BUC: Beneficiador de Usuários Conscientes em Grades

Geremias Corrêa[1] **e Maurício A. Pillon**[1]

[1]Departamento de Ciência da Computação (DCC)
Universidade do Estado de Santa Catarina (UDESC)
Joinville, SC – Brasil

geremias.correa@edu.udesc.br, mauricio.pillon@udesc.br

Abstract. *With the growth of grid computing, having priorities that vary according to the context and, therefore, in the scheduling algorithms that operate in it, elaborating optimizations that maximize the quality of the resources disposition and minimize the unwanted outgoings is essential. With this, a model is built based on the users confindence level, based on an algorithm that reposition the queues according to this information, helping the precise users at the expense of the non-precise ones. It also seeks, through this, to make users aware in sending more accurate information. The results obtained have a positive return on reducing the waiting time of these more accurate users by up to 93%*

Resumo. *Com o crescimento da computação em grade, possuindo prioridades que variam conforme o contexto e, portanto, nos algoritmos de escalonamento que nela atuam, elaborar otimizações que maximizem a qualidade da disposição dos recursos e minimizem os gastos indesejados é essencial. Com isso, é construído um modelo baseado no nível de confiança do usuário, a partir de um algoritmo que reposiciona as filas de acordo com essa informação, ajudando os usuários precisos em detrimento dos não precisos. Busca-se também, através disso, conscientizar os usuários no envio de informações mais acertadas. Os resultados obtidos tem um retorno positivo de redução do tempo de espera desses usuários mais precisos em até 93%*

1. Introdução

A busca da otimização e diversidade na disponibilização de recursos computacionais tem crescido de forma veemente nas últimas décadas. Grades computacionais surgem nos anos 1990 [Goes et al. 2005], porém ferramentas e algoritmos associados a área ainda são grande temas de pesquisa [Casagrande et al. 2020, Pillon et al. 2020, Eng et al. 2020]. Neste trabalho, é buscado o reposicionamento de tarefas de usuários nestas grades, visando o favorecimento de usuários preocupados com a precisão das informações submetidas ao *Resource Management System* (RMS) [Alves et al. 2013, Dubey et al. 2018].

A imprecisão das informações enviadas pelos usuários no momento da solicitação de recursos pode afetar os critérios de tomada de decisão do algoritmo de escalonamento e permitir o provisionamento inadequado de recursos. Os algoritmos existentes diferem na tomada de decisão e métricas de análise. Dentre as métricas, este trabalho focou-se na análise de comportamento do uso de *Walltime* [Poquet 2017], tempo de espera para tomada de decisão e identificação do uso de consciência da solicitação do usuário.

A principal contribuição deste trabalho está na concepção de um modelo de classificação de tarefas baseado no nível de confiança do usuário. O Algoritmo **Beneficiador do Usuário Consciente** (BUC) permite o favorecimento de tarefas, no posicionamento de fila de espera, de usuários que tenham históricos de requisições com informações precisas, em detrimento de outros usuários, estes despreocupados com os dados informados no momento da requisição. Os resultados preliminares obtidos demonstram a eficiência do algoritmo, proporcionando acréscimo de até 20% no tempo de espera, para usuários com grau de precisão próximos a zero ($P \sim 0$) e, redução, de até 93%, para aqueles que aproximam-se a precisões $P = 1$.

O trabalho se encontra disposto nas seguintes seções, a Seção 2, referencial teórico, a Seção 3, descrição do algoritmo, a Seção 4, experimentação e análise dos resultados e, por último, a Seção 5, considerações finais.

2. Grades Computacionais e Algoritmos de Escalonamento

A ideia de uma grade computacional trata de usuários solicitarem recursos computacionais sobre ela, através de um sistema de roteamento de serviços que permite a requisição remota desses recursos [Goes et al. 2005]. O provisionamento de recursos só é possível por meio de definições de políticas de uso e prioridades, modeladas por algoritmos de escalonamento, cada qual com suas características e métricas. Um algoritmo de escalonamento pode ser heurístico estático, heurístico dinâmico ou metaheurístico [Dubey et al. 2018].

O tempo para tomada de decisão é um fator crítico, o que favorece a escolha de algoritmos simples, *i.e.*, *First Come First Serve* (FCFS), *Extensible Argonne Scheduling System* (EASY), Max-Min e Min-Min. O FCFS apoia-se no princípio do quanto antes submetido, antes será atendido [Singh et al. 2017]. O EASY *Backfilling* [Mu'alem and Feitelson 2001] estende a regra do FCFS, porém, antecipa o atendimento em caso de existência de lacunas na sequência de execuções. O Max-Min prioriza as tarefas maiores em detrimento das menores, enquanto o Min-Min, simplesmente, inverte a priorização de atendimento de tarefas do Max-Min [Sharma and Atri 2017].

O algoritmo mais adequado depende das políticas estabelecidas pelos administradores da grade e pelo contexto em que essas tarefas estão inseridas. Todavia, a aplicação do algoritmo se baseia na disponibilidade de recursos e nas informações fornecidas pelos usuários, *i.e.*, tempo de uso e quantidade de recursos. O monitoramento de recursos de grades é automatizado e preciso. Em contrapartida, as previsões de uso informadas pelos usuários geralmente não apresentam a mesma precisão, por conta disso o estabelecimento de políticas que busquem minimizar tais problemas.

3. Algoritmo Beneficiador de Usuários Conscientes (BUC)

Em um ambiente composto de usuários despreocupados com a precisão de sua estimativa de uso dos recursos, algoritmos de escalonamento têm dificuldades de apoiarem-se nos valores de *Walltime* informados para a sua tomada de decisão. Certos algoritmos chegam até a contar com este erro, como é o caso dos baseados em *Backfilling*, porém falta tratamento dessas situações de forma mais concreta. O princípio deste trabalho é incentivar os usuários a aumentar o grau de precisão de suas previsões de uso dos recursos da grade. A técnica de bonificação é usada, favorecendo os usuários que apresentam maior grau de

precisão, em caso de fila de espera por recursos. Por outro lado, usuários que têm previsões imprecisas, tendem a perder posições na fila. Partindo dessa ideia base, podem-se obter previsões mais precisas, podendo-se, assim, propor algoritmos de escalonamento com decisões orientadas pelas previsões de uso de recursos informadas pelos usuários.

A proposta do algoritmo de reposicionamento da fila de espera por recursos, nomeado de **Beneficiador do Usuário Consciente** (BUC), tem a aplicação da técnica de bonificação e punição a partir de um *score* do usuário (S_u), que representa o grau de confiança na estimativa de tempo de uso de recursos. Inicialmente, todos os usuários têm seus $S_u = 60$, de uma escala de 0 a 100. Ao término da execução de cada tarefa, este valor é reavaliado, sendo incrementado, se o usuário foi preciso na sua estimativa de *Walltime* da tarefa, ou decrementado, caso contrário. A variação ocorre a partir da precisão P obtida, representada na Equação 1, em que W_t representa o *Walltime* da tarefa e R_t o tempo de execução real da tarefa. O valor varia entre 0 e 1, sendo que, quanto mais próximo de 1, mais preciso. Requisições que tenham seu *walltime* extrapolado e, consequentemente, são interrompidas pelo RMS, não implicam em reavaliação do S_u. Tal escolha se deve pois muitas das tarefas submetidas não possuírem finalidade de serem completadas, mas, sim, de apenas executarem seu procedimento enquanto seu *walltime* não estourar. Por conta dessa imprevisibilidade da motivação da requisição, tais tarefas foram optadas por serem descartadas da reavaliação do *score*.

$$P = R_t / W_t \tag{1}$$

O *score* do usuário (S_u) é apenas um dos componentes do S_f, valor absoluto usado para o posicionamento das tarefas na fila (Equação 2). A Equação também pondera métricas vinculadas a própria tarefa, *Walltime* (W_t) e o tempo de espera na fila (T_e). Desta forma, duas ou mais tarefas de um mesmo usuário, que competem pelos mesmos recursos e são beneficiados ou punidos pelo S_u, terão posicionamentos distintos, visto que tem valores próprios W_t e T_e. Finalmente, o último componente da Equação 2 é a constante (K_p), configurada pelo administrador da grade, com o intuito de controlar o impacto do S_u no cálculo do S_f. Portanto, quanto maior o S_f, melhor a posição da tarefa na fila. O índice de posicionamento de uma tarefa na fila (S_f) é reavaliado a cada novo evento. Um evento pode ser a remoção ou a inclusão de uma tarefa na fila.

$$S_f = (S_u * 0,8 + (1/(W_t/100000)) * 0,2) * K_p + T_e \tag{2}$$

4. Experimentação e Análise dos Resultados

O ambiente de simulação escolhido foi o Batsim[1], simulador de grades computacionais que permite descrever infraestruturas físicas de agregados heterogêneos e estender seus componentes, *i.e.*, algoritmos de escalonamento [Dutot et al. 2015, Poquet 2017]. A versão utilizada nos experimentos foi a versão estendida disponível em `https://www.pypi.org/project/batsim-py/` [Casagrande et al. 2020]. A grade computacional escolhida foi a GRID'5000 [Bolze et al. 2006], com dados reais do sítio de Lille anonimizados. Lille conta com 39 servidores distribuídos em 4 clusters heterogêneos.

Os dados trabalhados referem-se ao último ano completo de requisições, 2019, que totalizaram, no período, 2785 tarefas. A análise preliminar do comportamento dos usuários reais, em função de S_u, identificou a ausência de representantes em algumas

[1]Disponível em: `https://github.com/oar-team/batsim`

categorias. Para sanar este problema, injetou-se mais 300 tarefas a base, garantindo a representação em todas as categorias. O objetivo do experimento é avaliar o impacto no posicionamento da fila em função da consciência do usuário. O algoritmo que serve de base de comparação é o FCFS, cujo valores obtidos, por usuário, serviram para normalizar o eixo y da Figura 1, o que permite a análise comparativa entre 3 usuários reais e 3 injetados. Os valores de K_p aplicados foram 1440 (azul claro) e 10800 (azul escuro).

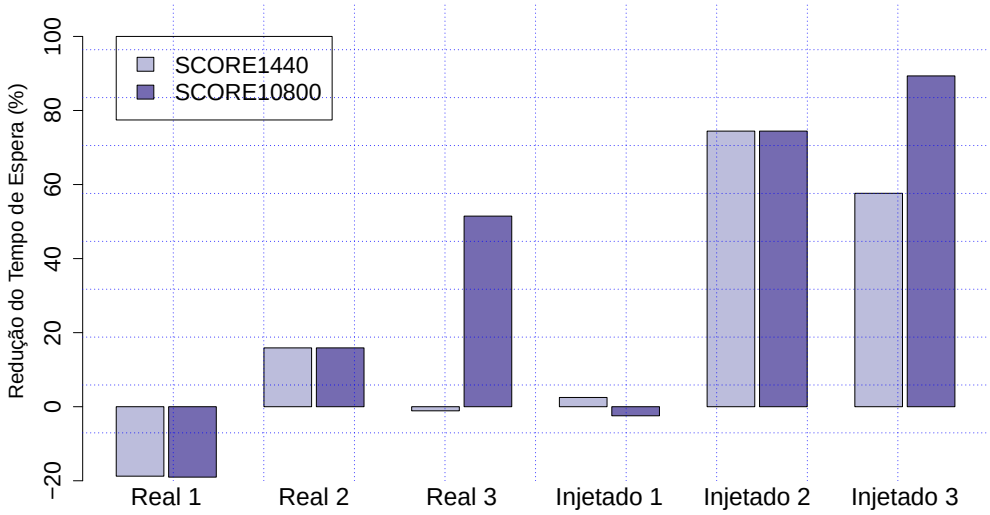

Figura 1. Redução percentual do tempo de espera de seis usuários.
Fonte: Produção dos próprios autores.

Os usuários Real 1, 2 e 3 foram escolhidos em função de dois fatores, maior número de tarefas submetidas no período analisado e a categoria que eles representam, um usuário pouco, um medianamente e um muito preciso, respectivamente, com o fim de representarem cada um dos perfis comuns e que representam boas disparidades de comportamento para avaliação da carga de entrada utilizada. Partindo dos usuários escolhidos para representar cada categorias, temos que o usuário Real 1 tem S_u médio $(2,7)$, o Real 2, $(45,93)$, e o Real 3, $(68,22)$. Os três usuários Injetados representam as categorias 0%, 50% e 100% de precisão (P) no *Walltime* (W_t) – seguindo a mesma lógica de prereresentatividade dos usuários Real 1, 2 e 3 –, sendo estes dispostos em Injetado 1, 2 e 3, respectivamente. Cada usuário Injetado submete 100 requisições com *Walltime* de 3600 segundos, o que é, de forma geral, um *walltime* considerado baixo para a média da carga de entrada, o que pode implicar em uma certa bonificação de pontuação pelo *walltime* da tarefa, sendo representado por certos ganhos de posições dependendo da sua concorrência. Os S_u resultantes dos usuários Injetados 1, 2 e 3 foram, respectivamente, $(4,8)$, $(96,14)$ e (99).

Os resultados são promissores, os benefícios para os usuários conscientes são visíveis, os usuários Real 2 e 3 e os Injetados 2 e 3 tiveram ganhos entre 20% e 90% em seus posicionamentos na fila. Os usuários com menor grau de precisão (P) chegaram ao percentual de 20% de perda de posição. Observou-se também que a escolha correta do K_p é essencial, pois o comportamento com $K_p = 10800$ apresentou resultados mais coerentes com o objetivo de justiça traçado pelo algoritmo BUC.

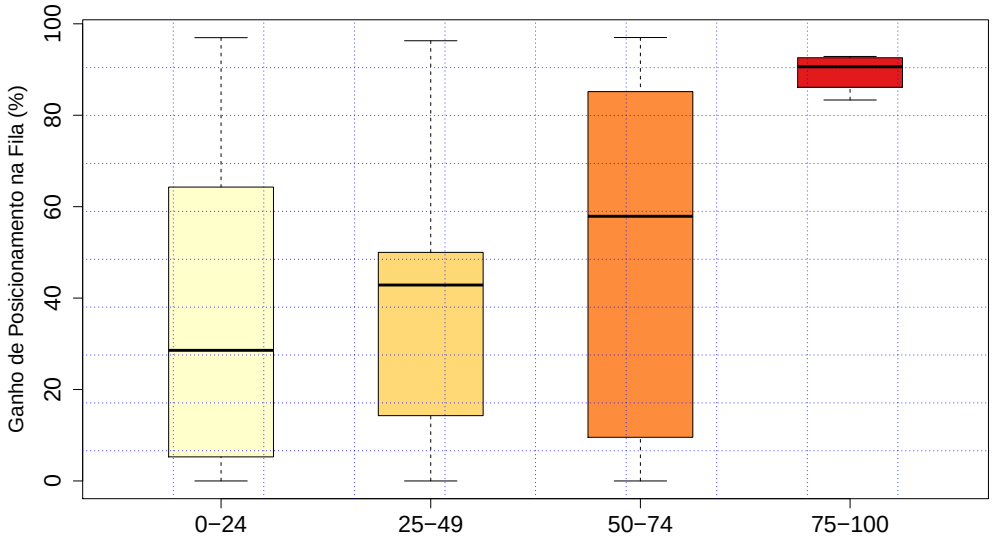

Figura 2. Distribuição de ganhos de posicionamento na fila de requisições agrupados por intervalo S_u e $K_p = 10800$.

Fonte: Produção dos próprios autores.

A análise individualizada por usuário, aplicado aos 6 usuários escolhidos, mostra a eficiência do algoritmo BUC. Este comportamento é exclusivo destes usuários ou pode ser generalizado por categoria? Na Figura 2, é possível observar o comportamento geral do impacto na fila com todos os usuários. Os usuários foram agrupados de acordo com seus S_u, considerando-se o posicionamento para filas maiores que 5 tarefas e com $K_p = 10800$. O eixo x representa o percentual de ganho de posição na fila de atendimento, isso, no momento que a tarefa é incluída na fila. Observe que o máximo em todas as categorias é próximo a 100%, o que indica que, pelo menos uma tarefa por categoria, foi posicionada nas primeiras posições da fila. Pode-se afirmar ainda que a variação no ganho de posicionamento das categorias $0-24\%$, $25-49\%$ e $50-74\%$ é grande. Porém, com a observação da mediana, é possível identificar uma tendência de maior ganho de posicionamento, conforme o índice de confiança do usuário sobe. O primeiro intervalo de S_u $(0-24\%)$ tem um ganho mediano de 30%, o segundo, valores superiores a 40%, o terceiro, aproxima-se de 60%, chegando, no último intervalo $(75-100\%)$, a mais de 90%.

Ainda analisando os valores obtidos, é possível ressaltar que o ganho de posições de uma tarefa depende das características de suas concorrentes. Portanto, uma determinada tarefa pode ser influenciada positivamente mesmo com S_u baixo, e vice-versa. Inclusive, em caso de um contexto de concorrência onde todos os usuários possuem o mesmo comportamento – sejam todos precisos ou todos imprecisos, por exemplo – constitui-se o BUC com influência nula ou próxima disso, ou seja, a existência do score não irá repercutir em alguma influência no ordenamento da fila, tendo, assim, o comportamento de ordenação das tarefas conforme o algoritmo de base utilizado, neste caso o FCFS. Além disso, na implementação realizada, o sistema é passível da malícia de determinados usuários em realizarem requisições leves e de *walltime* baixo, a fim de conseguir inflar o *score* do usuário rapidamente, dado que o fato do *walltime* da tarefa ser muito baixo pos-

sui razoável interferência no algoritmo, conseguindo, assim, ganhar vantagem em uma requisição mais demorada, dado estar, então, com o *score* maior.

Os resultados obtidos demonstram a eficiência do algoritmo BUC, apresentado bons ganhos percentuais no posicionamento, mesmo em intervalos com S_u distantes do ótimo ($P = 1$). A conscientização dos usuários na determinação do *Walltime* permite a especulação de outros algoritmos de escalonamento baseados na confiança do usuário a partir da métrica em questão, o que, consequentemente, permite aplicações que otimizem melhor a gerência dos recursos e a econômica dos mesmos.

5. Conclusão

A maximização do uso de recursos sem detrimento na qualidade de serviço prestado é a meta de qualquer administrador de grade computacional. Porém, na maioria das vezes, estes objetivos são conflitantes. Na prática, administradores de grades definem políticas que amenizam este conflito, normalmente, apoiam-se no histórico de utilização dos recursos e nas previsões de uso informado pelos usuários. Usuários solicitam recursos sem serem penalizados por suas imprecisões de demanda e, na maioria das vezes, nem estão conscientes que estas imprecisões impactam no provisionamento eficiente dos recursos.

A proposta BUC apoia-se na bonificação/punição de usuários, de acordo com o grau de precisão de suas previsões. Os resultados preliminares demonstram que o algoritmo é eficiente, favorecendo usuários preocupados com a exatidão de suas previsões em até 90% em seu posicionamento inicial na fila de recursos. Usuários despreocupados com suas previsões de demanda tiveram perdas de posicionamento de até 20%. A concepção desse algoritmo permite destacar usuários com previsões mais precisas na demanda de recursos. Com informações mais confiáveis da demanda de recursos é possível conceber novos algoritmos de escalonamento que considerem estas informações.

A curto prazo, como continuação deste estudo, pretende-se aplicar os conceitos de usuários conscientes a algoritmos clássicos de escalonamento de tarefas em grades computacionais, assim como a profundidade das estratégias da abordagem, a fim de melhorar seus resultados e possuir conclusões vantajosas e desvantajosas de forma mais concreta. A médio prazo, tem-se como objetivo a otimização do provisionamento de tarefas em grades computacionais, associando-se a métricas de consumo de energia.

Agradecimentos

Os autores agradecem a Fundação de Amparo a Pesquisa de Santa Catarina (FAPESC), a GRID'5000 e o Laboratório de Processamento Paralelo Distribuído (LabP2D), do CCT/UDESC.

Referências

Alves, D. M., Costa, M., Furtado, M. R. S., and Moravia, R. V. (2013). Computação em nuvem: Um estudo sobre seus conceitos, tecnologia e aplicação.

Bolze, R., Capello, F., Caron, E., Daydé, M., Desprez, F., Jeannot, E., Jégou, Y., Lanteri, S., and Leduc, J. (2006). GRID'5000: A large scale and highly reconfigurable experimental grid testbed. *The International Journal of High Performance Computing Applications*, 20(3):481–494.

Casagrande, L., Koslovski, G., Miers, C. C., and Pillon, M. (2020). DeepScheduling: grid computing job scheduler based on deep reinforcement learning. In *The 34-th International Conference on Advanced Information Networking and Applications (AINA-2020)*.

Dubey, K., Kumar, M., and Sharma, S. C. (2018). Modified HEFT algorithm for task scheduling in cloud enviroment. *6th International Conference on Smart Computing and Communications*, 125:725–732.

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.

Eng, K., Muhammed, A., Mohamed, M. A., and Hasan, S. (2020). A hybrid heuristic of variable neighbourhood descent and great deluge algorithm for efficient task scheduling in grid computing. *European Journal of Operational Research*, 284(1):75 – 86.

Goes, L. F. W., Neto, D. O. G., Ferreira, R., and Cirne, W. (2005). Computação em grade: Conceitos, tecnologias, aplicações e tendências. *Escola Regional de Informática de Minas Gerais*.

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.

Pillon, M., Amarante, T. d., Koslovski, G., and Miers, C. (2020). Escalonamento de tarefas em grades computacionais gruiado pelo consumo de energia com os algoritmos easy backfilling e heft. *REABTIC*.

Poquet, M. (2017). *Approche par la simulation pour la gestion de ressources*. PhD thesis, Université Grenoble Alpes.

Sharma, N. and Atri, S. T. S. (2017). A comparative analysis of min-min and max-min algorithms based on the makespan parameter. *International Journal of Advanced Research in Computer Science*, 8(3).

Singh, A. B., Bhat, S., Raju, R., and D'Souza, R. (2017). A comparative study of various scheduling algorithms in cloud computing. *American Journal of Intelligent Systems*, 7(3):68–72.