Encontrando Políticas de Escalonamento Eficientes para *Clusters* de Alto Desempenho com Simulação e Regressão Não Linear

Danilo Carastan-Santos[1], **Raphael Y. de Camargo**[1]

[1]Centro de Matemática, Computação e Cognição
Universidade Federal do ABC (UFABC)
Santo André – SP – Brasil

{danilo.santos,raphael.camargo}@ufabc.edu.br

Resumo. *Desenvolvemos uma metodologia para obtenção de políticas simples e eficientes na forma de funções não lineares para o escalonamento de tarefas em clusters de alto desempenho, por meio da utilização de simulações e regressão não linear. Resultados experimentais mostraram que as funções obtidas pelo nosso método aprimoraram o desempenho do escalonamento em até 7,25 vezes em tarefas de um modelo de workload. Estes resultados indicam que nossa metodologia é promissora, com futuro potencial de aprimoramento no desempenho do escalonamento em cenários reais.*

1. Introdução

O escalonamento dinâmico de tarefas em *clusters* de alto desempenho consiste em selecionar tarefas para execução de modo que algum critério de desempenho seja minimizado. Ao levar em consideração as diversas características das tarefas e dos *clusters*, efetuar essa seleção de tarefas torna-se um problema particularmente complexo. Deveras, encontrar escalonamentos ótimos em *clusters* é sabidamente um problema NP-Difícil [Garey and Johnson 1979] e a obrigatoriedade de se encontrar escalonamentos rapidamente torna a obtenção desses escalonamentos ótimos ainda mais inviável. Em busca de soluções alternativas, diversos sistemas utilizam políticas de escalonamento baseadas em funções utilidade [Tang et al. 2009]. Embora essas funções não gerem escalonamentos ótimos, elas trazem a vantagem de serem simples e de fácil processamento. Este trabalho propõe uma metodologia para obtenção de políticas de escalonamento na forma de funções utilidade não lineares para o escalonamento eficiente de *clusters* de alto desempenho. Resultados experimentais mostraram que nossa abordagem é capaz de obter politicas de escalonamento que melhoram significativamente o desempenho do mesmo.

O restante desse texto está estruturado como segue. Na Seção 2 é apresentada uma breve descrição das principais características da nossa abordagem. Na Seção 3 são mostrados alguns resultados experimentais preliminares. Por fim, na Seção 4 são levantadas as conclusões e as perspectivas de trabalhos futuros.

2. Metodologia para Obtenção de Políticas de Escalonamento

Objetivamos a obtenção de políticas de estalonamento de tarefas baseadas em três fatores: tempo de execução (r), quantidade de processadores requeridos (n) e tempo de chegada (s). A metodologia proposta inicia-se com a obtenção de informações por via

de simulações sobre como que o desempenho do escalonamento é afetado quando uma tarefa t é escolhida para execução, em função dos parâmetros r_t, n_t e s_t. Nesse passo, consideramos um cenário de simulação que consiste em um *cluster* homogêneo com 256 processadores e duas filas de tarefas S e Q. A fila S possui 16 tarefas que são escalonadas em qualquer ordem e são executadas no começo da simulação. O objetivo das tarefas em S consiste em representar realisticamente um estado inicial dos recursos do cluster, fazendo com que as tarefas da fila Q não sejam escalonadas em um *cluster* inicialmente vazio. A fila Q, por sua vez, possui 32 tarefas as quais são executadas logo após a fila S. As tarefas em Q são as tarefas nas quais extraímos os dados de desempenho do escalonamento. Utilizamos a métrica de desempenho denominada *Average Bounded Slowdown* (Equação 1), onde w_i representa o tempo que a tarefa i esperou para ser executada e τ é uma constante que previne valores de *slowdown* muito grandes para tarefas muito pequenas e na literatura geralmente lhe é atribuído o valor de 10 segundos.

$$AVEblsd(Q) = \frac{1}{|Q|} \sum_i \max\left(\frac{w_i + r_i}{\max(r_i, \tau)}, 1\right) \tag{1}$$

Para as tarefas em S e Q foram atribuitas tarefas geradas pelo modelo de *workload* de Lublin e Feitelson [Lublin and Feitelson 2003] que incorpora características de tarefas de diversos *clusters* reais e utilizamos o *software* de simulação SimGrid [Casanova et al. 2014] para efetuar as simulações. Foram efetuadas diversas simulações com pares (S, Q) distintos com o objetivo de capturar o comportamento das tarefas em diversos cenários de escalonamento. Para cada simulação sobre um par (S, Q), definimos como $\mathcal{P}$ uma coleção contendo permutações aleatórias das tarefas de Q. Para cada permutação $p \in \mathcal{P}$ também definimos $t_0(p)$ que denota a primeira tarefa da permutação p. Em seguida simulamos o escalonamento para todas as permutações $p \in \mathcal{P}$ e, ao fim das simulações, utilizamos as Equações 2 e 3 para determinar um *score* para as tarefas $t \in Q$. Esse *score* denota o impacto da tarefa t quando esta foi selecionada para ser a primeira a ser executada e, quanto menor o valor desse *score*, melhor é a escolha de t como primeira tarefa. Experimentos preliminares mostraram que para uma fila Q contendo 32 tarefas, 250 mil permutações aleatórias de Q proporcionam um bom equilíbrio entre qualidade dos resultados das simulações e tempo de execução das mesmas.

$$score(t_i) = \frac{\sum_{p_j \in \mathcal{P}} AVEblsd(p_j, t_i)}{\sum_{p_k \in \mathcal{P}} AVEblsd(p_k)} \tag{2}$$

$$AVEblsd(p, t) = \begin{cases} AVEblsd(p), & \text{se } t_0(p) = t \\ 0, & \text{caso contrário} \end{cases} \tag{3}$$

Ao fim das simulações sobre um par (S, Q), obtemos um conjunto composto por uma tripla de entrada (r_t, n_t, s_t) e um rótulo $score(t)$ para todas as tarefas $t \in Q$. A união dos conjuntos de cada par (S, Q) é utilizada como conjunto de treinamento em um algoritmo de regressão não linear. Esse algoritmo utiliza o método dos mínimos quadrados para ajustar uma função não linear $f(r, n, s)$ aos *scores* obtidos nas simulações. A regressão foi efetuada levando em consideração diversas funções não lineares. As melhores funções – seguindo a métrica de distância absoluta média entre os *scores* e a

Tabela 1. Melhores funções utilidade obtidas pela regressão não linear.

ID	Função Utilidade	c_1	c_2	c_3
F1	$c_1\sqrt{r} \cdot c_2 n + c_3 \log s$	0.0001352	0,2765352	0.0071470
F2	$c_1 r + c_2\sqrt{n} + c_3 \log s$	0.0000002	0.0003207	0.0069554

saída das funções – foram testadas em um ambiente de simulação refletindo um cenário real. Detalhes sobre as melhores funções encontradas e sobre o cenário real de simulação são apresentados na seção a seguir.

3. Resultados Experimentais

A Tabela 1 mostra duas das funções que obtiveram o melhor resultado no processo de regressão não linear. Para avaliar essas funções como políticas de escalonamento, simulações imitando um ambiente real de escalonamento foram efetuadas, onde as tarefas são submetidas continuamente por um determinado período de tempo e o escalonamento é re-efetuado sempre que um recurso é liberado ou quando uma nova tarefa é submetida. Como medida de comparação, utilizamos adicionalmente a política de escalonamento FCFS e duas políticas de escalonamento de *clusters* conhecidas [Tang et al. 2009]. A Figura 1 mostra os desempenhos das políticas de escalonamento com um trace gerado pelo modelo de *workload* de Lublin e Feitelson [Lublin and Feitelson 2003] – o qual é o mesmo modelo utilizado na geração dos dados de treinamento – simulando um período de 30 dias de submissão de tarefas e em um cluster simulado com 256 processadores. Podemos observar que o procedimento para capturar impacto do desempenho do escalonamento ao selecionar diferentes tarefas para execução obteve dados suficientes para que as funções não lineares ajustadas a esses dados consigam gerar bons escalonamentos. Em comparação com o melhor desempenho das políticas de escalonamento selecionadas para comparação, obtivemos ganhos de desempenho de até 7,25x. As funções e seus coeficientes obtidos pela regressão não linear devem se ajustar excepcionalmente às características das tarefas e à configuração de *cluster* utilizada no treinamento. Ao utilizar esta mesma configuração nos experimentos de teste, um alto ganho de desempenho é de fato esperado. Outra observação importante é que a função F2, embora possua um desempenho melhor que as políticas de escalonamento utilizadas como comparação, apresentou um desempenho consideravelmente menor que a função F1. Um estudo mais detalhado sobre as funções e seus comportamentos como políticas de escalonamento é necessário para compreendermos melhor este fenômeno.

4. Conclusão

Neste trabalho apresentamos uma metodologia para obtenção de políticas simples de escalonamento na forma de funções utilidade não lineares para *clusters* de alto desempenho. Funções utilidade apresentam uma vantagem ao serem utilizadas como políticas de escalonamento devido a sua simplicidade e mostramos que é possível obter tais funções utilizando uma metodologia de simulação e regressão não linear igualmente simples. Além disso, mostramos também que as políticas obtidas pelo nosso método podem melhorar significativamente o desempenho do escalonamento, com desempenhos de até 7,25 vezes em um ambiente com tarefas de um modelo de *workload*, sinalizando portanto um potencial de melhoria de desempenho em cenários reais. Ainda há muito o que explorar

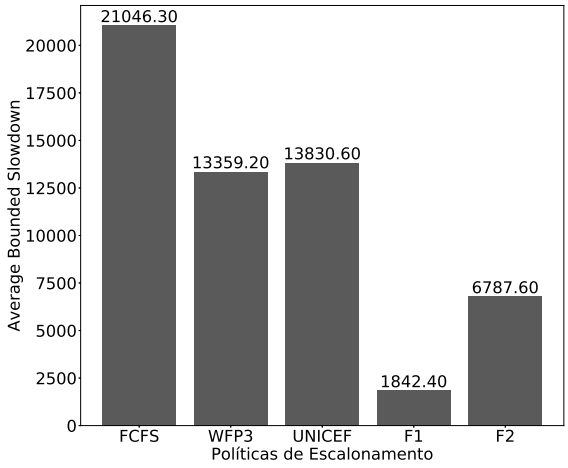

Figura 1. Desempenho das políticas de escalonamento.

sobre esse assunto, entretanto. O procedimento de simulação para a geração dos dados de treinamento apresentado (ver Seção 2) utiliza apenas informações locais sobre a primeira tarefa a ser selecionada. É possível que utilizando informações globais consigamos ganhos de desempenho ainda melhores, mas outras técnicas de aprendizado de máquina podem ser necessárias para esse caso como, por exemplo, redes neurais. Além disso, pretendemos fazer um estudo detalhado das funções obtidas pela regressão não linear para compreendermos melhor as diferenças de desempenho observadas entre essas funções e também para determinar o quanto essas funções são capazes de melhorar o desempenho do escalonamento em outras configurações de tarefas e *clusters*. Por fim, pretendemos estudar a possibilidade da adição de *backfilling* [Mu'alem and Feitelson 2001] de tarefas ao escalonamento gerado pelas funções e estudar o comportamento resultante.

Referências

[Casanova et al. 2014] Casanova, H., Giersch, A., Legrand, A., Quinson, M., and Suter, F. (2014). Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917.

[Garey and Johnson 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

[Lublin and Feitelson 2003] Lublin, U. and Feitelson, D. G. (2003). The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122.

[Mu'alem and Feitelson 2001] Mu'alem, A. W. and Feitelson, D. G. (2001). Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543.

[Tang et al. 2009] Tang, W., Lan, Z., Desai, N., and Buettner, D. (2009). Fault-aware, utility-based job scheduling on BlueGene/P systems. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–10. IEEE.