Algoritmos de Escalonamento de Tarefas em Grades Computacionais

Epifanio Dinis Benitez*, Gerson Geraldo H. Cavalheiro

Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Ciências Exatas e Tecnológicas
Universidade do Vale do Rio dos Sinos
São Leopoldo - RS - Brasil
epifanio@exatas.unisinos.br, gersonc@unisinos.br

Introdução

O advento de arquiteturas paralelas (ex. *clusters*) possibilitou que aplicações de áreas como a física e a biologia pudessem gerar resultados interessantes para a pesquisa. Este fato deu origem ao interesse dos pesquisadores em desenvolver algoritmos mais eficientes que por sua vez, necessitariam de uma arquitetura com abundantes recursos de armazenamento e processamento. Devido a essa e outras situações, surgiu a computação em grade (*grid computing*) [CIR 2003]. Sua estrutura está composta por máquinas geograficamente distribuídas, em escala regional e mundial, e apresenta um modelo computacional formado por servidores de aplicação ASP's (*Application Service Provider*) e de servidores de armazenamento SSP's (*Storage Service Provider*). Esta estrutura torna viável a execução de aplicações – tipicamente de simulação – que geram uma grande quantidade de dados, os quais devem ser distribuídos a outras máquinas, explorando de forma crítica os recursos de rede. Desta forma, a rede se torna a limitação do desempenho global do sistema. A solução a esta limitação se encontra no desenvolvimento de algoritmos de escalonamento que minimizem ao máximo o custo de comunicação entre as máquinas que participam da execução da aplicação, resultando em um ganho de desempenho global da aplicação e reduzindo o impacto global no uso da rede.

Apesar desta limitação, ambientes conhecidos [COS 2004, AND 2003] tentam obter ganho de desempenho através do escalonamento de tarefas desconsiderando informações importantes como o custo de comunicação e computação das tarefas. Entre esses algoritmos estão o WQ (*WorkQueue*) e o WQ-R (*WorkQueue with replication*). Nossa proposta consiste em minimizar o tempo de comunicação entre tarefas através da extensão do algoritmo DSC (*Dominant Sequence Clustering*) adaptando-o à computação em grade. Diferente dos demais algoritmos, este realiza o escalonamendo baseado em informações como o custo de processamento e comunicação das tarefas.

Na próxima seção se encontram descritos os algoritmos de escalonamento citados anteriormente. Na seqüência é apresentada a extensão do algoritmo DSC, seguida por uma avaliação de desempenho para validar o uso do algoritmo DSC como base de implementação, e finalmente, as conclusões.

*UNIBIC/UNISINOS

Algoritmos de escalonamento

WQ é um algoritmo de escalonamento que caracteriza aplicações do tipo *bag-of-tasks*. Sua estrutura é composta por uma fila de tarefas a serem executadas e por um escalonador, responsável por atribuir tarefas à máquinas. O escalonamento se dá através do envio de tarefas a todas as máquinas que estão ociosas, enquanto exista trabalho na fila de tarefas. Pode-se observar que o algoritmo foi projetado para manter os processadores ocupados pela maior quantidade de tempo possível, com o objetivo de ganhar desempenho. Note-se que este desempenho pode ser alcançado; o problema está em que o escalonador, pela ausência de informações, atribua tarefas que (i) geram alto custo computacional ou que (ii) consumam grande quantidade de memória a máquinas que não possuem recursos suficientes para suprir essas necessidades.

WQ-R se comporta da mesma forma que WQ (por ser uma extensão), porém, durante o funcionamento, caso alguma máquina termine sua execução antes que as outras e não existir tarefas na fila de tarefas a serem executadas, o escalonador irá, por alguma política pré determinada, escolher uma tarefa que esteja em execução para replicá-la na máquina que está ociosa.

O DSC [YAN 94] é um algoritmo de escalonamento estático baseado em custos de processamento e comunicação. O DSC busca alcançar uma solução ótima para o problema de DAG's[1] (Directed Acyclic Graph) com valores arbitrários, explorando o caminho crítico das tarefas da aplicação.

A entrada do algoritmo é um grafo dirigido anotado, que descreve as tarefas que compõem a aplicação acompanhadas de seu custo de comunicação, computação e a relação de dependência com as outras tarefas. Essa relação dá origem ao DAG que á analisado de forma direta e inversa. A forma inversa de um DAG consiste em inverter as direções de todas as arestas de dependências, aplicar o algoritmo DSC e inverter-lo novamente para sua forma original. Após a análise, o algoritmo tenta identificar e reduzir o caminho crítico, denominado *DS* (seqüência dominante), pois assim será diminuido o tempo de execução paralela. A redução é alcançada através do agrupamento (*clustering*) das tarefas pertencentes ao DS em uma mesma máquina. Isto faz com que o custo de comunicação entre essas tarefas seja igual a zero e que o tamanho do DS seja reduzido a cada agrupamento. O agrupamento é realizado comparando o custo de comunicação das tarefas que estão sendo agrupadas com o custo que será agregado para a comunicação com tarefas que estão fora deste agrupamento. Se o custo de comunicação for maior do que a redução do DS o agrupamento é rejeitado, pois insere maiores custos do que minimização no DAG.

Extensão do algoritmo DSC

A extensão proposta não irá alterar o funcionamento do algoritmo DSC, pois irá trabalhar com a sua saída conhecido na literatura como *scheduled DAG*. Estão sendo cri-

[1]DAG é uma tupla G = (*V,E,C,T*), onde V é um conjunto de tarefas (nodos), E são as arestas dirigidas que representam a comunicação, C é o custo de comunicação das arestas e T, é o custo de computação das tarefas.

adas heurísticas que permitam escalonar as tarefas de forma eficiente. Podemos citar dois pontos importantes que estão sendo estudados na concepção da extensão do algoritmo.

1. Duplicação de Tarefas: A duplicação de tarefas baseada em conhecimento pode levar a obtenção de um melhor ganho de desempenho do que as abordagens tradicionais. Em [RAN 2000, CHO 2002] pode ser visto que algoritmos com duplicação sempre alcanç am melhores resultados que os algoritmos sem duplicação. Nosso estudo busca identificar situações onde a duplicação é viável. Uma situação favorável, se dá quando existe uma tarefa que possui um custo de comunicação muito elevado em comparação as suas tarefas sucessoras. Neste caso, podemos duplicar a tarefa para um ou mais nodos onde a comunicação é mais crítica. Reduzir o tempo de comunicação implica em diminuir o *makespan* (tempo de execução da última tarefa).

2. Recuperação de Resultados: A recuperação de resultados é, fora o processamento em sí, a parte mais importante em uma execução em larga escala. Nossa preocupação se concentra em determinar quais nodos que compõem a grade são candidatos a armazenarem resultados de outros nodos. A determinação desses nodos se dará através de um histórico de *reliability*[2].

Avaliação de desempenho

A implementação realizada é uma aplicação sintética semelhante à de seqüênciamento de DNA, formada por 9 tarefas, onde cada tarefa está formada pela tupla, custo de processamento e comunicação. As tarefas são: $T_1(1,25)$, $T_2(6,13)$, $T_3(5,13)$, $T_4(8,4)$, $T_5(5,4)$, $T_6(2,4)$, $T_7(2,3)$, $T_8(6,5)$ e $T_9(1,0)$. As avaliações foram realizadas com 9 máquinas Pentium IV de 1.8Ghz com 256MB de memória RAM (arquitetura monoprocessada) e rede Ethernet de 10 MBits utilizando *switchs*. As aplicações foram compiladas utilizando o GCC 3.3.5 sobre GNU/Linux. O objetivo destas avaliações é validar o uso do algoritmo DSC como base estudo frente aos outros algoritmos que são amplamente utilizados. Os tempos apresentados são uma média de 10 execuções. O resultado com o algoritmo DSC se encontra na Tabela 1.

Tabela 1: Tempos de execução com o algoritmo DSC

Clusters	Tarefas	Processamento	Comunicação
Cluster 0	1,2,3,4	40570 ms	0 ms
Cluster 1	6,8	16130 ms	402 ms
Cluster 2	5,9	12300 ms	690 ms
Cluster 3	7	4030 ms	349 ms

A medição com o algoritmo WQ-R foi realizada no ambiente Mygrid [COS 2004], onde as tarefas foram lanç adas na mesma ordem de execução do lanç amento com o algoritmo DSC, para serem comparáveis. O tempo total de execução no ambiente é de

[2]Probabilidade do nodo não falhar durante o perí odo de execução da aplicação.

131743 milisegundos, contra o tempo do algoritmo DSC que é de 74471 milisegundos. Este resultado justifica o o esforç o para implementar uma extensão ao DSC.

Conclusão

O ganho de desempenho que pode ser obtido com algoritmos de escalonamento justifica o estudo dos mesmos. Atualmente, comparados a outras àreas de escalonamento, algoritmos para computação em grade recebem pouca atenção. Isto se deve em parte ao primeiro objetivo de computação em grade, que consistia somente na execução de tarefas em larga escala. Atualmente, a abundância de recursos permite existem aplicações (médicas, biológicas e físicas) que podem dar contribuição à humanidade através de seus resultados. Isto exige que cientistas da computação criem mecanismos para transformar o ambiente de computação em grade em mais um ambiente de processamento de alto desempenho. Este trabalho apresenta o algoritmo DSC e uma breve descrição sobre a extensão que está sendo implementada sobre ele. Comparado aos algoritmos WQ e WQ-R – abordagens clássicas –, o DSC levou uma ampla vantagem por considerar informações tais como, o custo de comunicação e o custo de computação das tarefas . Isto mostra que estamos no caminho certo, para aplicar algoritmos baseados em conhecimento (*full-knowledge*) na computação em grade. O obejtivo final deste estudo é implementar esta extensão no ambiente MyGrid.

Referências

[AND 2003] ANDRADE, N. et al. Ourgrid: an approach to easily assemble grids with equitable resource sharing. **JSSP**, p.61–86, September 2003.

[CHO 2002] CHOE, T.-Y.; PARK, C. A task duplication based scheduling algorithm with optimality condition in hererogeneous systems. **ICPPW '02: Proceedings of International Conference on Parallel Processing Workshop**, p.531, 2002.

[CIR 2003] CIRNE, W. Grades computacionais: arquiteturas, tecnologias e aplicacoes. **ERAD 2003**, v.1, p.103–132, Janeiro 2003.

[COS 2004] COSTA, L. B. et al. Mygrid: a complete solution to running bag-of-tasks applications. **Proceeddings of Simposio Brasileiro de Redes de Computadores**, v.1, p.407, Maio 2004.

[RAN 2000] RANAWEERA, S.; AGRAWAL, D. P. A task duplication based algorithm for heterogeneous systems. **IPDPS '00: Proceedings of International Symposium on Parallel and Distributed Processing**, p.445, 2000.

[YAN 94] YANG, T.; GERASOULIS, A. Dsc: scheduling parallel tasks on an unbounded number of processors. **IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS**, v.5, p.951–967, September 1994.