Arquiteturas tolerantes a falhas

Taisy Silva Weber[1]

1 Arquiteturas de Sistemas Tolerantes a Falhas

É interessante que um sistema de computação seja suprido das técnicas de tolerância a falhas adequadas para garantir a confiabilidade desejada, sem que as aplicações tomem conhecimento das técnicas empregadas. A tolerância a falhas de um sistema de computação deveria idealmente estar suportada pelos níveis de hardware e software de abstração menor do que o nível ocupado pelas aplicações. Naturalmente essa característica não é suprida por todos os sistemas. Geralmente um especialista deve se ocupar do planejamento de certas tarefas complementares, como por exemplo estabelecimento de pontos de recuperação ou elaboração de rotinas diversitárias.

Um nível adequado para suprir tolerância a falhas é a arquitetura do sistema de computação. A arquitetura representa os componentes de hardware de um sistema (como processadores, memórias, controladores, interfaces) e suas interconexões (como barramentos e linhas seriais e paralelas de comunicação).

Recursos de tolerância a falhas implementados na arquitetura para detecção de erros, diagnóstico, recuperação e reconfiguração são mais eficazes do que os implementados exclusivamente nos níveis de aplicação e de sistema operacional sem o suporte dos níveis inferiores. A seguir são mostrados, como ilustração, exemplos de arquiteturas para sistemas tolerantes a falhas.

2 Tolerância a falhas em microprocessadores

Microprocessadores como os da família Intel x86 formam a base de computadores pessoais, estações de trabalho e servidores de rede. Como foram desenvolvidos inicialmente aplicações não críticas, os microprocessadores mais populares só recentemente começaram a apresentar alguns mecanismos intrínsecos para o suporte de técnicas de tolerância a falhas.

Devido principalmente ao aumento de desempenho e capacidade que os microprocessadores vêm apresentando, cresce o número de aplicações nas áreas de controle de processos industriais, controle de tráfego e instrumentação onde um certo grau de tolerância a falhas seria desejável. Naturalmente, desenvolvendo hardware adicional como votadores e comparadores, microprocessadores convencionais, mesmo sem recursos para suporte a tolerância a falhas, podem ser aproveitados para construir sistemas tolerantes a falhas.

---

[1] Professora orientadora do PPGC, UFRGS, UFRGS, Diretora Administrativa da Sociedade Brasileira de Computação

Uma melhor solução em termos de custo e tolerância a falhas pode ser alcançada, entretanto, se suporte para detecção e recuperação forem supridas diretamente pelo microprocessador, sem necessidade de hardware adicional. Um bom exemplo de microprocessador com essa característica é o iAPX432 da Intel [John84]. O suporte a tolerância a falhas implementado por esse processador não está relacionado às suas características específicas (como conjunto de instruções, modos de endereçamento e registradores internos) e pode ser implementado em qualquer sistema digital integrado com apenas um pequeno acréscimo na área de silício ocupada. Atualmente todos microprocessadores Pentium apresentam recursos derivados dessa primeira experiência da Intel.

2.1 Tolerância a Falhas no iAPX432

Basicamente um chip da família iAPX432 pode ser configurado como mestre ou verificador (figura 1). Um mestre pode operar sozinho ou ligado a um verificador. Um verificador sempre deve estar ligado a um mestre.

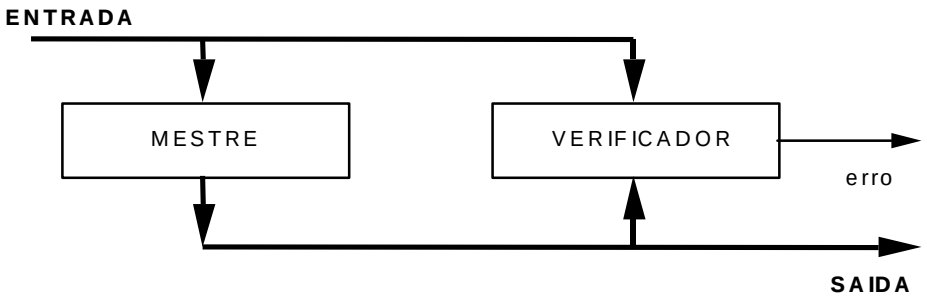

Figura 1 - Configuração mestre-verificador no iAPX432

Um chip configurado como verificador tem seus pinos de saída revertidos para entrada. Os pinos revertidos recebem sinais diretamente das saídas correspondentes do chip mestre. As entradas originais dos dois chips, mestre e verificador, estão ligadas em curto circuito. Internamente ao verificador, os sinais gerados como saída são comparados aos sinais recebidos pelos pinos revertidos. Ocorrendo qualquer discrepância na comparação, o verificador sinaliza erro.

O par mestre-verificador permite facilmente detecção de erro. Com apenas essa redundância simples, o tratamento do erro não é possível por hardware. Usando uma maior redundância, por exemplo quatro chips, ligados dois a dois, tanto detecção quando reconfiguração são possíveis. Essa arquitetura quadruplicada apresenta um par mestre-verificador primário e outro par estepe. Os dois pares são ativos, mas apenas o par primário fornece resultados ao sistema. Quando o verificador do par ativo detecta um erro, chaveia-se para o par estepe, que passa a partir desse momento a operar sozinho. Redundância quádrupla degrada, nesse caso, para duplex, mas o funcionamento do sistema é garantido sem queda de desempenho.

A redundância usada no iAPX432 é independente da função específica realizada pelo chip e pode ser implementada em qualquer circuito digital. O chip resultante tem sua área em silício aumentada em função do comparador, das chaves bidirecionais para todos os pinos de saída, do sinal de controle adicional necessário para configurar o chip como mestre ou verificador e do sinal de erro gerado no comparador (figura 2). Todos

os circuitos construídos usando essa técnica devem ser usados aos pares para detecção de erros.

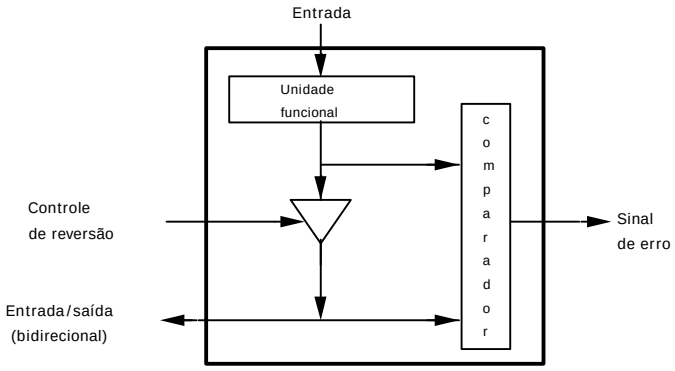

Figura 2- Unidade básica para redundância em microprocessadores

2.2 Tolerância a falhas no Pentium

No microprocessador Intel 486 foi usada verificação de paridade para as transferências internas de dados entre caches e unidades de execução. Já no Pentium, adicionalmente à paridade nas caches, a TLB e a memória de microcódigo também são verificadas quando à paridade. No Pentium foram introduzidos recursos adicionais para verificação de exceções suportada por hardware (*machine check exception*). Também o esquema de mestre / verificador (i432) com dois chips voltou a ser adotado pela Intel a partir do Pentium.

O Pentium Pro mantém todas as técnicas do Pentium e adicionalmente:

- paridade nos bytes de dados substituída por 8 bits de ECC
- 2 bits de paridade para barramento de endereço associado a técnicas de *retry*
- bits de paridade para sinais de controle

No Pentium Pro verificação de exceções é conduzida pela MCA (*machine check architecture)* que adiciona 3 registradores de controle e 5 bancos de 4 registradores de erro aos recursos do microprocessador.

3 Tolerância a falhas em sistemas de grande porte

Sistemas de grande porte para aplicações universais (*mainframes*) apresentam arquitetura composta de várias unidades processadoras de alto desempenho, uma memória comum de grande capacidade, além de canais que permitem a ligação a uma grande quantidade e variedade de periféricos. Apesar das aplicações de mainframes não serem geralmente críticas, os requisitos de disponibilidade impostos a esses sistemas tem crescido fortemente nos últimos anos.

Os vários processadores de um mainframe, por serem de alto desempenho e velocidade, são também de alto custo, o que impede a aplicação de redundância pura e simples (ou seja, a mera replicação de componentes) como técnica para aumentar a disponibilidade.

A solução mais comum, usada pelos fabricantes para aumentar a disponibilidade, é incluir um processador auxiliar com funções de console e manutenção. Esse processador de manutenção [Liu84], é de pequeno porte e autônomo, operando independentemente do mainframe, mas ligado diretamente a ele para poder supervisioná-lo (figura 3).

Um processador de manutenção deve ter capacidade de autoteste e ser construído com componentes confiáveis. Não deve interferir no processamento normal do computador. A existência desse processador não dispensa o uso de outras técnicas de tolerância a falhas, como códigos de correção e detecção de erros. Tais códigos permitem recuperar erros transitórios de alta incidência a baixo custo, sem necessidade de intervenção do processador de manutenção.

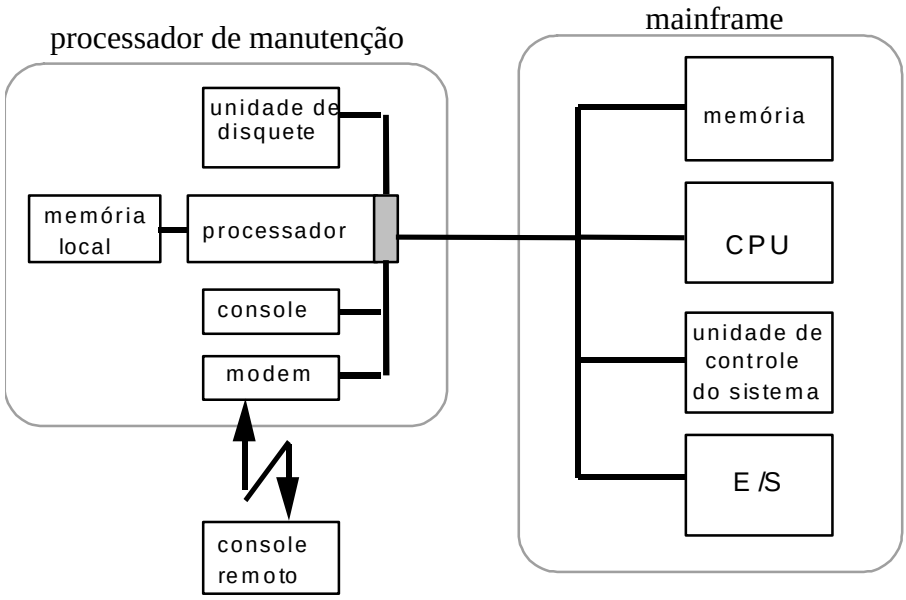

Figura 3 - Mainframe com processador de manutenção

As principais funções do processador de manutenção são:

- inicialização e console do sistema;
- supervisão contínua do sistema durante operação;
- diagnóstico de falhas;
- recuperação do sistema quando uma falha é detectada;
- teste durante desenvolvimento e produção.

Tolerância a falhas é suprida pelas funções de supervisão, diagnóstico e recuperação. Supervisionando continuamente o sistema, situações de erro podem ser imediatamente detectadas. Na detecção de um erro, o processador de manutenção atua da seguinte maneira:

- foi detectado um erro transitório: o processo em erro é interrompido e, tão rápido quanto possível, é recuperado para um estado livre de erros. O sistema operacional recebe então indicação de reiniciar o processo recuperado;
- foi detectado um erro permanente: é localizado o componente danificado de hardware. Procura-se então uma configuração que garanta a operação normal. Podem acontecer dois casos:

- reconfiguração é possível, mesmo com desempenho degradado. Por exemplo, se um processador falhar, o processo prejudicado pode ser alocado para outro processador;
- reconfiguração não é possível; o processador de manutenção diagnostica a falha, facilitando e acelerando o posterior reparo do sistema.

O processador de manutenção é ligado a uma central de manutenção através de rede. Assim o processador pode ser suprido de programas de diagnóstico sofisticados e atualizados quando necessário. Pode também avisar imediatamente o pessoal de manutenção da necessidade de trocar placas ou componentes específicos.

Desde que seja garantida alta confiabilidade para o processador de manutenção, ele representa uma solução eficaz e de baixo custo para tornar computadores de grande porte mais disponíveis, diminuindo o tempo de reparo do sistema. Um exemplo de sua aplicação pode ser encontrado nos computadores IBM de grande porte [RSNG82] e em quase todos os computadores de grande porte atuais.

4 Computadores de bordo

Na aviação exigências quanto a confiabilidade são extremamente altas, uma vez que vidas humanas estão em jogo em situações onde é impossível interrupção do sistema para reparos. Computadores de bordo tem por função controlar ativamente a aeronave em uma situação cuja normalidade é caracterizada por instabilidade. Correções para manter estabilidade de vôo devem ser feitas continuamente, no instante preciso em que são necessárias (ou seja, em tempo real) e com tempo de atuação curto.

Para computadores de bordo é exigida uma confiabilidade da ordem de 10^{-9} falhas por hora para um vôo de 10 horas. Além disso deve ser considerado:

- reparo é possível apenas durante os intervalos de vôo, e não mais freqüentemente do que a cada centena de horas de vôo;
- não é admissível qualquer tipo de interrupção no funcionamento do sistema;

Essas exigências extremas quanto à confiabilidade só podem ser alcançadas através da aplicação em larga escala de tolerância a falhas. Como exemplo podem ser citados dois computadores de bordo desenvolvidos na década de 70 sob encomenda da NASA: FTMP (Fault Tolerant Multi-Processor) e SIFT (Software Implemented Fault Tolerance). Os dois processadores foram projetados tendo por base a mesma especificação. Tanto FTMP [HSL78] como SIFT [Wens78] são baseados em redundância modular tripla (TMR). Entretanto, nos dois sistemas o votador é implementado de forma diversa. No FTMP o votador é um elemento de hardware, todos os processadores são sincronizados e o relógio central é tolerante a falhas. No SIFT votação é realizada por software, os processadores são assíncronos, sem relógio central, de tal forma que também o sincronismo para o fornecimento de resultados para votação deve ser garantida por software.

Os dois sistemas, FTMP e SIFT, apresentam alta confiabilidade para aplicações em tempo real. FTMP apresenta entretanto um esquema de votação mais eficiente e mais rápido, pois é realizado em hardware. No FTMP tolerância a falhas não é visível a partir da aplicação, ao contrário do que ocorre no sistema SIFT onde a votação é realizada por software.

Baseado no SIFT foi desenvolvido o computador de bordo para o Space Shuttle [Prad96], o que de certa forma atesta que a flexibilidade obtida pela votação em software supera em vantagem a velocidade obtida pela votação em hardware.

5 Sistemas Comerciais Tolerantes a Falhas

São citados dois exemplos de computadores de grande porte especialmente desenvolvidos para aplicações comerciais tolerantes a falhas: Tandem [Katz78] e Stratus [Hend83]. Esses dois sistemas foram os mais populares para aplicações em sistemas comerciais de transações [Serl84] durante a década de 80 até meados da década de 90. Tandem e Stratus são também dois bons exemplos de mecanismos de tolerância a falhas implementados em software (Tandem) e em hardware (Stratus).

Atualmente a área de computadores tolerantes a falhas de grande porte é praticamente inexpressiva em termos de novidades e negócios. Todos os grandes fabricantes comercializam soluções ditas de alta disponibilidade ou mesmo tolerantes a falhas. Tandem, incorporada a Compac, e a Stratus continuam aplicando suas técnicas proprietárias na área de servidores de redes.

5.1 Tandem

Um sistema é composto vários módulos computacionais. Cada módulo é formado por um processador, uma memória local, um canal de entrada e saída e fonte de alimentação (figura 4), interligados por um barramento duplicado. Além dos módulos processadores, o sistema dispõe de uma série de controladores de dispositivos de entrada e saída. Os controladores podem aparecer duplicados. Cada um deles está conectado a dois canais de E/S.

Complementando redundância em hardware, o sistema possui também redundância dinâmica em software. O sistema operacional GUARDIAN é formado por um kernel e um grande número de processos, em especial processos de supervisão para cada um dos processadores. Tanto para processos do sistema como para processos do usuário, GUARDIAN permite a criação de pares. Um par é formado por um processo primário ativo e um processo substituto passivo. O processo primário envia pontos de recuperação ao processo substituto.

Diagnóstico de erros se processa da seguinte forma:

- erros em um módulo são detectados em outros módulos processadores;
- a cada segundo, o processo supervisor de um módulo envia sinal de vida a todos os outros módulos no sistema;
- a cada dois segundos, o processo supervisor verifica se recebeu sinal de vida de cada um dos outros módulos. Se faltar um sinal, o processo entende que o módulo correspondente falhou.

Além desse controle mútuo, para cada operação de entrada e saída é realizado controle de time-out. Em caso de falha, o processo de entrada e saída substituto entra em operação.

Um vez diagnosticada a falha em um módulo, todos os processos substitutos relacionados aos processos primários que estavam sendo executados no módulo são

rolados para o último ponto de recuperação (recuperação por retorno) e ativados, tornando-se então processos primários. O sistema é reconfigurado em função dos novos processos primários. Tão logo o módulo faltoso seja reparado, os novos processos primários criam seus processos substitutos nesse módulo. Em caso de falha de um canal de entrada e saída, o processo substituto correspondente é rolado e ativado, enquanto o processo primário é desativado passando a ser substituto do primeiro.

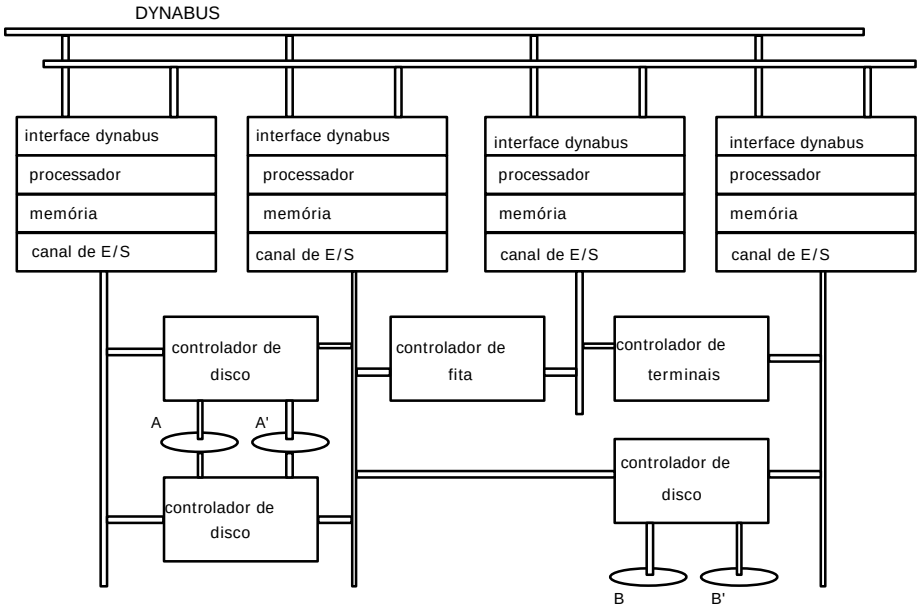

Figura 4 - Tandem NonStop

5.2 Stratus Continuous Processing

Um sistema típico pode ser composto de 1 a 32 módulos, interconectados através de uma rede local (Strata Link) (figura 5). Os elementos em um módulo são interligados por um barramento interno. Os módulos do sistema Stratus não estão disponíveis para redundância dinâmica.

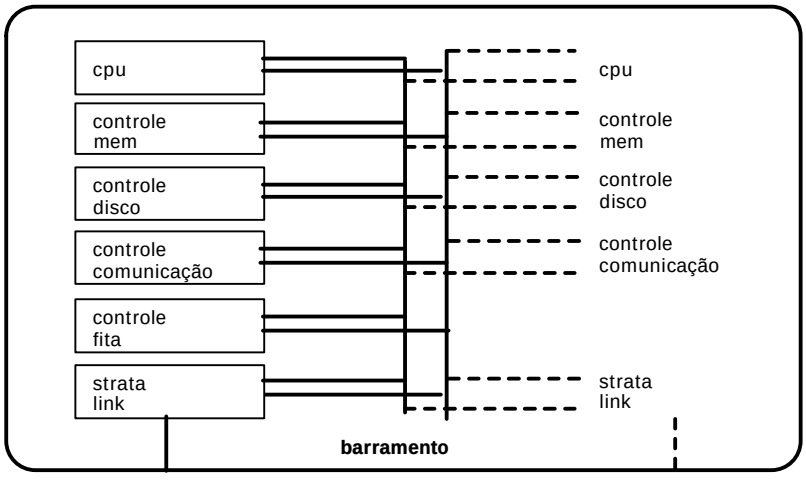

Figura 5 - Módulo de um computador Stratus Continuous Processing

Cada módulo é composto de dois grupos idênticos de componentes de hardware e um circuito lógico para comparação dos resultados de todas as operações que são realizadas

em paralelo (redundância estática). Todos os módulos podem aparecer por sua vez duplicados. Essa duplicação entretanto é transparente ao usuário e às aplicações.

O sistema operacional VOS é um sistema multiusuário que permite acesso transparente aos recursos do sistema através da rede local. Tanto o sistema operacional como os programas de aplicação apresentam apenas alguns recursos óbvios de tolerância a falhas, uma vez que o sistema foi especificado para prover tolerância a falhas por hardware.

Cada módulo do sistema compara os resultados fornecidos pelos elementos duplicados. Quando a comparação indica erro, nenhum resultado é fornecido como saída, o módulo é desconectado do sistema, sendo então enviado um sinal de erro a um programa de manutenção. Esse programa providencia testes no módulo para determinar se a falha é permanente ou transitória. Em ambos os casos, é registrado o problema e indicado o erro em um terminal de supervisão. Se o módulo faltoso aparecia duplicado no sistema, sua falha permanece invisível à aplicação, pois a unidade redundante garante a continuidade do processamento.

Caso o programa de manutenção tenha detectado uma falha transitória, o módulo que sofreu a falha é ressincronizado com a unidade redundante correspondente e entra imediatamente em operação. Caso seja uma falha permanente, o módulo é substituído manualmente sem interromper o processamento normal.

6 Tolerância a Falhas em Arquiteturas Paralelas

Dada a baixa relação custo/desempenho característica dos microprocessadores, sistemas de computação especificados para operar em alta velocidade são construídos preferencialmente pela conexão de um grande número de microprocessadores em paralelo. Algumas aplicações específicas se prestam bem para processamento paralelo, como por exemplo reconhecimento de padrões e processamento de imagens.

Esse tipo de aplicação permite interrupções curtas para reparo do sistema, mas de preferência não muito freqüentes, pois a distribuição de tarefas para os vários processadores demanda um tempo de inicialização considerável, comprometendo o desempenho. Devido a grande complexidade desses sistemas, que podem contar com dezenas, centenas e em casos extremos até com milhares de processadores, o emprego de técnicas para garantir operação livre de falhas de hardware é necessária.

Pela própria natureza do processamento paralelo, existe um bom número de unidades redundantes. Essa característica pode ser usada também vantajosamente para garantir tolerância a falhas.

7 Sistemas Distribuídos

Sistemas distribuídos são construídos por vários processadores independentes. Esses sistemas se diferenciam de computadores paralelos pelo acoplamento fraco entre os processadores, ou seja, os elementos de um sistema distribuído não tem acesso a uma memória comum. Toda a interação deve ser feita por troca de mensagens através de canais de comunicação. Além disso, sistemas distribuídos são geralmente construídos com elementos não homogêneos, assíncronos e sem um controle centralizado.

Sistemas distribuídos apresentam um problema intrínseco, qual seja, garantir a integridade e consistência dos dados distribuídos pelos vários processadores. Esse problema força a aplicação de técnicas de tolerância a falhas, mesmo quando não existe qualquer outra exigência mais forte quanto a confiabilidade imposta pela aplicação.

Assim como em arquiteturas paralelas, sistemas distribuídos apresentam um redundância natural, extremamente proveitosa para o emprego de técnicas de tolerância a falhas. Defeito em um processador não precisa provocar necessariamente a queda de todo o sistema, e o sistema pode ser facilmente reconfigurado usando apenas os processadores disponíveis.

A área de tolerância a falhas em sistemas distribuídos é vasta e excitante. Garantir dependabilidade envolve solucionar problemas de consenso, ordenação e atomicidade na troca de mensagens entre grupos de processos, sincronizar relógios quando necessário, implementar réplicas consistentes de objetos, garantir resiliência de dados e processos num ambiente sujeito a quedas de estações tanto clientes como servidoras, particionamento de redes, perda e atrasos de mensagens e, eventualmente, comportamento arbitrário dos componentes do sistema. Leitores interessados no assunto podem encontrar maiores detalhes nos livros do Jalote (Jal94), Birman (Bir96, Mullende (Mul93) e Porto [Jans97].

8 Conclusão

O texto apresentou algumas arquiteturas de computadores tolerantes a falhas. Praticamente todos os sistemas apresentados toleram erros provocados por falhas de hardware. Apesar de grande parte dos computadores citados não serem mais comercializado, suas arquiteturas sugerem soluções interessantes, independentes da tecnologia e época que foram usadas, e que podem ser aplicadas em sistemas atuais.

9 Bibliografia e referências bibliográficas

[Aviz98] AVIZIENIS, A. Infraestructure-based design of fault-tolerant systems. In: Proceedings of the IFIP International Workshop on Dependable Computing and its Applications. DCIA 98, Johannesburg, South Africa, January 12-14, 1998. p. 51-69.

[Bir96] BIRMAN, K. *Building secure and reliable network applications*, 1996

[HSL78] HOPKINS, A. L; SMITH, T. B.; LALA, J. H. FTMP - a highly realible fault-tolerant multiprocessor for aircraft. *Procedings of the IEEE*, New York, 66(10):1221-1239, Oct. 1978.

[Jal94] JALOTE, P. *Fault tolerance in distributed systems*. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[Jans97] JANSCH-PORTO, I. E. S; WEBER, T. S. Recuperação em Sistemas Distribuídos. In: XVI Jornada de Atualização em Informática, XVII Congresso da SBC, Brasília, 2-8 agosto de 1997. anais. pp 263-310

[John84] JOHNSON, D. The Intel 432: a VLSI architecture for fault-tolerant computer systems. *Computer*, New York, 17(8):40-48, Aug. 1984.

[Katz78] KATZMAN, J. A. A fault-tolerant computing system. In: Hawaii International Conference of System Sciences, 1978, *Proceedings*. p. 85-102.

[Lapr85] LAPRIE, J. C. Dependable computing and fault-tolerance: concepts and terminology. In: Annual International Symposium on Fault Tolerant Computing, 15. Ann Arbor, jun. 19-21, 1985. *Proceedings*. New York, IEEE, 1985. p. 2-11.

[Lapr98] LAPRIE, J. C.; Dependability: von concepts to limits. In: Proceedings of the IFIP International Workshop on Dependable Computing and its Applications. DCIA 98, Johannesburg, South Africa, January 12-14, 1998. p. 108-126.

[Liu84] LIU, T. S. The role of a maintenance processor for a general-purpose computer system. *IEEE Transations on Computers*. New York, c-33(6):507-517. June 1984.

[LyVa62] LYONS, R.E.; VANDERKULK, W. The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research and Development*. New York, 6(3): 200-209, abr. 1962.

[Mul93] MULLENDE, S. *Distributed systems*. Addison-Wesley, ACM Press, New York, 1993.

[Prad96] PRADHAN, D. K., *Fault-Tolerant System Design*. Prentice Hall, New Jersey, 1996.

[RSNG82] REILLY, J; SUTTON, A.; NASSER, R.; GRISCOM, R. Processor controller for the IBM 3081. *IBM Journal of Research and Development*, 26(1):22-29. Jan. 1982.

[SiSw82] SIEWIOREK, D. P.; SWARZ, R. S. *The Theory and Practice of Reliable System Design*. Bedford, Digital, 1982.

[Toy78] TOY, W. N. Fault-tolerant design of local ESS processors.*Proceedings of the IEEE*, New York, 66(10):1126-1145, Oct. 1978.

[VonN56] VON NEWMANN, J. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In: *Automata Studies*, Shannon & McCarthy eds. Princeton Univ. Press, 1956. p. 43-98.

[Web90] Weber, T.; Jansch-Pôrto, I.; Weber, R. *Fundamentos de tolerância a falhas*. Vitória: SBC/UFES, 1990. (apostila preparada para o IX JAI - Jornada de Atualização em Informática, no X Congresso da Sociedade Brasileira de Computação).

[Wens78] WENSLEY, J. H. et al. SIFT: design and analysis of fault-tolerant computer for aircraft control. *Proceedings of the IEEE*, New York, 66(10):1240-1254, Oct. 1978.

[Wens83] WENSLEY, J. H. Industrial-control system does things in threes for safety. *Electronics*, New York, 56(2):98-102. Jan. 1983.