UM ESTUDO PARA EXTENSÃO DO GET_METRICS DO LIMA PARA CONSULTAR INFORMAÇÕES DO PBS

Rodolfo Aparecido de Moraes Rodovalho[1]; Nilton Cézar de Paula[2]

[1]Estutande do Curso de Ciência da Computação da UEMS, Unidade Universitária Dourados;

E-mail: rodolfo_ramr@hotmail.com

[2]Professor do Curso de Ciência da Computação da UEMS; Unidade Universitária Dourados;

E-mail: nilton@comp.uems.br

Sistemas de Computação

Resumo

A proposta deste trabalho foi realizar um estudo para a extensão das funcionalidades do comando GET_METRICS do sistema de monitoramento LIMA (***L**ight-we**I**ght **M**onitoring **A**rchitecture*) para consultar informações do PBS (*Portable Batch System*) sobre recursos de computação existentes em uma grade computacional. A atual versão do GET_METRICS suporta apenas o Condor e o Ganglia, mas com essa extensão será possível estudar o problema da integração de informações de diferentes sistemas de gerenciamento de recurso local e então favorecer o uso de uma quantidade maior de recursos de uma grade computacional. A implementação da extensão do GET_METRICS vem para auxiliar em um descobrimento mais eficaz das informações, pois ele retorna todos nós do ambiente que atendem aos requisitos do usuário, caso existam.

Palavras-Chave: Sistema de Monitoramento. Grade Computacional. Gerenciamento de Recursos. Consultar Informações.

Introdução

A popularidade da Internet e o aumento da velocidade nas redes de comunicação têm possibilitado agregar recursos distribuídos de diferentes domínios administrativos para compor um poderoso ambiente de computação para a execução de uma grande quantidade de aplicações (FOSTER; KESSELMAN; TUECKE, 2001).

Um dos objetivos de uma grade é oferecer serviços, processamentos e dados de maneira transparente e, para isto, os recursos computacionais, como *clusters* e computadores,

precisam ser localizados eficientemente (FOSTER; KESSELMAN, 1999). Neste sentido, sistemas de monitoramento de recursos foram propostos para manter informações sobre os recursos de uma grade computacional.

Exemplos de sistemas de monitoramento de recursos são o Hawkeye (HAWKEYE, 2009), Ganglia (MASSIE; CHUN; CULLER, 2004) e o Monalisa (CIRSTOIU et al., 2007). Porém, todos eles, com exceção do Hawkeye, permitem distribuir as informações sobre os recursos de maneira hierárquica ou centralizada, mas replicando todas as informações, o que pode causar intenso tráfego nas redes de comunicação.

Para reduzir o tráfego de informações nas redes, foi proposto o sistema de monitoramento LIMA (***L**ight-we**I**ght **M**onitoring **A**rchitecture*) (DE PAULA, 2009), que também tem o objetivo de oferecer acesso fácil às informações distribuídas sobre os recursos de uma grade. O acesso às informações é realizado usando a interface GET_METRICS do LIMA que traduz uma solicitação do usuário em um conjunto de filtros e obtém as informações que se deseja de um banco de dados. A Figura 1 mostra a forma básica do comando GET_METRICS.

```
GET_METRICS < lista de métricas > TYPE <tipo>
TOOL  < ferramenta de monitoramento > TYPE <tipo>
LOCATION <domínio>
[WHERE <condição>]
```

Figura 1: Forma básica do comando GET_METRICS.

A cláusula GET_METRICS relaciona as métricas sobre as quais se deseja obter os valores. Ela aceita além dos nomes das métricas o operador ALL que representa todas as métricas e em TYPE é definido o tipo de métrica que pode ser estático ou dinâmico. Para a cláusula TOOL é informado a ferramenta que produz os valores das métricas (por exemplo, Ganglia ou Condor) e no TYPE o tipo de monitoramento alvo que pode ser recurso computacional, aplicação ou rede. Na cláusula LOCATION é definida o local de abrangência da consulta de hierarquia de distribuição de dados. E a última cláusula, WHERE, pode ser especificado o endereço do pool, o recurso para recuperar métricas dinâmicas ou ainda um filtro sobre determinadas métricas (DE PAULA, 2009).

Porém atualmente, o GET_METRICS consulta apenas informações sobre os recursos da grade que vêem do Condor (THAIN; TANNENBAUM; LIVNY, 2003) e do Ganglia. Para melhor entender o problema da integração de informações de diferentes sistemas de

gerenciamento de recursos local, faz-se necessário realizar um estudo para estender o GET_METRICS para consultar informações coletadas do PBS (BERMAN; FOX; HEY, 2003). O PBS é um sistema que foi originalmente criado para a NASA e que hoje é bastante utilizado em grades computacionais para gerenciar *clusters* e conjuntos de computadores de um domínio local e suas principais funções são: submeter uma aplicação para execução, monitorar e controlar a carga de processamento das execuções.

Materiais e Metódos

Primeiro realizou-se um estudo do PBS para saber seu funcionamento geral, com isto pode-se extrair informações necessárias para construir a extensão do GET_METRICS. A Figura 2 mostra um exemplo da saída do comando '*pbsnodes –a*' do PBS.

```
$ pbsnodes -a
Nome_do_Servidor
     state = free
     np = 4
     properties = QuadCore
     ntype = cluster
     status = opsys=linux,
          uname=Linux clt01 2.6.24-1-amd64 #1 SMP Sat May 10 09:28:10 UTC 2008 x86_64,
          sessions=? 0,          nsessions=?0,          nusers=0,          idletime=82978,
          totmem=11793408kb,          availmem=11734580kb,          physmem=3977796kb,
          ncpus=4,          loadave=0.00,          netload=55769945,
          state=free,          jobs=,          varattr=,          rectime=1224350600
```

Figura 2: Comando pbsnodes –a do PBS.

O exemplo mostra as características de um nó de execução de uma grade computacional. Com estas características o comando GET__METRICS fará uma filtragem de todos os nós do ambiente que atendam a determinados requisitos do usuário. Podemos destacar entre os parâmetros de saída: *state, opsys, np, jobs, totmem e availmem.*

No parâmetro *state* os argumentos podem assumir os seguintes valores***:*** *busy, down, free, job-exclusive, job-sharing, offline, reserve, time-shared e unknown*. Cada argumento representa um estado do nó com relação a execução de aplicações. Por exemplo, se o *state* do nó esta como *busy*, então significa que o nó está cheio e não receberá nenhuma aplicação para execução. O argumento *opsys* mostra o sistema operacional que esta em uso, o *np* o número de processadores, os *jobs* estão listados todos os trabalhos em execução no nó. Caso não apareça nada, significa que nenhum trabalho está sendo executado no momento. Em *totmem e*

availmem mostram-se, respectivamente, a quantidade de memória total e a quantidade de memória disponível no nó.

Resultados e Discussões

Analisando a saída do comando '*pbsnodes –a'* tem-se as configurações de todos os nós da grade computacional. Com base nisto, a implementação da extensão da interface GET_METRICS fará a separação das informações necessárias para o usuário, limitando assim a um número menor ou igual de nós no arquivo de saída.

Para a implementação, foi utilizada a linguagem C e uma lista encadeada como estrutura de dados principal. Além disto, quatro funções principais foram desenvolvidas: 'carrega_arquivo', 'analisa', 'busca' e 'verifica_condicao'.

A função 'carrega_arquivo', obtém todas as informações dos nós existentes. Para isso, utiliza-se de outras três funções para auxiliar no armazenamento das informações. Duas destas funções, a 'divide_parametro1' e 'divide_parametro2', efetuam a quebra dos argumentos e após isso passam os dados para a função 'insere_no' que armazena as informações.

A função 'analisa' faz uma análise do comando GET_METRICS, fornecido pelo usuário, e realiza as devidas checagens de consistência. Para fazer a análise são necessárias duas funções, uma chamada 'quebra_metrica' e outra chamada 'quebra_condicao', que auxiliam no armazenamento da busca. Se o comando GET_METRICS estiver correto, então a função 'analisa' chama a função 'busca'.

A função 'busca' realiza uma busca de acordo com os parâmetros passados pela função 'analisa', fazendo comparações entre os parâmetros fornecidos pelo usuário e os parâmetros contidos nos nós.

Por último tem-se a função 'verifica_condicao' que mostra os nós que atendem as especificações do usuário e escreve o resultado da busca em um arquivo de texto de saída.

Um exemplo do uso do comando GET_METRICS é a escrita da seguinte consulta: *GET_METRICS node TYPE estático TOOL pbs TYPE host LOCATION local WHERE np=3*. Isto vai gerar um arquivo de saída com o nome de todos os *hosts* (nós) locais que pertencem ao PBS e atendem ao requisito de terem a quantidade de processadores igual a 3.

Isto demonstra que o comando GET_METRICS auxilia na forma de selecionar nós que atendam as necessidades do usuário dando assim mais opções de gerenciamento da grade computacional.

Agradecimentos

A Deus porque é o ponto de referência e nada acontece sem a Sua permissão.

A minha família que sempre me deu forças para seguir em frente.

A UEMS pelo suporte financeiro, estrutura de ensino e pesquisa. Ao meu orientador Prof. Dr. Nilton Cézar de Paula por todo auxílio e tempo dedicado a este projeto. E a todas as pessoas que possibilitaram que este trabalho acontecesse.

Referências

BERMAN, F.; FOX, G. C.; HEY, A. J. G. 2003. **Grid Computing:** Making the Global Infrastructure a Reality. John Wiley & Sons, Inc.

CIRSTOIU, C. et al. 2007. **Monitoring, Accounting and Automated Decision Support for the ALICE Experiment Based on the MonALISA Framework.** In Proceedings of the 2007 Workshop on Grid Monitoring, p.39-44.

DE PAULA, N.; C. 2009. **Um Ambiente de Monitoramento de Recursos e Escalonamento Cooperativo de Aplicações Paralelas em Grades Computacionais.** Tese (Doutorado), Universidade de São Paulo.

FOSTER, I.; KESSELMAN, C. 1999. **The Grid:** Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. 2001. **The Anatomy of the Grid:** Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, v.15, p.200-222.

HAWKEYE 2004. Disponível em: <http://www.cs.wisc.edu/condor/hawkeye/>.
Acesso em: 20 junho.

MASSIE, M. L.; CHUN, B. N.; CULLER, D. E. 2004. **The Ganglia Distributed Monitoring System:** Design, Implementation, and Experience. In Proceedings of Parallel Computing, v.30, issue 7.

THAIN, D.; TANNENBAUM, T.; LIVNY, M. 2003. **Condor and the Grid, chapter in Grid Computing:** Making the Global Infrastructure a Reality. John Wiley & Sons, Ltd.