Um Modelo Computacional Tolerante a Falhas para Aplicações Paralelas Utilizando MPI

Oberdan R. Pinheiro[1], Josemar R. de Souza[1,2]

[1]Centro Integrado de Manufatura e Tecnologia (SENAI-CIMATEC) – Salvador, BA – Brasil

[2]Universidade do Estado da Bahia (ACSO/UNEB) – Salvador, BA – Brasil

{oberdan.pinheiro,josemarsbr}@gmail.com

Resumo. *O desempenho computacional disponibilizado pelos sistemas paralelos resulta da capacidade de dividir o trabalho em partes menores e encaminhar cada uma delas para ser processada paralelamente em diferentes nós de um cluster computacional. A interrupção de uma das partes paralelizadas pode comprometer a computação como um todo, pois a aplicação depende do resultado de todos os componentes paralelizados. A pesquisa propõe um modelo computacional utilizando o paradigma SPMD implementado sob o esquema mestre/trabalhador para detectar e tratar falhas da classe fail stop em aplicações paralelas característica não presente no padrão MPI 1.1.*

1. Introdução

Tradicionalmente, os clusters baseados em passagem de mensagens foram construídos tendo - se como principal preocupação o provimento de um ambiente eficiente e portável, relegando a segundo plano a questão da tolerância a falhas. No caso de aplicações paralelas que utilizam varias máquinas e cuja execução se estende por muitas horas, à falha de uma única máquina neste período normalmente faz com que toda a computação já realizada seja perdida.

A pesquisa propõe um modelo computacional utilizando o paradigma SPMD implementado sob o esquema mestre/trabalhador para detectar falhas da classe fail stop em aplicações paralelas utilizando MPI, permitindo que as mesmas sejam reiniciadas automaticamente em outro nó disponível do cluster. O modelo utiliza a técnica de registros em log para o provimento de tolerância à falhas e a técnica de heartbeat para efetuar o monitoramento das aplicações paralelas.

2. Tolerância a Falhas

O termo "tolerância a falhas" foi apresentado originalmente por Avizienis em 1967. Entretanto estratégias para construção de sistemas mais confiáveis já eram usadas desde a construção dos primeiros computadores. Apesar de envolver técnicas e estratégias tão antigas, a tolerância à falhas ainda não é uma preocupação rotineira de projetistas e usuários, ficando sua aplicação quase sempre restrita a sistemas. [Weber, 2003]

Tolerância a falhas tem como objetivo permitir que um sistema comporte-se de maneira bem definida durante a ocorrência de falhas, visando minimizar o aparecimento

das mesmas ou então tratá-las quando ocorrerem. Esse comportamento pode ser enquadrado nas classes conhecidas por crash failure, fail stop e bizantino [F. Cristian 1991] [SCHNEIDER and Fred B. 1990].

2.1. Tolerância a Falhas em Sistemas Distribuídos

Sistemas distribuídos apresentam uma redundância natural, extremamente proveitosa para o emprego de técnicas de tolerância à falhas. O defeito em um nodo processador ou na rede de comunicação não precisa provocar necessariamente a queda de todo o sistema, e o sistema pode ser reconfigurado usando apenas os nodos disponíveis. [Weber, 2003]. As técnicas de tolerância à falhas têm como objetivo obter sistemas computacionais confiáveis e com uma maior dependabilidade.

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. [Weber, 2003].

Apesar disso, em seu trabalho Batchu (2003), demonstra que o padrão MPI apenas se preocupa em fornecer alto desempenho, escalabilidade e portabilidade entre arquiteturas [Batchu 2003].

3. Trabalhos Correlatos

Existem algumas soluções de tolerância à falhas para o padrão MPI.

CoCheck utiliza a técnica de checkpoint, implementada através de um coordenador central, mas que incorre em um grande overhead devido ao registros dos checkpoints [Bosilca 2002].

O FT-MPI é uma das soluções que não implementa as técnicas tradicionais de tolerância a falha, esta foi desenvolvida por Batchu (2003) e traz fortes mudanças a semântica do padrão MPI. Esta solução implementa um gerenciador de falhas através da manipulação da estrutura do padrão MPI communicator.

Entre os problemas encontrados nas soluções propostas estão a existência de alto overhead ou ainda modificações não previstas ao padrão MPI.

4. Modelo Conceitual

Para que um programa possa ser executado por um computador paralelo com memória distribuída e passagem de mensagem uma das opções é distribuir a mesma cópia do seu código entre os processadores, e os distintos conjuntos de dados a processar serão distribuídos entre as diversas máquinas. Tem-se um modelo chamado SPMD (Single Program Multiple Data) [Josemar Souza et al. 2003].

O modelo proposto utiliza o paradigma de programação SPMD, implementado sob o esquema mestre/trabalhador. A Figura 1 apresenta o modelo proposto, o qual é detalhado nas seções posteriores.

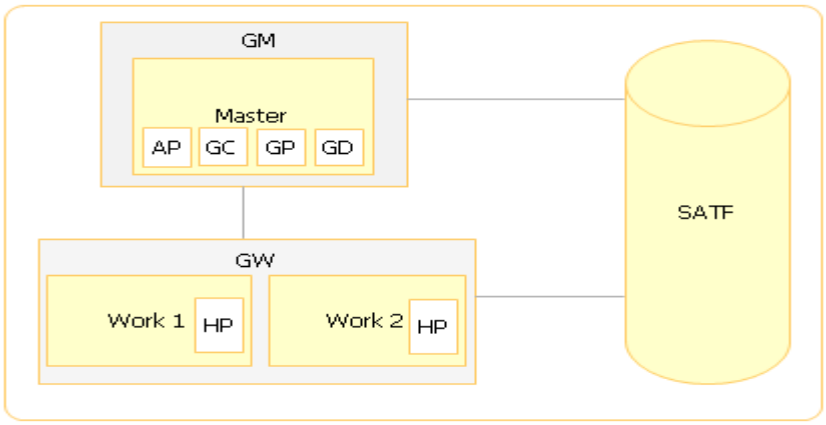

Figura 1. Modelo proposto

O GM (Grupo Master) é o módulo responsável por decompor o problema em diversas tarefas menores e distribuir essas tarefas entre os workers. Esse grupo é composto de outros componentes: AP (Aplicação Paralela) é o algoritmo paralelizado, GC (Gestor de Configuração) é o componente responsável por manter o ambiente configurado, ou seja, o CG é responsável por manter o estado dos workers participativos no cluster computacional através de uma estrutura de dados armazenada no SATF conforme demonstrado na Figura 2.

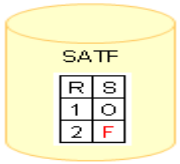

Figura 2. Estrutura de dados para manter o estado dos workers

O R (Rank) representa a posição e o id do work dentro do cluster, já o S (Status) determina o estado atual em que se encontra o work, podendo assumir os valores: O (Operante) e F (Falho). O GP (Gestor de Processos) é o componente responsável por aguardar o envio de pacotes periódicos (heartbeat) por parte dos workers, quando o GP detectar que um work não enviou o pacote dentro do tempo estabelecido, então o mecanismo assume que o nó que parou de enviar os pulsos está indisponível e solicita ao GC que atualize o estado do work para falho. Para permitir a continuidade da computação o GP em conjunto com o mestre recupera o bloco de dados destinado ao work que parou de funcionar através dos registros em log, em seguida o master vai escalonar outro work disponível para dar continuidade ao processamento. Todo o processo com estado falho será desligado logicamente do cluster e não mais terá tarefas delegadas para processar. Já o GD (Gestor de Dados) é o responsável por manter os registros em log alocados para os workers. Sempre que um conjunto de dados for enviado para um work o master solicitará ao GD que registre em log o bloco de dados enviado para o work no SATF, em caso de falha o master poderá solicitar ao GD o bloco de dados enviado para o work que parou de funcionar.

O SATF (Sistema de Arquivos Tolerante a Falhas) é o local de armazenamento confiável que utiliza o sistema de arquivos de rede NFS (Network File System) aonde

são registrados os logs e estruturas de dados para manter o cluster operante em caso de falha. Esse componente é imprescindível em razão de que os arquivos produzidos e recuperados pelas técnicas de tolerância em software serão armazenados e recuperados do mesmo.

O GW (Grupo Work) é o módulo que recebe os trabalhos, processam e devolvem para o master, que gerencia, organiza e controla os dados processados pelos workers. O componente HP (Heartbeat Process) faz uso de um thread para implementar o mecanismo do heartbeat com apenas uma única transmissão, nesse cenário os workers enviam periodicamente mensagem para o master. Esta alternativa irá gerar uma quantidade menor de tráfego na rede conforme demonstrado na Figura 3.

Figura 3. Heartbeat com apenas uma única transmissão.

5. Metodologia para Avaliação do Modelo

A pesquisa desenvolvida é experimental, tendo como objeto de estudo um modelo computacional para detectar e tratar falhas da classe fail stop em aplicações paralelas característica não presente no padrão MPI 1.1.

O experimento será conduzido no Centro Integrado de Manufatura e Tecnologia (SENAI CIMATEC). O ambiente consistiu de um cluster de computadores formado por 16 blades HP ProLiant DL120 G6 Quad core Intel® Xeon® HP X3450 (2.67GHz, 95W, 8MB, 1333, HT, Turbo), utilizando-se do sistema operacional Ubuntu/Linux 64 bit, mpich 1.2.52 que implementa todo o padrão MPI 1.1 e o compilador gnu gcc 3.2.3. A rede de conexão utilizada é baseada no padrão Ethernet. A aplicação paralela para a validação do modelo será implementada utilizando o algoritmo de multiplicação de matrizes, por ser facilmente escalável tanto em cômputo quanto em comunicação. A inserção da falha deverá ocorrer através da desconexão física de alguns nodos do cluster.

5. Conclusão

A partir da implementação do modelo esperamos demonstrar a eficácia da solução apresentada em provê aspectos de tolerância à falhas com a diminuição do overhead para a monitoração do ambiente, pois o modelo faz uso da técnica do heartbeat com apenas uma única transmissão. Algumas melhorias e acréscimos ao modelo vêm sendo planejados tais como tratamento de falhas referente ao processo master, isso poderá ser alcançado através dos backups em registros de log e das estruturas de dados armazenadas no SAFT que utiliza o sistema de arquivos de rede NFS (Network File System).

Referências

Batchu, Rajanikanth Reddy. Incorporating Fault - Tolerant Features into Message - Passing Middleware. Maio de 2003. 105 f. Dissertação de Mestrado. Department of Computer Science and Engineering, Faculty of Mississippi State University, Mississippi, 2003.

Bosilca, George et al. MPICH- V Toward a Scalable Fault Tolerant MPI for Volatile Nodes, IEEE, Junho de 2002.

F. Cristian, “Understanding Fault-Tolerant Distributed Systems”. Communications of the ACM, v. 34, n.2, pp. 57-78, Feb. 1991.

JOHNSON, David Bruce 1989. Distributed System Fault Tolerance Using Message Logging and Checkpointing – Ph.D thesis, Rice University, (Dez.1989), Houston , Texas.

Josemar Souza, D. Rexachs, E. Luque. Análise da distribuição de carga em um cluster heterogêneo. Universidade Católica do Salvador Bahia, Salvador, 2003.

LAMPORT, Leslie e SHOSTAK, Robert e PEASE,Marshall 1982. The Byzantine Generals Problem – ACM Transactions on Programming Language and Systems, vol. 4, 3 (Jul. 1982), 382-401.

ROBERTSON, Alan. Linux-HA Heartbeat System Design, http://www.usenix.org/publications/library/proceedings/als00/2000papers/papers/full_papers/robertson/robertson.pdf, (Jul/2011).

SCHNEIDER, Fred B. 1990. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial – ACM Computing Surveys, vol. 22, 4 (Dez.1990), 299-319.

Weber, Taisy Silva, Um roteiro para exploração dos conceitos básicos de tolerância a falhas. Apostila do Programa de Pós- Graduação – Instituto de Informática - UFRGS. Porto Alegre, 2003.