Análise Comparativa da Eficiência Energética de Algoritmos para Escalonamento de Jobs em Grades Computacionais

Gustavo B. da Silva, Lucas C. Casagrande, Guilherme P. Koslovski, Maurício A. Pillon

[1]LabP2D – Universidade do Estado de Santa Catarina (UDESC)

Resumo. *Juntamente à crescente demanda de recursos computacionais, é notável o crescimento do consumo de energia de data centers. Utilizando a abordagem de grades, este trabalho busca demonstrar o consumo de energia em grades com diferentes perfis de carga de trabalho. Via simulação, os resultados demonstram que as decisões de cada política possuem um impacto significativo no consumo de energia. Logo, é possível concluir que o consumo de energia é um aspecto importante que não deve ser negligenciado durante a adoção de uma política de escalonamento em um ambiente em produção.*

1. Introdução

Um dos maiores problemas no ramo da computação é a alta demanda de recursos, visto que problemas computacionais complexos exigem muito das arquiteturas dos sistemas. Outras áreas, além da informática, também dependem do poder computacional para concluir seus projetos, como as ciências exatas, biológicas, humanas, entre outras. Com o avanço das tecnologias que integram a arquitetura de computadores, abordagens distribuídas e paralelas obtiveram novas concepções de utilização. Sucessivamente, arquiteturas como *clusters*, grades computacionais e nuvens começaram a ser relacionadas com a aplicação de aprendizado de máquina, sistemas escalonadores de tarefas e a análise de processamento para a economia de energia.

Especificamente, as grades computacionais permitem a divisão de tarefas entre diversos servidores, seja em um sistema local ou distribuído. O termo conhecido como computação em grade demonstra a junção de recursos distribuídos de forma geográfica como servidores de armazenamento, redes de alto desempenho, supercomputadores, banco de dados e aglomerados que buscam solucionar problemas de larga escala. Existem diversos projetos focados na utilização de grades computacionais para a resolução de problemas como a *Grid 5000*, *IBM World Community*, *NASA information Power Grid* (*IPG*) e a *NCSA GridTech*. Tais projetos são plataformas de grande escala, possibilitando o experimento de novas tecnologias e teorias científicas.

Um ambiente de grade é capaz de receber diversas requisições para alocação de recursos, sendo esses recursos compartilhados, realocados e reutilizados inúmeras vezes em uma curta janela de tempo, como acontece com a *Grid 5000* por exemplo. Em sistemas assim, é necessário que exista um sistema escalonador de tarefas coeso, capaz de compreender as requisições dos usuários e adaptá-las à infraestrutura real da grade.

O alto consumo energético é um dos principais pontos de estudo de *data centers*, visto que tal consumo é responsável por maior parte do custo operacional dos mesmos, fica claro que além de afetar diretamente no custo operacional, também existem questões ecológicas para incentivar o estudo sobre eficiência energética. Logo, a fim de demonstrar

o consumo de energia de algoritmos de escalonamento para tarefas em grades computacionais, esse artigo apresenta o estudo de cargas e consumos com diferentes algoritmos, utilizando o ambiente de simulação Batsim [Dutot et al. 2015].

2. Algoritmos de Escalonamento

Inicialmente, os tipos e classificações de grades são revisados, posteriormente indicando quais algoritmos para escalonamento de tarefas podem ser aplicados.

2.1. Tipos e Classificações de Grades

Existem diferentes tipos de grades, e cada qual possui uma finalidade e áreas de aplicação:

- Grades computacionais: são compartilhados ciclos de processamento, sendo divididos em três formas distintas de explorar os recursos computacionais da grade. Por primeiro, é executado uma aplicação em uma máquina disponível da grade, independentemente do seu respectivo local. No segundo, a aplicação é dividida em partes menores, para possibilitar o paralelismo pela grade. Por fim, o terceiro método é baseado na execução de tarefas que necessitam rodar diversas vezes em máquinas diferentes dessa mesma grade.
- Grades de dados: o espaço de armazenamento de cada máquina é compartilhado pela grade, o que aumenta a capacidade de armazenamento total, a eficiência e confiabilidade dos dados. Essa abordagem utiliza todo o espaço do sistema de arquivos como único para toda a grade, abstraindo a localização de arquivos, mesmo que estejam divididos em partes entre as máquinas.
- Grades de rede: oferecem serviços de comunicação tolerante a falhas e que mantenham alto desempenho. É possível utilizar máquinas com conexões ociosas a fim de transmitir parte de dados ou regular redundância nas transmissões.

2.2. Algoritmos de Escalonamento de Tarefas

Considerando que a abordagem principal deste trabalho possui o foco em grades computacionais, os algoritmos são fundamentais na escolha de um escalonador de tarefas. Logo, serão listados algoritmos relevantes para o presente trabalho:

- *First Fit*: No algoritmo de escalonamento *First Fit* o processo é alocado na primeira lacuna disponível que possua recursos suficientes para executar o mesmo.
- *Easy BackFilling (EASY)*: é uma otimização do algoritmo *First Come First Served* (FCFS), visto que procura balancear as metas de utilização e persistir o pedido do FCFS. Também é necessário que seja especificado o tempo máximo de execução de cada tarefa. Dessa maneira, é possível que outras tarefas menores sejam escalonadas, enquanto a tarefa na cabeça da fila está esperando, principalmente se não afetarem o início da tarefa que está na cabeça da fila. É importante notar que existem variantes dessa abordagem [Lelong et al. 2017].
- *Conservative Backfilling*: o algoritmo de escalonamento *Conservative Backfilling* (CBF) é uma alternativa menos agressiva do que o EASY com um desempenho similar. O algoritmo determina a alocação para cada tarefa quando a mesma entra no sistema. Após isso, outras tarefas podem ser alocadas se, e somente se, for possível começar sua execução imediatamente sem atrasar as outras tarefas pré-alocadas [N'takpé and Suter 2017].

- *Packer*: algoritmo utilizado no escalonador aglomerado TETRIS, diferente da maioria dos algoritmos de escalonamento que se atentam apenas ao uso de CPU e de memória das máquinas, empacota tarefas de acordo com múltiplos recursos. Para determinar o recurso com mais impacto em cada tarefa, o TETRIS aprende adaptativamente os requisitos de tarefas enquanto monitora os recursos disponíveis nas máquinas. Assim, a heurística de empacotamento projeta as duas informações em um espaço euclidiano, selecionando o par (tarefa, máquina) com o produto escalar de maior valor. Além disso, o TETRIS utiliza uma versão multi-recurso do algoritmo *Shortest Remaining Job Time* para trabalhos que são grafos acíclicos direcionados (GADs) de tarefas dependentes e combinam ambas as heurísticas, *Packer* e *Shortest Remaining Job Time*, para reduzir o tempo médio da conclusão dos trabalhos [Grandl et al. 2015].

3. Experimentos e Resultados

Como foco deste trabalho, temos a análise do consumo energético de grades computacionais escalonadas de modos divergentes. Logo, para avaliar o consumo de energia com diferentes políticas de escalonamento, foi realizada uma simulação com o Batsim em torno de um cenário de testes gerado para tal. Em resumo, o Batsim é um simulador de código aberto que permite simular comportamentos de plataformas computacionais em que uma carga de trabalho é balanceada de acordo com um algoritmo de escalonamento que foi desenvolvido por [Dutot et al. 2015].

3.1. Cenários de Testes

Para analisar o consumo de energia, foram elaborados quatro perfis de carga de trabalho, com tamanho de trabalhos variados. Cada perfil possui um porcentual de trabalhos pequenos com tempo de execução entre [1, 5] unidades de tempo e de trabalhos grandes com tempo de execução entre [15, 20]. Esses tempos de execução, são distribuídos de forma uniforme pelos intervalos. Já a quantidade de recursos solicitada por cada trabalho é definida a partir de uma distribuição uniforme entre os intervalos [1,5] e [5,10]. O intervalo de chegada segue um processo de Bernoulli com 40% de probabilidade de um *job* ser submetido em cada unidade de tempo e além disso, o tempo máximo de submissão dos trabalhos é definido em 1000 segundos, o qual gera uma média de 400 trabalhos por carga de trabalho. Por fim, para cada perfil são geradas 10 cargas distintas.

3.2. Resultados

Na Figura 1 é demonstrado a média do consumo de energia, em joules, junto ao desvio padrão para cada uma das políticas avaliadas considerando todas os perfis de carga de trabalho. Note que o percentual indica a proporção de trabalhos pequenos presentes na carga de trabalho.

Analisando a Figura 1, constata-se que o *Conservative Backfilling* (CBF) teve o menor consumo de energia em todas os perfis de carga de trabalho. Houve uma redução de, aproximadamente, 16% em relação ao *First Fit* e de 10% quando comparado ao Packer, considerando um perfil de carga composta por 20% de *jobs* pequenos. Vale salientar que, com exceção do *First Fit*, esta diferença diminui em conjunto com a redução na utilização no sistema em virtude da menor presença de *jobs* longos. Logo, considerando o perfil de carga com 80% de *jobs* pequenos, não houve um impacto significativo no

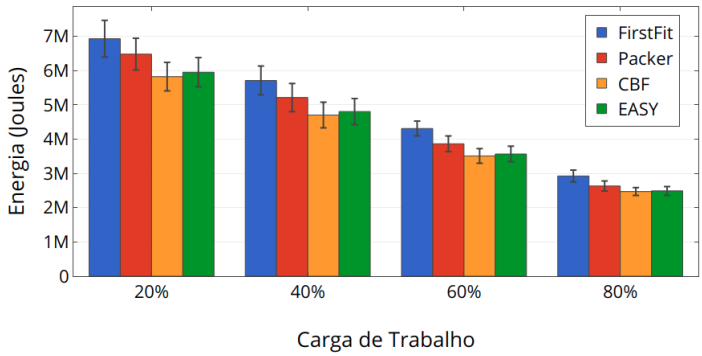

Figura 1. Consumo energético com diferentes políticas de escalonamento.

consumo de energia entre as políticas. Já o *Easy Backfilling* (EASY) apresenta um comportamento similar ao CBF e não houve uma diferença significativa em relação as duas.

4. Considerações Finais

Um escalonador de tarefas deve ser consistente de modo que seja adaptável a infraestrutura da grade. Embora o crescimento do consumo energético seja inevitável à medida que aumenta a demanda de recursos, é possível utilizar abordagens que diminuam este consumo para diferentes tipos de cargas de trabalho.

Agradecimentos: Os autores agradecem o apoio do Laboratório de Processamento Paralelo Distribuído (LabP2D) da Universidade do Estado de Santa Catarina (UDESC) e a Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC).

Referências

Dutot, P.-F., Mercier, M., Poquet, M., and Richard, O. (2015). Batsim: a realistic language-independent resources and jobs management systems simulator. In *Job Scheduling Strategies for Parallel Processing*, pages 178–197. Springer.

Grandl, R., Ananthanarayanan, G., Kandula, S., Rao, S., and Akella, A. (2015). Multi-resource packing for cluster schedulers. *ACM SIGCOMM Computer Communication Review*, 44(4):455–466.

Lelong, J., Reis, V., and Trystram, D. (2017). Tuning easy-backfilling queues. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 43–61. Springer.

N'takpé, T. and Suter, F. (2017). Don't hurry be happy: A deadline-based backfilling approach. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82. Springer.