MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-9592-PRE-5220

UM SISTEMA DE SUPERVISÃO TOLERANTE E FALHAS

Maria de Fátima Mattiello-Francisco

Trabalho apresentado no III Simpósio de Computadores Tolerantes a Falhas – PUC/RJ – Rio de Janeiro, Brasil – 20-22 setembro 1989.

INPE
São José dos Campos
2003

UM SISTEMA DE SUPERVISÃO TOLERANTE A FALHAS

M. F. MATTIELLO FRANCISCO*

SUMÁRIO

Este trabalho propõe uma solução para i corporar técnicas de tolerância a falhas a um sistema de supervisão, a fim de provê-lo de maior disponibilidade na ocorrência de falhas simples de "hardware". Uma arquitetura básica com redundância dupla é adotada. Em termos de "software" explora-se a recuperação retroativa com o estabelecimento dos pontos de recuperação a cargo dos aplicativos de Supervisão.

ABSTRACT

This paper proposes the incorporation of fault tolerant techniques in a Supervision System to provide it of high available when occur simple hardware's fault.A basic architecture with dual redundancy is used. With regard to the software, the backward-error-recovery is explored by establishing recovery points under the control of the supervision applicatives.

*Bacharel em Ciencia da Computação (ICMSC, USP/São Carlos 1980); Mestre em Eletrônica em Telecomunicação - Sistemas Digitais/Analógicos, INPE/1986); Sistema de Computação Tolerante a Falhas; Divisão de Software, Departamento de Eletrônica-DEL, Diretoria de Engenharia e Tecnologia Espacial-ETE, Instituto de Pesquisas Espaciais-INPE, Caixa Postal 515, São José dos Campos/SP.

1 - INTRODUÇÃO

Os Sistemas de Supervisão de modo geral apresentam a estrutura típica de um sistema de controle de processos de tempo real na qual são identificados três taref s básicas: aquisição, processamento e atuação.

Considerando que um dos requisitos fundamentais dos Sistemas de Supervisão é a alta disponibilidade, o sistema de computação utilizado para Supervisão deve possuir medidas de segurança de modo a evitar colapso no controle dos processos devido a falhas no próprio sistema de computação. Recentemente, o uso de mais de um processador para esta finalidade, em operação simultânea ou cooperativa, passou a ser frequente. Desta forma, a alta disponibilidade destes sistemas pôde ser obtida com a redundância de "hardware" adicionada aos mecanismos de "software" para tolerância a falhas, incorporados ao Sistema Operacional e utilizados pelos próprios aplicativos de supervisão.

Neste contexto, a partir de uma arquitetura básica caracterizada pela redundância dupla, na qual reside o Sistema Operacional iRMX86, o presente trabalho propõe uma solução para incorporar no iRMX86, mecanismos que viabilizem a recuperação retroativa.

2 - ARQUITETURA BÁSICA

A arquitetura básica adotada é composta de uma configuração de "hardware" acrescida da camada central do Sistema Operacional, aquela que se interfaceia com o "hardware" para apoiar a multiprogramação.

2.1 - hardware

Em termos de "hardware" utiliza-se uma arquitetura básica com redundância dupla (Figura 2.1) composta por duas Unidades de Processamento interligados entre si por dois barramentos paralelos idênticos de alta velocidade (com Acesso Direto à Memória).

Cada Unidade de Processamento possui:

- um processador (INTEL 86,88 16 bits)
- um controlador de barramento
- uma memória interna de leitura/escrita (RAM)
- um canal de Entrada/Saída (E/S)

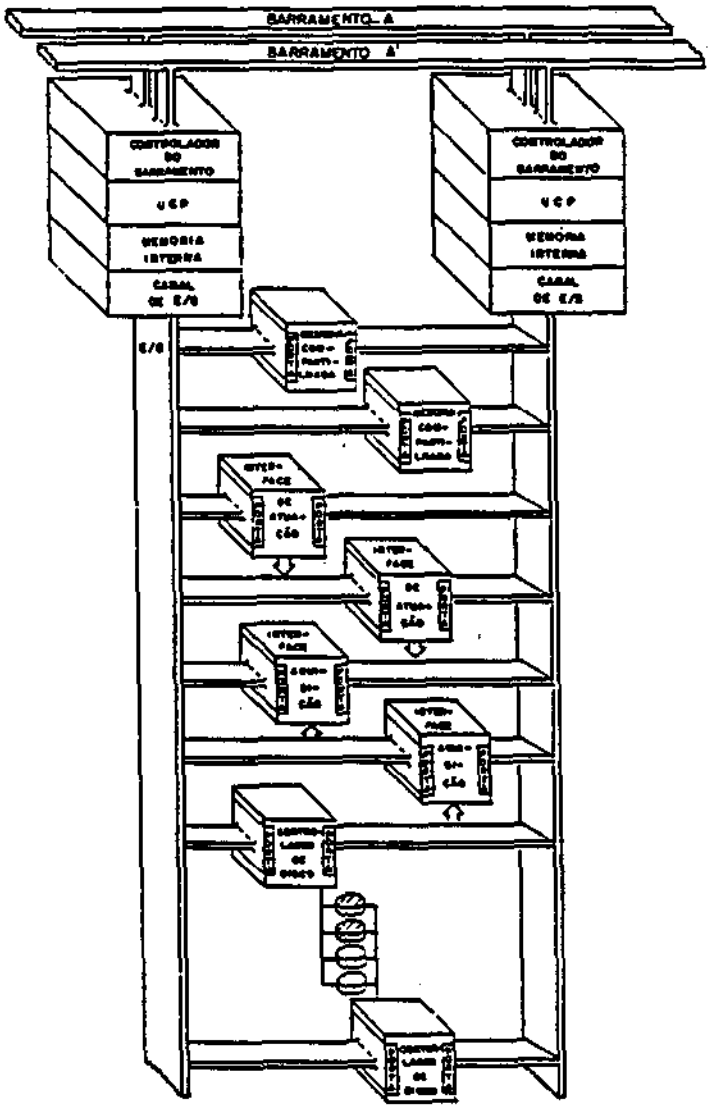

Fig. 2.1 - Arquitetura básica do Sistema de Supervisão

O canal de E/S da Unidade de Processamento é utilizado como via de acesso a recursos compartilhados do tipo portas duais. Um destes recursos consiste de uma memória compartilhada fisicamente constituída por dois módulos de memória de leitura/escrita (MEMBRAM) idênticas às memórias internas. Para tolerar falhas simples de memória compartilhada esta é duplicada.

O canal de E/S também é utilizado como via de acesso às interfaces de aquisição e interfaces de atuação cujos controladores também são do tipo portas duais e duplicados.

Esta arquitetura permite que controladores de disco e terminais de vídeo, do tipo portas duais, sejam configurados. No caso do disco, seus acionadores também devem ser do tipo portas duais; cada acionador conectado a dois controladores. Isto permite que os dados do disco permaneçam acessíveis mesmo na ocorrência de falha simultânea de um processador e um controlador.

Considera-se a existência de espelhamento automático de disco, isto é, cada par de discos possui exatamente as mesmas informações de modo que qualquer operação de escrita é executada simultâneamente nos dois discos.

Outro mecanismo disponível nesta arquitetura é o CÃO DE GUARDA, baseado em um temporizador que gera uma interrupçã na UNIDADE a qual se conecta, no término de um intervalo de tempo atingido nas seguintes situações:

1- a outra unidade parou
2- foi detectado algum tipo de defeito de "hardware" na outra unidade.

Em operação normal, quando não ocorrerem as situações 1 ou 2, o temporizador CÃO DE GUARDA deve ser reinicializado antes que se complete o intervalo de tempo em questão, evitando que seja gerada a interrupção (Mattiello, 1986)

2.2 - Sistema Operacional

Sobre o "hardware" apresentado opera um "software" básico, o Sistema Operacional iRMX86, que tem como funções principais gerenciar a utilização dos diversos recursos do sistema, viabilizar a execução dos programas de supervisão além de, permanecer sempre disponível para reconhecer e atender estímulos externos em tempo real.

O Sistema Operacional iRMS86 foi projetado para computadores INTEL iSBC 86,88 (processadores iaPX86,88) e caracateriza-se por ser multitarefas voltado para controle de processos em tempo real, fato que atende perfeitamente os requisitos da aplicação de supervisão.

A escolha deste pacote de "Software" deve-se à sua facilidade de expansão, e disponibilidade no mercado o que permite tornar um Sistema de Supervisão, tolerante a falhas, sem ter que desenvolver, especificamente, um Sistema Operacional tolerante a falhas.

A arquitetura básica do Sistema Operacional iRMX86, é representada na figura 2.2.

Como as partes que compõem o iRMX86 podem ser configuráveis, no ambito deste trabalho considera-se suficiente e também de interesse apenas a configuração do NÚCLEO. Esta restrição procura atender aplicações simples de supervisão sem inviabilizar aquelas mais complexas que fazem uso dos outros níveis do iRMX86 (INTEL, 1982 b)

É importante citar que a arquitetura do iRMX86 é orientada para objetos (tarefas, "jobs", segmentos, caixa postal, semáforos, regiões e outros) (INTEL, 1982 c).

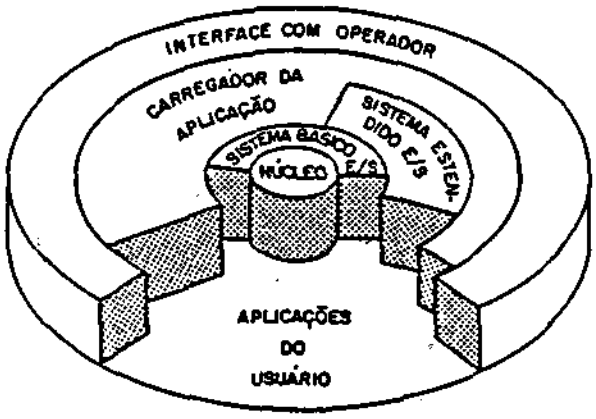

Fig. 2.2 - Arquitetura do Sistema Operacional iRMX86

Cada tipo de objeto definido pelo iRMX86 possui um conjunto de chamadas ao sistema (primitivas). Por exemplo, o objeto TAREFA possui as seguintes primitivas:

- CRIA bjeto
- DESCARTA objeto
- ENVIA Segmentos para outras tarefas
- RECEBE Segmentos de outras tarefas
- OBTÉM informação sobre objetos
- CATALOGA objetos com novos descritores
- DESCARTA objetos dos catálogos

3 - FILOSOFIA DE OPERAÇÃO

Utilizando a arquitetura básica apresentada, propõe-se que na operação de Supervisão uma Unidade de Processamento funcione no modo de operação ATIVO e a outra no modo de operação PASSIVO. A Unidade ATIVA deve executar todas as funções de supervisão enquanto que a Unidade PASSIVA deve funcionar simplesmente como reserva da ATIVA. Isto tolera falhas simples de "hardware" sem acarretar degradação no sistema.

Cada Unidade de Processamento conta com uma cópia do Sistema Operacional adotado (iRMX86).

Para funcionar como reserva, a Unidade PASSIVA tem conhecimento atualizado da situação dos processos executados pelo processamento da Unidade ATIVA. Isto é facilmente obtido da memória compartilhada (MEMÓRIA ESTÁVEL).

A falha de uma Unidade é sentida pela outra na ausência de um sinal de controle (tipo EU ESTOU BEM) enviado periodicamente uma Unidade para a outra, através de uma linha de controle do barramento que une as duas Unidades (ATIVA e PASSIVA).

Quando a ausência deste sinal for sentida pela Unidade PASSIVA, ela assume que a ATIVA falhou e desencadeia um procedimento para recuperação do sistema tornando-se ATIVA.

Caso a ausência deste sinal seja sentido pela ATIVA, nada deverá ocorrer em termos de reconfiguração do sistema de Supervisão.

A implementação desta lógica faz uso do temporizador CÃO DE GUARDA, ao nível de "hardware" e de manipuladores de exceção ao nível de "software".

Em operação normal, a reinicialização do temporizador CÃO DE GUARDA da Unidade PASSIVA, disparada pela Unidade ATIVA indica que esta última está bem. A geração deste sinal é feita pela próppria Unidade ATIVA através do atendimento de uma INTERRUPÇÃO DIAGNOSE que deve ocorrer em intervalos de tempo menores que o período associado ao CÃO DE GUARDA.

A rotina de atendimento à INTERRUPÇÃO DIAGNOSE, como o próprio nome sugere, executa testes diagnósticos na Unidade ATIVA, antes de disparar o sinal que reinicializa o CÃO DE GUARDA da PASSIVA. Caso os testes diagnósticos detectem erros na Unidade ATIVA, o sinal para a Unidade PASSIVA não é gerado. A ausência deste sinal implica na ocorrência da interrupção do CÃO DE GUARDA que, quando tratada pela Unidade PASSIVA faz com que esta torne-se ATIVA executando todo o chaveamento do sistema.

A operação de reinicialização do temporizador CÃO DE GUARDA conectado à Unidade ATIVA é análoga ao da PASSIVA, o que significa que esta última pode ficar executando AUTO DIAGNOSE (testes de consistência e testes de tempo) enquanto atua como reserva.

4 - TÉCNICAS ADOTADAS PARA TOLERÂNCIA A FALHAS

Para atender os objetivos do sistema de supervisão de forma a provê-lo da capacidade de tolerância a falhas, as seguintes técnicas, não dependentes da aplicação, foram selecionadas.

4.1 - Serviço "Hot-Standby" Simplificado

Dispondo da arquitetura e filosofia de operação apresentada, o sistema pode prover um serviço implementado por duas instâncias idênticas de cada tarefa, uma em cada unidade de processamento (ATIVA e PASSIVA). A alocação destas instâncias é feita na inicialização do sistema, de forma idêntica nas duas unidades de processamento (mesmo endereço absoluto).

Por serem os recursos: memória compartilhada e disco, do tipo portas duais, a Instância PASSIVA, pode funcionar exclusivamente como reserva, isto é, não é necessário que ela execute nem mesmo a função de manter uma cópia atualizada da situação da tarefa. O único requisito é que estas informações sejam armazenadas numa Memória Estável acessível pelas duas INSTÂNCIAS da tarefa (LOQUES, 1984a)

Assim sendo, a Unidade PASSIVA deve executar apenas o processamento necessário para o envio de sinalização "EU ESTOU BEM" para a Unidade ATIVA ("reset CÃO DE GUARDA).

No caso de falha da Unidade ATIVA, todas as INSTÂNCIAS PASSIVAS são tornadas ATIVAS e o processamento é continuado pela Unidade PASSIVA que a partir de então, torna-se ATIVA.

Caso a falha ocorra na Unidade PASSIVA, as Instâncias da Unidade ATIVA não são afetadas.

4.2 - Confinamento de Erro

As condições excepcionais consideradas são aquelas detectáveis pelo Sistema Operacional iRMX86 o qual caracteriza-se como **interface interpretativa** apoiando a detecção de erros pelos programas em execução nos casos de testes de consistência e testes de tempo.

Qualquer erro sinalizado pelo Sistema Operacional durante a execução da aplicação de Supervisão é manipulado como uma exceção (Anderson and Knight, 1983) que pode levar a uma parada do processador. Esta técnica evita que a Unidade ATIVA prossiga a execução errônea.

Apesar de altamente dependente da interface interpretativa, esta técnica não proíbe que asserções (Anderson and Lee, 1981) sejam introduzidas nas INSTÂNCIAS para detectar erros previstos da aplicação e possivelmente contorná-los.

Com esta técnica tem-se que: qualquer falha detectada pela INSTÂNCIA ATIVA de uma tarefa desvia o processador para um manipulador de exceção que parará completamente todas as atividades da Unidade ATIVA, inclusive a geração do sinal "EU ESTOU BEM" enviado para a Unidade PASSIVA . A ausência deste sinal faz com que a Unidade PASSIVA detecte tal situação e efetue a reconfiguração das Instâncias.

Em termos de confinamento de erro, uma alternativa mais abrangente do que a adotada seria considerar que um defeito pudesse afetar somente algumas Instâncias ATIVAS, não necessariamente todas executadas pela Unidade ATIVA. Neste caso, apenas as tarefas envolvidas com aquele tipo de defeito seriam substituídas pelas respectivas Instâncias PASSIVAS. Isto implicaria processamento distribuído nas 2 unidades, o que foge do escopo deste trabalho, visto que o iRMX86 não permite multiprocessamento.

4.3 - Técnica de Recuperação

A técnica de recuperação de erro adotada é a Recuperação Retroativa (Randell, et al, 1978). Ela depende da provisão de pontos de recuperação para registrar e preservar dados sobre a situação do sistema.

Com relação ao estabelecimento de pontos de recuperação, propõe-se considerá-los nos aplicativos do sistema de supervisão. Para tanto a interface interpretativa, o iRMX86, deverá prover o programador da aplicação de uma primitiva específica ("SAVE").

5 - EXTENSÃO DO iRMX86 PARA APOIAR TOLERÂNCIA A FALHAS

A forma escolhida para dotar a Supervisão de tolerância a falhas consiste em expandir o iRMX86 com recursos que possibilitam tornar as tarefas confiáveis e a recuperação do sistema transparente à aplicação. Para isto, propõe-se que os seguintes dois mecanismos de tolerância a falha sejam incorporados ao iRMX86:

- **Recurso para Tolerância a falhas:** esta camada constitui uma extensão do Núcleo do iRMX86 e implementa mecanismos de apoio à recuperação de erro (Fig. 5.1)

- **Gerente de Recuperação:** camada responsável pela recuperação transparente às atividades da aplicação, permitindo que o sistema seja recuperado de falhas simples sem que as atividades de supervisão tenham que ser reinicializadas.

O Gerente de Recuperação utiliza os serviços oferecidos pelo NÚCLEO do iRMX86 e as facilidades providas pelo Recurso Para Tolerância a Falhas.

Convém ressaltar que o Sistema Operacional iRMX86 por ser padrão, disponível no mercado, é fechado à modificação. Isto significa que não se tem acesso à sua estrutura de dados e código. Assim, os mecanismos de tolerância a falhas propostos utilizam apenas as formas permitidas pelo iRMX86 para serem a ele incorporados.

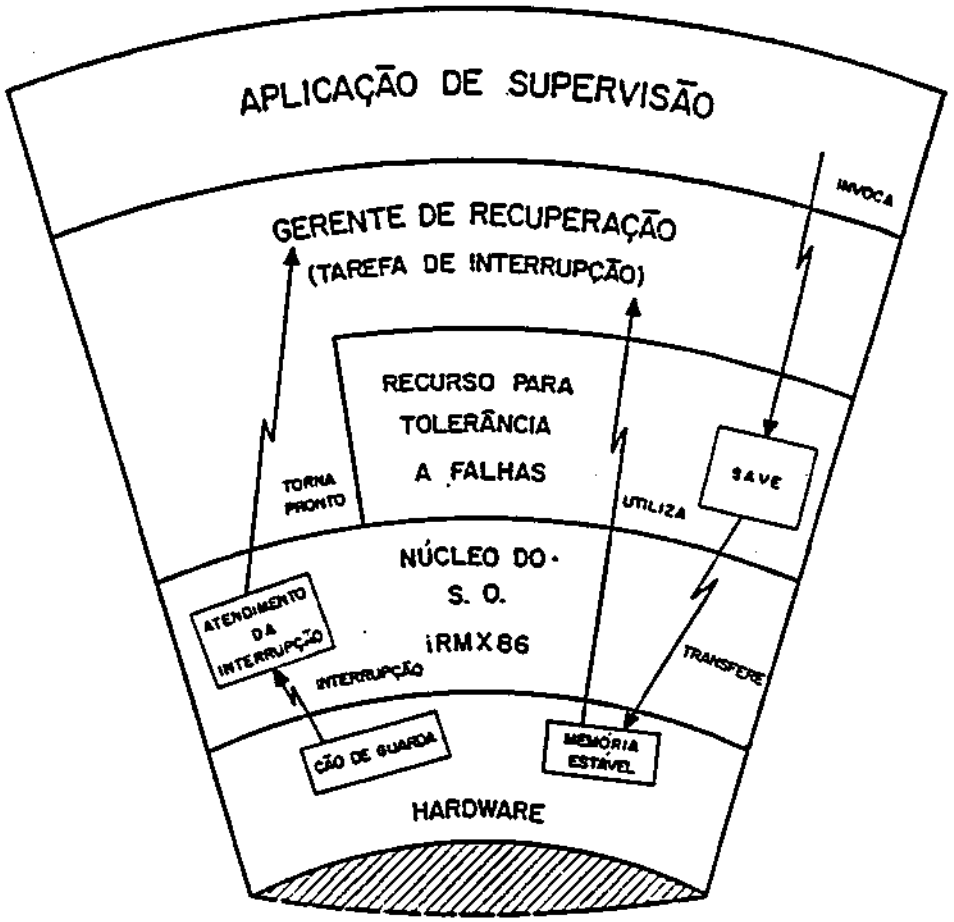

Fig. 5.1 - Sistema de Supervisão Multiníveis Tolerante a Falhas.

5.1 - Recurso para Tolerância a Falhas

Utiliza de forma simplificada o conceito de **memória estável** (Loques, 1984 c). O recurso memória compartilhada, disponível na arquitetura básica, deve manter as informações da situação de sistema atualizadas nos pontos de recuperação estabelecidos pela aplicação, através da invocação do SAVE, primitiva incorporada ao Sistema Operacional iRMX86.

Para implementação de memória estável duas alternativas são apresentadas:

ALT 1 - Transferência total do estado do sistema a cada operação "SAVE" - implementação da estratégia ponto de teste (checkpoint) - Fig. 5.2.

Nesta alternativa, em caso de falha, a memória estável deve conter no mínimo:

1- Toda a área de dados do NÚCLEO do iRMX86;

2- Toda a área de dados das INSTÂNCIAS ATIVAS. Estas áreas podem ser especificadas pela aplicação se a alocação das tarefas na memória for estática e feita pelo usuário na inicialização do sistema, através da criação das áreas e dos ambientes (JOBs) aos quais elas pertencem.

Os endereços absolutos das áreas a serem transferidas devem constar de uma TABELA DE TRANSFERÊNCIA criada automaticamente na inicialização do sistema, quando ocorre a alocação estática dos "jobs" e tarefas da aplicação.

Os dados desta Tabela são carregados com a invocação de 2 novas primitivas, em substituição às já providas pelo iRMX86, incorporadas como mecanismos de apoio a tolerância a falhas são elas:

CRIAJOB (extensão da primitiva CREATE$JOB provida pelo iRMX86);

CRIATAREFA (extensão da primitiva CREATE$TASK provida pelo iRMX86).

Após a inicialização do sistema, a tabela de transferência não deve ser alterada.

O conteúdo dos endereços especificados nesta tabela são copiados da memória interna da Unidade ATIVA para a MEMÓRIA ESTÁVEL, a cada ponto de recuperação, (invocação da primitiva "SAVE") definido pela Instância ATIVA em execução.

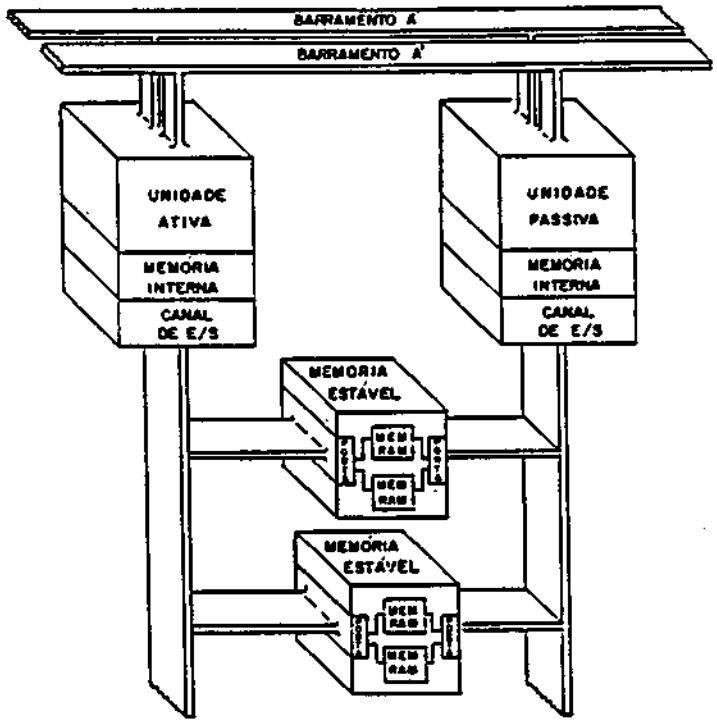

Fig. 5.2 - Memória Estável - ALTERNATIVA 1

O "SAVE" é adicionado ao iRMX86 como uma nova chamada ao sistema. Assim, a cada invocação desta primitiva (ponto de recuperação) não só a área de dados da Instância ATIVA que a invocou será transferida para a Memória Estável, mas também a área de dados de todas as outras Instâncias da Unidade ATIVA (SAVE VIRTUAL)

ALT 2 - transferencia parcial do estado do sistema na operação "SAVE" - implementação da estratégia "recovery cache" (Anderson and Lee, 1981).

Nesta alternativa adiciona-se uma pilha à memória compartilhada proposta em ALT 1 (Figura 5.3). Esta pilha permite armazenar os endereços de memória modificados, o que viabiliza transferir apenas as modificações feitas depois do "SAVE" anterior.

A ALT 2 não utiliza a Tabela de Transferência proposta em ALT 1. Portanto, nada impede que as tarefas da aplicação sejam alocadas dinâmicamente, já que este tipo de alocação é provido pelo iRMX86.

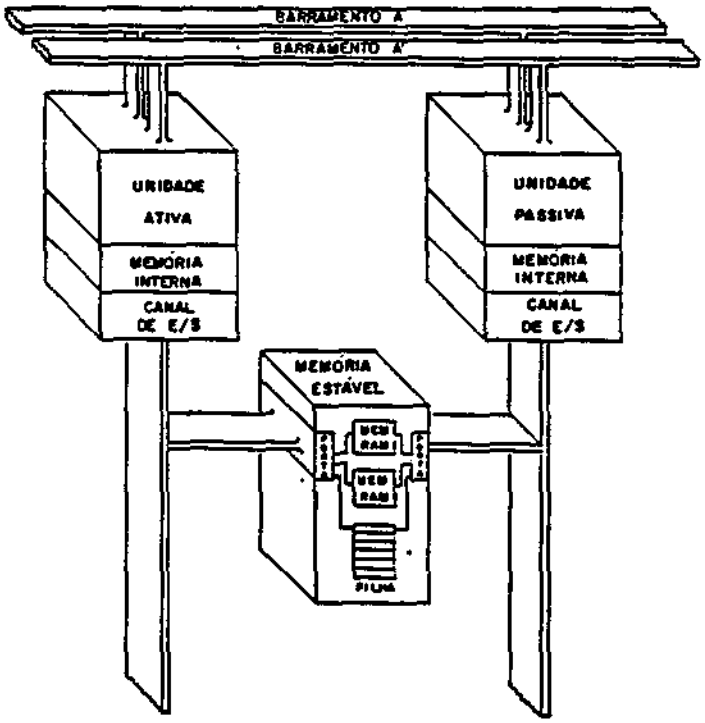

Fig. 5.3 -Memória Estável - ALTERNATIVA 2

5.2 - Gerente de Recuperação

Esta camada, incorporada ao Sistema Operacional iRMX86, tem por objetivo prover a recuperação automática de falhas de forma transparente à aplicação, isto é, sem qualquer interrupção aparente do processamento de supervisão.

O Gerente de Recuperação é uma tarefa que reside em ambas as Unidades ATIVA e PASSIVA. Caracteriza-se como uma tarefa de interrupção ativada pela interface interpretativa (iRMX86) por ocasião da manipulação da exceção correspondente ao CÃO DE GUARDA.

Considerando que a função do Gerente de Recuperação é efetuar, o mais rápido possível, a recuperação do sistema, deverá ser associada a esta tarefa alta prioridade. Isto é possível dada a característica do iRMX86: preemptivo por prioridade. Assim, garante-se que esta tarefa de interrupção quando tornada PRONTA receberá imediatamente o controle do processador, permanecendo neste estado (EM EXECUÇÃO) até que termine a recuperação.

5.3 - Aspectos de Implementação dos Mecanismos Propostos

As camadas propostas para extensão do iRMX86 preservam as características multiníveis apresentadas na figura 2.2.

Como pode ser observado na figura 5.1, o Sistema de Supervisão foi estruturado em níveis de abstração (estruturação vertical). A interação entre os níveis é implementada pelas chamadas ao sistema providas pelo iRMX86 acrescidas das novas primitivas propostas. Estas chamadas ao sistema funcionam como interpretadores entre níveis. Nota-se por exemplo, a primitiva "SAVE". Ela é invocada pela aplicação, porém é um mecanismo implementado pelo RECURSO PARA TOLERANCIA A FALHAS, que por sua vez utiliza a memória estável, provida pelo nível de "hardware" para armazenar o estado do sistema.

Na ALT 1 devem ser implementadas, ainda no nível RECURSO PARA TOLERANCIA A FALHAS, as primitivas CRIAJOB e CRIATAREFA que permitem a geração da Tabela de Transferencia.

O nível do GERENTE DE RECUPERAÇÃO é essencialmente implementado por uma Tarefa de Interrupção tornada PRONTA por ocasião da ocorrência da interrupção do CÃO DE GUARDA, através do atendimento desta interrupção, executado pelo NÚCLEO (interface interpretativa).

A utilização da Tabela de Transferência (nível RECURSO PARA TOLERANCIA A FALHAS) pelo nível superior (GERENTE DE RECUPERAÇÃO) em ALT 1, também preserva a característica de sistema multinível.

6 - EFICIENCIA E CONSISTÊNCIA DA RECUPERAÇÃO

A facilidade dos pontos de recuperação serem definidos pelo projetista do Sistema de Supervisão, apesar de bastante adequada para sistemas do tipo controle de processos, pode acarretar muitas interrupções no processamento de supervisão. O tempo gasto para transferência das informações de estado do sistema, mesmo que apenas sejam transferidas as modificações, não deve ser significativo a ponto de comprometer o desempenho de tempo real da supervisão.

As duas alternativas apresentadas no item 5 permitem a recuperação do sistema em caso de falha simples de "hardware". Entretanto, diferenças fundamentais em termos de eficiência podem ser constatadas.

O fato de o iRMX86 ser fechado a aplicação exige que a cada "SAVE" dado por uma Instância ATIVA, um "SAVE VIRTUAL" nas demais Instâncias da Unidade ATIVA seja executado. Isto implica uma transferência de dados bastante significativa a cada "SAVE", proporcional ao número de tarefas executadas pelo sistema. Com isto, a invocação de um "SAVE" por uma Instância ATIVA deixa de ser independente da invocação de um "SAVE" por outra Instância ATIVA. A frequência com que esta função é invocada pelas tarefas é um parâmetro crítico, pois quanto maior a ocorrência do "SAVE" num intervalo de tempo, menor o tempo restante deste intervalo para a execução das atividades de supervisão. Este fato já seria crítico se o "SAVE" dado por uma tarefa transferisse apenas as informações de estado daquela tarefa, quanto mais no caso do "SAVE VIRTUAL".

Neste contexto torna-se interessante a adoção da ALT 2 pois, apesar de a restrição do "SAVE VIRTUAL" também se aplicar a ela, executa-se apenas a transferência das áreas de memória que foram modificadas depois do último "SAVE", o que naturalmente significa bem menos transferências que em ALT 1. Além disso, a frequência com que o "SAVE" é invocada pelas Instâncias ATIVAS pode ser considerado um parâmetro menos crítico, já que quanto mais "SAVE" menos modificações efetuadas e, portanto, menor o número de transferências executadas. Com isto pode-se afirmar a não dependência entre as tarefas na invocação do "SAVE".

Considerando ainda, que o projetista de um sistema de supervisão tem total conhecimento da aplicação, torna-se possível e muito conveniente explorar este fato no sentido de otimizar o uso dos pontos de recuperação.

Em sistemas de supervisão a maioria das tarefas tem suas execuções ciclicamente repetidas. Assim, em geral, estas tarefas da aplicação tem uma frequência natural de repetição determinada por exemplo por um dispositivo supervisionado.

Dada esta característica, a otimização no uso dos pontos de recuperação pode ser feita aplicando o conceito de ciclos de controle (Martins e De Paula, 1984). Neste caso, o projetista do siatema de supervisão estrutura a execução das tarefas em ciclos de controle e determina o período do ciclo de controle do sistema de supervisão. Então, aplica a seguinte regra:

REGRA I : Invocar "SAVE" no início de cada ciclo de controle do sistema.

Duas outras preocupações relevantes são a repetitividade (preservação) dos dados a serem adquiridos da Interface de Aquisição e, a repetição da saída pela Interface de Atuação, no caso de falha do siatema e possível reexecução de tais atividades. O conceito de comunicação confiável (Mattiello, 1986) deve ser aplicado através das seguintes regras:

REGRA II : Invocar "SAVE" imediatamente após uma aquisição de dados da Interface de Aquisição.

REGRA III: Invocar "SAVE" imediatamente após uma saída de dados pela Interface de Atuação.

Quanto à repetição da entrada, preservação dos dados adquiridos, a regra II garante que no caso de falha, se a falha ocorreu depois da aquisição ter sido realizada, a sua recuperação estará consistente com os mesmos dados de entrada.

Quanto à geração de saída, o problema da consistencia pode ser resolvido com a implementação de ASSERÇÕES aliadas à aplicação da regra III. As asserções visam garantir a consistência dos dados resultantes de algum processamento crítico, evitando que certas ATUAÇÕES inconsistentes sejam efetivadas. A regra III impede a repetição indesejável de certas saídas.

7 - CONCLUSÃO

A proposta apresentada é uma forma viável e consistente para recuperar de maneira retroativa um Sistema de Supervisão, no caso de falhas simples de "hardware", sem ter que modificar seu "software" básico.

Apesar desta proposta não ter sido implementada, a análise do iRMX86 mostrou que é possível incorporar a este Sistema Operacional apenas duas novas funções (RECURSO PARA TOLERANCIA A FALHAS e GERENTE DE RECUPERAÇÃO) para torná-lo tolerante a falhas. E, esta extensão do iRMX86 nem mesmo exige a definição de novos objetos. A incorporação de uma única chamada ao sistema ("SAVE") é suficiente para prover a aplicação de supervisão da recuperação retroativa através do estabelecimento de pontos de recuperação.

Cuidado deve ser tomado pelo projetista da aplicação de supervisão para não abusar dos pontos de recuperação de modo a não comprometer o "tempo real". Contudo, o Sistema de Supervisão Tolerante a Falha proposto não está livre das limitações intrínsecas à técnica de recuperação retroativa adotada.

8 - REFERÊNCIAS

ANDERSON, T.; KNIGHT, J.C. A framework for software fault tolerance in real time systems. IEEE Transactions on Software Engineering, SE-9(3): 355-364, May, 1983.

ANDERSON, T.; LEE, P.A. Fault tolerance principles and practice. Londres Prentice/Hall, 1981.

INTEL. Introduction to the iRMX86 operating system. Santa Clara, CA., Intel, 1982a. Manual de referência do sistema.

-.iRMX86 nucleus reference manual. Santa Clara, CA., Intel, 1982b. Manual de referência do sistema.

-.iRMX86 operating system. Santa Clara, CA., Intel, 1982c. Manual de referência do sistema.

LOQUES FILHO, O.G. A fault tolerant distributed computer control system. Doctor of Philosophy. London, University of London. Department of computing, Feb., 1984a.

-. Uma técnica para a construção de software distribuido tolerante a falhas. In: SIMPÓSIO DE SOFTWARE BÁSICO, 4., ITA São José dos Campos, 29-31, out. 1984. Anais, São José dos Campos, SBC, 1984b.p. 166-172.

-. Uma arquitetura flexivel para sistemas distribuidos tolerantes a falhas. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 5.; CONFERÊNCIA LATINO-AMERICANA DE INFORMÁTICA, 11., Porto Alegre, SBC, 1985, p. 436-446.

MARTINS, R.C.O.; DE PAULA A.R. A fault tolerant 16 bits multiprocessing unit for on-board satellite applications. Sta. Anna-de-Bellevue, Canadá, Spar Aerospace, Apr., 1984.(SPAR TR = RML-009-84-57).

MATTIELLO, M.F. Um Sistema de Supervisão Tolerante a Falhas, Dissertação de Mestrado, INPE-4419-TDL/307, Junho, 1986.

RANDELL, B.; LEE, P.A.; TRELEAVEN, P.C. Reliability issues in computing system design. Computing Surveys, 10(2): 123-165, June 1978.

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

AUTORIZAÇÃO PARA PUBLICAÇÃO

Número
EEI-157/2002

Título

Um Sistema de Supervisão Tolerante a Falhas

Autor

Maria de Fátima MATTIELLO-FRANCISCO

Tradutor

Editor

INPE-5592-PRE/5220

Origem	Projeto	Série	No. de Páginas	No. de Fotos	No. de Mapas
ETE/DSS			16		

Tipo

☐ RPQ ☒ PRE ☐ NTC ☐ PRP ☐ MAN ☐ PUD ☐ TAE

Divulgação

☒ Externa ☐ Interna ☐ Reservada ☐ Lista de Distribuição Anexa

Periódico / Evento

III Simpósio de Computadores Tolerantes a Falhas / 20 a 22 Setembro 1989
SVC/RJ, Brasil

Convênio

Autorização Preliminar

__/__/__
Data

Edenilse F.E. Orlandi
Titular da Unidade
Chefe da Divisão de
Desenv. de Sistemas de Solo

Revisão Técnica

☐ Solicitada ☒ Dispensada

Recebida __/__/__ Devolvida __/__/__

Leonardo Fernando Perondi
Titular de Nível "A"
Engenharia e Tecnologia Espacial

Assinatura do Revisor

Revisão de Linguagem

☐ Solicitada ☒ Dispensada

Recebida __/__/__ Devolvida __/__/__

Titular de Nível "A"

Assinatura do Revisor

Autorização Final

__/__/__
Data

Edenilse F.E. Orlandi
Titular da Unidade
Chefe da Divisão de
Desenv. de Sistemas de Solo

Palavras Chave

tolerância a falhas - redundância - controle de processos - tempo real - software

(continua na folha 2/2)

Secretaria

__/__/__ Recebida __/__/__ Devolvida __/__/__

Data

_____ _____

Encaminhado Por Devolvido Por

Controle e Divulgação

__/__/__ Recebido Por: _____ Devolvido Para: _____

Data

Pronto Para Publicação em: __/__/__ ___/___/___

Data

No. _____ Quant._____ _____

Assinatura

Observações