Estudo Comparativo de Algoritmos de Escalonamento para Grades Computacionais

Alvaro H. Mamani-Aliaga, Alfredo Goldman

[1]Instituto de Matemática e Estatística
Universidade de São Paulo (USP)
Caixa Postal 05508-090 – São Paulo – SP – Brasil

{alvaroma,gold}@ime.usp.br

Abstract. *The study of scheduling in grid computing environments is very important, because, depending of a good schedule, the grid will achieve a good performance. In this paper we study three algorithms: (a)* HEFT*, (b)* CPOP *e (c)* PCH*. Each algorithm has good performance in certain circumstances. Each algorithm is evaluated in two different applications, workflow Montage and workflow Epigenomics. The HEFT and CPOP perform well in both applications, and the PCH algorithm performs well in application more parallelizable.*

Resumo. *O estudo do escalonamento em ambientes de computação em grade é muito importante, pois, dependendo de um bom escalonamento, a grade atingirá um bom desempenho. Neste artigo é feito um estudo de três algoritmos: (a)* HEFT*, (b)* CPOP *e (c)* PCH*. Cada algoritmo tem boa performance em determinadas circunstâncias. Cada algoritmo é avaliado em duas aplicações diferentes, o workflow Montage e o workflow Epigenomics. O algoritmo* HEFT *e* CPOP *apresentam bom desempenho em ambas aplicações, já o algoritmo* PCH *apresenta bom desempenho na aplicação mais paralelizável.*

1. Introdução

Nas últimas décadas, disponibilidade de computadores poderosos tem aumentado enormemente, assim como a sua interligação com redes de alta velocidade. Estes fatos têm permitido a agregação de recursos sem importar a distância entre eles para obter grande capacidade no processamento. Esta agregação de recursos geograficamente dispersos tem sido chamada, "Computação em Grade" (*Grid Computing*) [Foster et al. 2002], uma alternativa para obter grande capacidade de processamento. Na computação em grade, os recursos computacionais são heterogêneos e podem ser agregados ou retirados do ambiente em qualquer momento. Neste cenário o "escalonamento" (*scheduling* em inglês) é um grande desafio, que tem como objetivo principal atingir um bom desempenho no tempo de termino na execução de aplicações independentemente do tipo destas. O problema do escalonamento de tarefas é importante para ser estudado, consiste em alocar tarefas de uma aplicação em recursos computacionais, com o intuito de atingir um bom desempenho. Um dos objetivos principais nos algoritmos de escalonamento é minimizar o tempo de término das aplicações (*Makespan*. No processo do escalonamento, a aplicação pode ser de dois tipos: (a) aplicação com tarefas dependentes e (b) aplicação com tarefas independentes. O escopo do presente trabalho é focado nas aplicações com tarefas dependentes. Este tipo de aplicações são modeladas por um grafo acíclico dirigido (*Directed Acyclic Graphs, DAGs*) $G = (V, E)$, onde, G representa o DAG, V o conjunto de tarefas t e E o conjunto de dependências entre cada tarefa da aplicação.

2. Metodologia

2.1. Algoritmos de Escalonamento

Infelizmente, algoritmos de escalonamento de sistemas paralelos e distribuídos tradicionais, como é o caso da estratégia FIFO, podem não ser bem adaptados em ambientes de grade. Os algoritmos avaliados neste trabalho são descritos a seguir.

O algoritmo *Heterogeneous Earliest Finish Time* (**HEFT**) [Topcuouglu et al. 2002] possui duas fases. Na fase de *Priorização de tarefas* é calculada uma "prioridade" para cada tarefa baseada na média dos custos de computação e custos de comunicação. Depois de calcular as prioridades é gerada uma lista das tarefas em ordem não crescente (ordem topológica), que é uma ordem linear que preserva as restrições de precedência. Na fase de *seleção de recursos*, é onde a tarefa t_i é alocada ao recurso r_j, que minimize o seu tempo de término (*earliest finish time*).

O algoritmo *Critical Path On a Processor* (**CPOP**) [Topcuouglu et al. 2002], possui duas fases. Na fase de *priorização de tarefas* é calculada usando a média de computação e a média de comunicação. São selecionadas as tarefas que pertencem ao caminho crítico as quais tem igual prioridade que a tarefa de entrada. Na fase de *seleção de recursos* é selecionado um recurso de caminho crítico (*critical-path processor*), o que minimiza os custos de cálculo cumulativo de tarefas no caminho crítico. Se a tarefa selecionada está no caminho crítico, então é escalonada no recurso de caminho crítico, caso contrário, ela é atribuída a um recurso que minimiza o tempo de termino.

O algoritmo *Path Clustering Heuristic* (**PCH**) [Bittencourt et al. 2006] utiliza as técnicas heurísticas: *list scheduling* e *clustering*. Este algoritmo agrupa as tarefas formando *clusters*. Na fase de *seleção de tarefas e agrupamento* são criados os *clusters* de tarefas, baseados no cálculo das prioridades e tempos de termino. Na fase de *seleção de recursos* é alocado cada *cluster*, gerado na fase anterior, no recurso que oferece o melhor tempo de termino do conjunto de tarefas que pertencem ao *cluster*.

2.2. Workloads

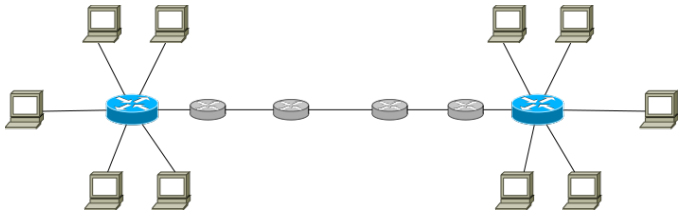
Os *workloads* usados para serem simulados neste trabalho são de aplicações paralelas reais (com tarefas dependentes). No trabalho de Shishir Bharathi *et. al.* [Bharathi et al. 2008]. Os workloads escolhidos para serem simulados, serão descritos a seguir:

O *Workflow* Montage: o projeto *Montage* foi criado pela NASA/IPAC e está disponibilizado como software livre. A aplicação *Montage* é usada para gerar mosaicos personalizados do céu usando pontos de múltiplas imagens de entrada no formato *Flexible Image Transport System*.

O *Workflow* Epigenomics: o projeto *Epigenomics* foi criado pelo centro do genoma da *University Southern California (USC)* e pelo *Pegasus Team*. Atualmente o centro do genoma da *USC* está envolvido no mapeamento do estado epigenético de células humanas sobre uma grande escala genômica.

3. Resultados Experimentais

Para simular os algoritmos, foi utilizado o simulador SimGrid [Casanova et al. 2008]. Também foram especificados dois aglomerados de cinco máquinas cada um, representado na Figura 1(a). Na 1(b) é mostrado o poder de processamento de cada recurso computacional do aglomerado. Foram implementados além dos algoritmos de escalonamento *HEFT*, *CPOP* e *PCH*, um escalonamento de tipo FIFO. Na estrutura do *workflow Montage* existem

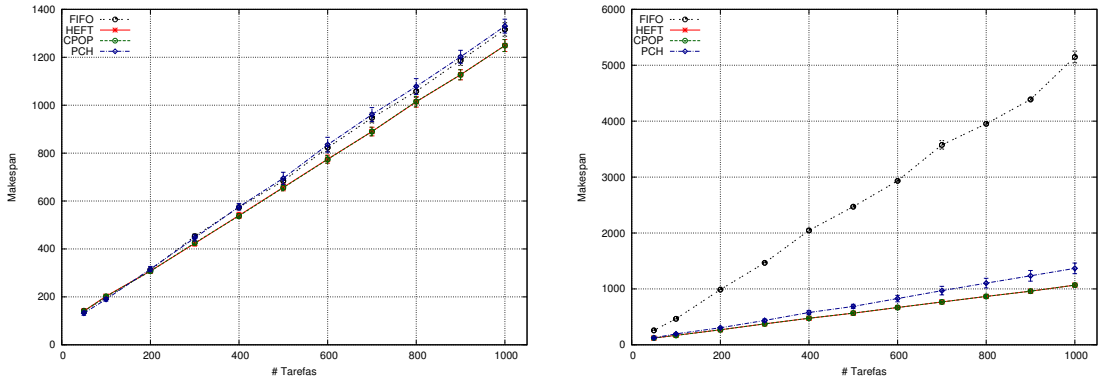


(a) Representação do aglomerado usado nos experimentos, cada roteador azul representa ao aglomerado e a interligação com o outro.

Id	Poder Computacional (*GFlops*)	
	Homogêneo	Heterogêneo
C1-00	5,00	1,00
C1-01	5,00	2,00
C1-02	5,00	3,00
C1-03	5,00	4,00
C1-04	5,00	5,00
C2-05	5,00	6,00
C2-06	5,00	7,00
C2-07	5,00	8,00
C2-08	5,00	9,00
C2-09	5,00	9,00

(b) Tabela de poderes computacionais dos recursos computacionais dos recursos

Figure 1. Representação dos recursos computacionais.



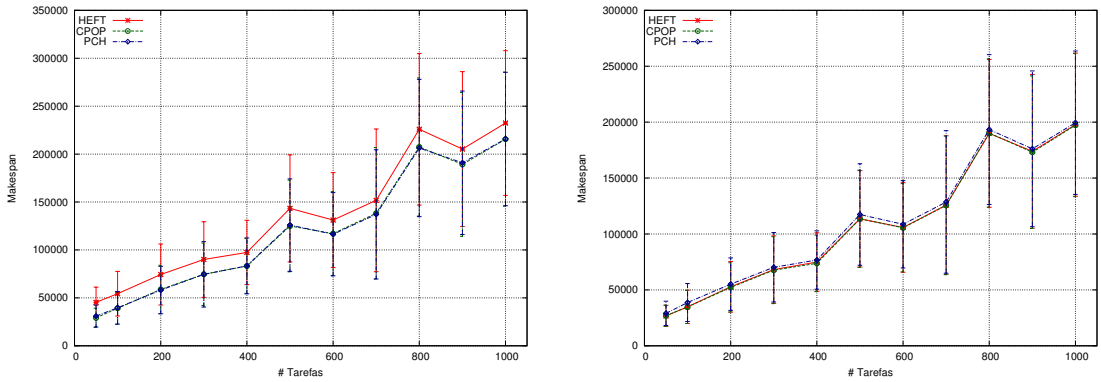
(a) Aglomerado homogêneo

(b) Aglomerado heterogêneo

Figure 2. Resultado da simulação do *Workload Montage* com as estratégias: *HEFT*, *CPOP*, *PCH* e o tipo de escalonamento *FIFO*.

dois gargalos, o qual dificulta o tempo de finalização da aplicação. Enquanto o *workflow Epigenomics* as tarefas são mais paralelizáveis. Como podemos observar na Figura 2(a), no aglomerado homogêneo, o comportamento da estratégia *FIFO* é próximo aos outros algoritmos. Este comportamento é aceito, desde que a arquitetura dos recursos computacionais seja homogênea, enquanto o algoritmo *PCH* não oferece um desempenho tão bom em comparação com os algoritmos *HEFT* e o *CPOP*, ele apresenta um melhor desempenho para aplicações com cinquenta e cem tarefas, melhorando em uma média de 7%, mas, para um número de tarefas maior que cem, o desempenho do algoritmo piora em uma média de 6%. Na Figura 2(b), o comportamento da estratégia *FIFO* não apresentou um bom desempenho, a respeito dos outros algoritmos. Podemos ver que em arquiteturas heterogêneas o uso de uma estratégia de tipo *FIFO*, não é adequado, enquanto, o algoritmo *PCH* não apresenta um desempenho bom, comparado com o *HEFT* e *CPOP*. Para um número de cinquenta tarefas o algoritmo é 2% pior dos outros, já para aplicações maiores, neste caso de cem até mil tarefas, o algoritmo piora desde 15% até 28%. Em ambas as arquiteturas, tanto o algoritmo *HEFT* quanto o *CPOP*, apresentam um desempenho similar, cuja diferencia entre eles são apenas décimas.

Na Figura 3(a) o algoritmo *PCH* melhora consideravelmente o desempenho em comparação com o algoritmo *HEFT*. O algoritmo *CPOP* conserva um bom desempenho para este tipo de aplicações, para uma arquitetura homogênea. Enquanto para uma arquitetura heterogênea os algoritmos estão próximos. O algoritmo *PCH* é 3% pior, enquanto os algoritmos *HEFT* e *CPOP* possuem uma diferencia apenas de décimas.



(a) Aglomerado homogêneo (b) Aglomerado heterogêneo

Figure 3. Resultado da simulação do *Workload Epigenomics*.

4. Conclusões

Para aplicações com tarefas dependentes, uma estratégia de tipo *FIFO* é adequada somente em arquiteturas homogêneo. Já em arquiteturas heterogêneas é preciso um estudo dos recursos computacionais e da aplicação para obter um desempenho aceitável no escalonamento. Os algoritmos *HEFT* e *CPOP* apresentaram bom e similar desempenho com a aplicação *Montage*, enquanto o algoritmo *PCH* não, este fato é porque na aplicação *Montage* existem gargalos, onde um agrupamento das tarefas não é adequado. Por outro lado, em aplicações altamente paralelizáveis, o algoritmo *PCH* apresenta melhor desempenho, pois neste tipo de aplicações o algoritmo faz uso melhor dos recursos, agrupando as tarefas em *clusters*. Como extensão para o presente trabalho é almejado a simulação dos algoritmos em arquiteturas reais, por exemplo, DAS-3, Grid5000, GridPP, entre outros.

5. Agradecimentos

Os autores agradecem ao CNPq pelo apoio financeiro.

References

Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.-H., and Vahi, K. (2008). Characterization of scientific workflows. *The 3rd Workshop on Workflows in Support of Large-Scale Science (WORKS08)*.

Bittencourt, L. F., Madeira, E. R. M., Cicerre, F. R. L., and Buzato, L. E. (2006). Uma heuristica de agrupamento de caminhos para escalonamento de tarefas em grades computacionais. In *Anais SBRC 2006*, pages 83–98, Curitiba, Brazil.

Casanova, H., Legrand, A., and Quinson, M. (2008). SimGrid: a Generic Framework for Large-Scale Distributed Experiments. In *10th IEEE International Conference on Computer Modeling and Simulation*. IEEE Computer Society Press.

Foster, I., Kesselman, C., Nick, J. M., and Tuecke, S. (2002). Grid services for distributed system integration. *Computer*, 35(6):37–46.

Topcuouglu, H., Hariri, S., and Wu, M.-y. (2002). Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *IEEE Trans. Parallel Distrib. Syst.*, 13(3):260–274.