DPM - Distributed Processor Manager

Fausto Richetti Blanco, Diego Wolmer Garcia, César A. F. De Rose

Pontifícia Universidade Católica do Rio Grande do Sul
Avenida Ipiranga 6681, Telefone: 5133203558 ramal 4463, Fax: 5133203758
{blanco, wolmer, derose}@cpad.pucrs.br

Introdução

Atualmente, a tendência para área de processamento de alto desempenho é a utilização de arquiteturas COW (*clusters of workstations*) devido, principalmente, aos custos reduzidos e aos resultados satisfatórios atingidos. No entanto, um *cluster* exige um certo esforço no que se refere à configuração e gerência.

Geralmente (em *clusters* com um elevado número de nós principalmente), uma aplicação não utiliza todos os nós do agregado, deixando vários outros disponíveis. Esses nós livres podem ser utilizados por outros usuários sem que ocorra perda de performance no processamento, pois as aplicações executam em computadores diferentes.

Dessa forma, reservar todos os nós de um *cluster* para cada usuário que deseja executar algo não é satisfatório. É desejado, entretanto, que cada usuário tenha um grupo de nós do cluster ao qual apenas ele tem acesso. As ferramentas que cuidam da distribuição dos nós aos usuários são chamadas de gerentes.

Existem, ainda, aplicações que necessitam de mais nós à medida que são executadas. Portanto, o gerente deve dar a possibilidade de que a aplicação possa agregar mais nós ao seu grupo em tempo de execução. Esse processo é chamado de alocação dinâmica.

O DPM (*distributed processor manager*) é um gerente totalmente distribuído, desenvolvido no CPAD [CPA 01], que permite que as aplicações requisitem ou liberem nós em tempo de execução. Ainda, é um gerente extremamente rápido e facilmente adaptável a qualquer topologia de rede. Permite também a alteração da topologia lógica mapeada dinamicamente, de maneira simples e rápida.

Alocação Dinâmica

O compartilhamento do agregado torna possível que múltiplos usuários utilizem diferentes nós de um mesmo *cluster*, evitando que haja um desperdício de processamento. Contudo, é possível que uma aplicação precise de maior processamento durante sua execução, criando uma necessidade de ocupar mais nós.

Para que essa necessidade seja atendida, o gerente deve poder alocar nós aos usuários enquanto suas aplicações executam. É uma tarefa complexa, porém, promove maior flexibilidade e evita ainda mais o desperdício no agregado.

Atualmente, a maioria das aplicações distribuídas executam com um número fixo de nós. Entretanto, uma parte dessas aplicações possui um comportamento variável, isto é, em determinados momentos o melhor desempenho é atingido com um certo número de nós, e em outros momentos um número diferente de nós proporciona o melhor desempenho.

Este tipo de aplicação, chamada de aplicação maleável [HOL 59], atinge seu melhor desempenho através de *clusters* cujo gerente suporte alocação dinâmica. Pois, dessa forma, a aplicação pode utilizar mais ou menos nós, de acordo com a sua necessidade.

DPM - Distributed Processor Manager

O DPM (*distributed processor manager*) é um gerente que implementa alocação dinâmica de forma distribuída. Ou seja, ele não centraliza as informações de alocação de nós do *cluster* em um computador, tornando cada nó do agregado responsável por seu estado (alocado ou livre).

Isso torna o sistema ainda mais complexo, já que as informações sobre os nós estão espalhadas pelo agregado, criando a necessidade de se realizar buscas na rede para obter tais informações. Todavia, o objetivo de todo usuário de um *cluster* é o desempenho, portanto o DPM deve realizar tais buscas da forma mais rápida possível para que a performance das aplicações não seja comprometida.

Além disso, por ser um gerente distribuído, o DPM é executado em todos os nós do agregado na forma de um *daemon*. Este *daemon* foi cuidadosamente implementado, pois deve ocupar o mínimo possível de memória e necessitar de um nível extremamente baixo de processamento.

As mensagens dos *daemons* trafegam através de uma topologia lógica, que é mapeada sobre o agregado segundo um arquivo de configuração que deve ser criado pelo administrador do sistema. O DPM utiliza árvores para representar a topologia lógica utilizada, a figura 1 ilustra um exemplo de topologia passível de ser utilizada com o DPM.

Exemplo de Arquivo de configuração:

```
no1 no2 no3
no2 no4 no5
```

Topologia descrita:

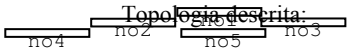

Figura 1

A vantagem de se poder mapear topologias lógicas sobre o agregado é que elas podem acompanhar a topologia física da rede, o que possibilita alocações de partes específicas da rede.

Funcionamento

O funcionamento do DPM baseia-se na topologia lógica mapeada sobre o agregado; entretanto, por ser distribuído, não existe um mapa completo dessa topologia em lugar algum (exceto no arquivo que a define). Um *daemon* conhece apenas os nós que são seus filhos (aqueles que vem abaixo dele na árvore descrita na topologia) e seu pai (aquele que vem acima dele na árvore). Ele só pode enviar mensagens aos seus filhos ou ao seu pai.

Se for necessário a um administrador, é possível modificar a topologia depois de inicializados os *daemons*, ele pode utilizar rotinas de administração que realizam operações de remoção e adição de filhos e de mudança de pai.

A alocação inicial dos nós do *cluster* é realizada pelo administrador, uma vez que esse deve ter controle sobre as aplicações que estão em execução. Assim, somente aplicações que tenham tido pelo menos um nó alocado inicialmente pelo administrador podem alocar ou desalojar nós.

Depois da alocação inicial, o usuário pode executar suas aplicações nos nós que lhe foram concedidos. Para utilizar o DPM (e conseqüentemente poder alocar ou liberar nós), o usuário deve utilizar uma biblioteca criada para a comunicação com o *daemon*. Essa biblioteca é responsável pela comunicação com o gerente local; este, por sua vez, se encarrega de propagar as tarefas através da rede mapeada. Isso está ilustrado na fig. 2.

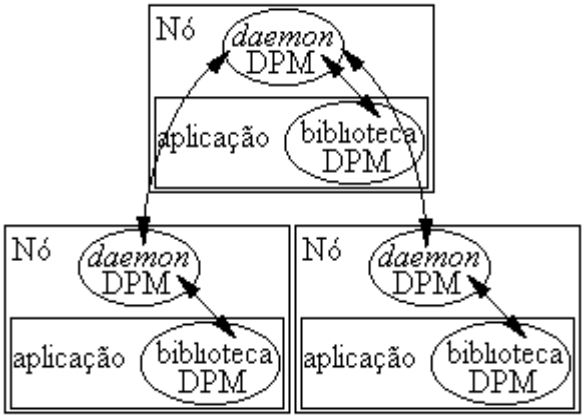

Figura 2

Todas as mensagens do DPM respeitam a topologia lógica em uso. Assim, toda mensagem que deve percorrer a rede inicia sua busca através dos filhos. Apenas se necessário a mensagem será propagada para o pai. Dessa forma, o DPM utiliza recursão para que toda a rede mapeada seja alcançada.

Utilização

O DPM possui uma API (*application programming interface*) onde estão implementadas rotinas, tipos de dados e códigos de erro. Estas rotinas se dividem em três classificações:

- rotinas de administração;
- rotinas de aplicação;
- rotinas comuns.

As rotinas de administração só podem ser utilizadas por administradores do sistema, e realizam as operações de inserção e remoção de aplicações no sistema e operações sobre a topologia lógica mapeada sobre o *cluster*.

As rotinas de aplicação são utilizadas pelos usuários, para que estes possam alocar e desalocar nós durante a execução de suas aplicações.

As rotinas comuns são utilizadas tanto por administradores como por usuários. Possuem um caráter mais genérico, como por exemplo, rotinas de inicialização e finalização da API, e manipulação de dados.

Além da API, estão implementados quatro utilitários que auxiliam o administrador a gerenciar o DPM. São eles: *startup*, que inicializa remotamente os *daemons* nos nós do cluster; *request*, que realiza a alocação inicial de nós; e *configure* e *xconf*, que geram o arquivo de descrição da topologia lógica. A única diferença entre esses últimos é que o *xconf* executa em modo gráfico, ao passo que o *configure* executa em modo texto.

Conclusão e Trabalhos Futuros

O DPM está em fase final de testes e atingiu um resultado final de performance muito alto. Em medições realizadas no iCluster [ICL 01], o tempo de alocação de 205 nós foi de 0,015662 segundos. Este dado é extremamente positivo, considerando a quantidade de nós e a complexidade da topologia utilizada pelo DPM (árvores).

A topologia lógica hierárquica criada pelo DPM diminui a interferência causada nas máquinas, uma vez que as mensagens nem sempre necessitam passar por todos os *daemons* do sistema.

A tolerância a falhas é um dos recursos que deve ser agregado ao DPM em breve, bem como a reconfiguração automática da topologia lógica. Ainda, de forma a prover mais transparência às aplicações, deverão ser modificados alguns ambientes de execução para permitir a criação de processos dinamicamente, de acordo com os nós disponibilizados pelo DPM.

Referências

[HOL 59] HOLLAND, J. et al. **A universal computer capable of executing an arbitrary number of sub-programs simultaneously**. Proc. East Joint Computer Conference, Vol 16, pp 108-113, 1959.

[ICL 01] The iCluster Project, 2001. http://icluster.imag.fr

[CPA 01] Centro de Pesquisa em Alto Desempenho. http://www.cpad.pucrs.br