Detecção de Falhas Hierarquizada baseada em *Threads* para GRIDS

Marcelo C. P. e Santos e Lúcia M. A. Drummond

Instituto de Computação – Universidade Federal Fluminense (UFF)
Rua Passo da Pátria, 156 - Bloco E - 3° andar - Niterói - RJ, CEP 24.210-240

E-mail:{mpinto,lucia}@ic.uff.br

Resumo: *Este trabalho apresenta o projeto de um sistema para detecção de falhas para ambientes GRIDS. O sistema, desenvolvido em C e MPI, é baseado em threads e é hieraquizado, o que o torna escalável e resulta em um baixo impacto no desempenho da aplicação.*

1. Introdução

Este trabalho trata do problema de a detecção de falhas em ambientes distribuídos. O objetivo foi projetar um sistema de detecção escalável, que introduzisse o mínimo de impacto no desempenho da aplicação, fosse adaptado ao ambiente de GRIDS computacionais e de fácil utilização em diferentes aplicações. O sistema foi desenvolvido utilizando a linguagem de programação C e a biblioteca de comunicação MPI ("Message Passing Interface") . As principais características deste sistema que o diferenciam dos demais [1][2] são: i) o sistema foi desenvolvido como um conjunto de *threads* que atualizam estruturas de dados, para consulta pela aplicação, contendo a informação sobre a atividade normal ou "morte" dos diversos processadores, ii) a detecção é hierarquizada, ou seja, existe um detector local responsável pela detecção de falhas de um conjunto de processadores e um líder geral, chamado de detector central que trata da gerência dos detectores locais.

Acreditamos que estas características nos permitiram atingir integralmente o objetivo mencionado.

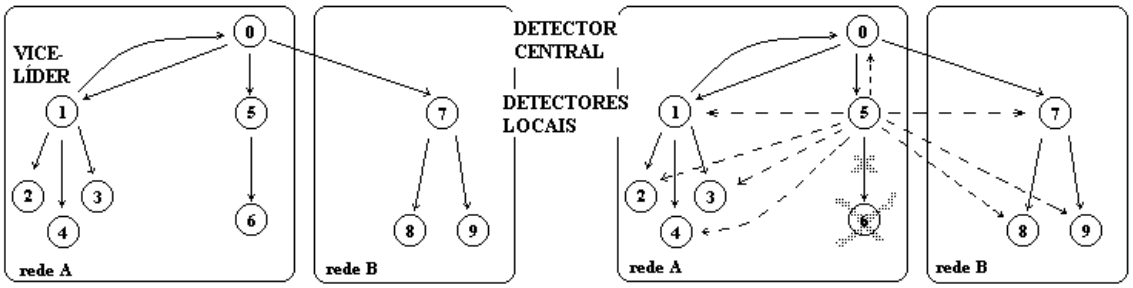
2. Projeto do Sistema de Detecção de Falhas

O sistema de detecção de falhas proposto é baseado na associação de um conjunto de *threads* para cada processador utilizado pela aplicação, cada uma delas com uma função específica. Inicialmente os processadores são divididos em grupos, e a cada grupo é associado um detector local. Os detectores locais são controlados por um detector central. Os detectores locais enviam solicitações de sinal de vida ao seu conjunto de processadores associado. Os tempos de espera pelas respostas são armazenados e múltiplos destes são utilizados como limites de espera para os próximos pedidos de sinal de vida. Se um limite for violado o processador é considerado "morto". Isto torna o sistema adaptativo às variações de tempo de execução e comunicação que comumente ocorrem em GRIDS.

Os detectores locais são gerenciados pelo detector central que ao decretar a morte de um deles avisa aos demais processadores, e tenta eleger no grupo um novo detector local. Não conseguindo, todo o grupo é declarado como morto.

Um "vice-líder" detecta falhas no detector central, e assume suas funções em caso de falha.

Resumindo, basicamente são empregadas quatro tipos de *threads*: uma para detectar falhas em processos (detector local), uma para detectar falhas em detectores (detector central), uma para enviar mensagens de sinal de vida e uma outra para registrar a morte de processos, conforme pode ser visto na figura a seguir.



A) OS DETECTORES ENVIAM SOLICITAÇÕES DE SINAL DE VIDA E ESPERAM RECEBÊ-LAS EM UM TEMPO PRÓXIMO AO DA ÚLTIMA RESPOSTA.

B) AO SER DETECTADA UMA MORTE, TODOS OS PROCESSADORES SÃO AVISADOS (LINHA PONTILHADA).

3. Adaptação da Aplicação

Considerando que a cada processador é associado exatamente um processo da aplicação, o usuário deverá iniciar o MPI com suporte a *threads* serializadas através do MPI_Init_Thread(&argc, &argv, MPI_THREAD_SERIALIZED, &Conseguido) [3] e chamar uma função (iniciaDetector) para que o detector entre em operação. Durante a execução da aplicação, estará disponível um vetor informando os nós ativos, (exemplo de utilização: if(ativo[processo])). Antes de executar o MPI_Finalize, a aplicação deverá chamar uma função que terminará as *threads* de detecção de falhas (fimDetector).

3. Conclusões Preliminares

A utilização de *threads* para a implementação do detector torna o tratamento das mensagens referentes à detecção de erros independente da aplicação. Desta forma, não necessitamos intercalar código de aplicação e detecção de erros, tornando o sistema mais modular e de manutenção mais fácil. Tais vantagens podem ser estendidas a outras necessidades dos sistemas paralelos, com pequeno impacto no desempenho da aplicação, conforme temos verificado através de testes preliminares.

O sistema proposto nos parece ser bastante interessante para ambientes GRIDS, especialmente porque podemos associar um detector local de falhas por grupos de processadores que estejam geograficamente mais próximos, reduzindo a troca de mensagens de detecção entre processadores interconectados por canais de comunicação mais lentos.

References

[1] Robertson, A: Linux-HA Heartbeat System Design, Proceedings of the Forth Annual Linux Showcase and Conference (ALS), 2000.

[2] Woo, Namyoon, Heon Y. Yeon, Taesoon Park, MPICH-GF: Transparent Checkpointing and Rollback-Recovery for Grid-enabled MPI Processes, IEICE Transactions on Information and Systems , Vol E87-D, No 7, Julho 2004, pp. 1820-1828.

[3] Message Passing Interface Fórum. MPI-2: Extension to the Message Passing Interface, julho de 1997.