery in the line protocol if and only if both condition (5) and (6) are satisfied. Furthermore, if S is feasible, the worst recoverability for S is given by (7).

$$\chi < D \tag{5}$$

$$C+\chi+\sum_{j=1}^{i} d1(j-1,j)+d1(i,0) < Y, \ \forall\ 0 \le i \le n-1 \tag{6}$$

$$R^{\min} = \sum_{i=1}^{n} p_i\left(1-\frac{C+\chi+\sum_{j=0}^{i} d1(j-1,j)+d1(i,0)}{Y}\right) \tag{7}$$

where p_i is the same as in equation (4).

Multicasting Protocol

In the multicasting protocol, the communication capability between the controller and the storage devices is multicasting. In this protocol, the checkpointing is performed by the control process multicasting the plant state to all the storage device. For recovery process, when a control process failure occurs, the controller will multicast a recovery request and then wait for the internal state from any storage device. For each storage device, when it receives the recovery request from the controller, if its internal state is valid, it will send its internal state back to the controller. The controller will use the first arrived internal state for recovery.

Theorem 3 Let $S=\{1,2,\ldots,n\}$ be any set of storage devices. Then S is feasible for checkpointing and recovery in the line protocol if and only if both condition (5) and (8) are satisfied. Furthermore, if S is feasible, then the worst recoverability for S is given by (9).

$$C+\chi+d2(i) < Y \tag{8}$$

$$R^{\min} = \sum_{i=1}^{n} p_i\left(1-\frac{C+\chi+d2(i)}{Y}\right) \tag{9}$$

where p_i is the same as in equation (4).

References

[1] J. Huang, P.-J. Wan, and V. Thomas, "Quasi-Durable Checkpointing: A Real-Time Fault-Tolerance Approach," *Technical Report*, Honeywell Technology Center, July 1996.