Optimizando Escalonamentos de Alinhamentos Biológicos sob Restrições de Memória

Gabriel C. Mills[1], **Cristina Boeres**[1], **Vinod E.F. Rebello**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brazil

gmills@id.uff.br, {boeres,vinod}@ic.uff.br

Abstract. *In related work, pipeline scheduling has been shown to be asymptotically optimal in terms of the makespan, while consuming the least amount of memory for a given degree of parallelism, when executing biological alignment experiments with variants of similar species. This paper aims to determine if makespans shorter than those of pipeline schedules exist for experiments of limited size and, if so, how much improvement is possible and at what cost.*

Resumo. *Em um trabalho anterior, escalonamentos em pipeline foram demonstrados ser assintoticamente ótimo em termos de makespan enquanto requer o menor consumo de memória para executar um conjunto de alinhamentos biológicos. Em particular, este artigo foca em determinar se makespans menores que aqueles de escalonamentos em pipeline existem para experimentos de tamanho limitado e, se existir, quanto é possível melhorar e a qual custo.*

1. Introdução

Muitas aplicações de HPC e de *big* data têm demandas de memória que variam durante suas execuções, mas muitos sistemas de gerenciamento de recursos continuam reservando o consumo máximo delas ao escalonar tais *jobs*. Um exemplo que ganhou muita publicidade é o alinhamento de sequências genéticas, uma das áreas chave da Bioinformática responsável pela análise de sequências genéticas para estudar organismos e a similaridade de suas características. A recente pandemia mostrou a importância desse tipo de estudo no entendimento de infecções e no desenvolvimento de formas de prevenção e combate, exemplificada pelas análises por cientistas ao redor do mundo das 18 milhões de sequências de DNA das variantes de SARS-CoV-2 disponíveis em bases de dados públicos. Neste contexto, o escalonamento *pipeline* de *jobs* de alinhamento de sequências genéticas se mostrou extremamente eficiente em relação execuções tradicionais, conseguindo aproveitar melhor a memória disponível e baratear o custo financeiro da execução [Sodré et al. 2022]. Foi comprovado que estes escalonamentos consomem a menor quantidade de memória para um dado grau de paralelismo e estão assintoticamente ótimos em termos de *makespan*.

Este trabalho visa investigar este problema de escalonamento do ponto da vista da aquisição de recursos de nuvem. Na maioria dos casos, os provedores de nuvem oferecem recursos de computação IaaS sob um esquema de pagamento conforme o uso, com preços por hora definidos para instâncias configuradas com capacidades fixas e predeterminadas. Normalmente, os preços aumentam linearmente com o número de CPUs e a

quantidade de memória para uma dada classe de instância. Por outro lado, o desempenho de muitas aplicações paralelas ou de *big data* é frequentemente limitado pelo número de processadores ou pela quantidade de memória disponível. O problema de encontrar a instância mais adequada para alcançar um desempenho aceitável dentro de um determinado orçamento tornou-se um foco de grande interesse [Soares 2023]. Este trabalho investiga uma abordagem para melhorar a utilização de uma escolhida instância e reduzir os custos de um experimento sem sacrificar o desempenho através de um escalonamento inteligente de seus *jobs*, sem a necessidade de alterar a configuração da aplicação. A premissa central deste trabalho continua sendo que atrasar os *jobs* melhora o desempenho! A pergunta quer queremos responder é, dado uma quantidade de memória disponível, se existe um escalonamento mais rápido do que o do escalonamento pipeline.

2. Metodologia

Para este trabalho foi desenvolvido um algoritmo genético para gerar ordens diferentes dos alinhamentos para ser avaliados por uma heurística gulosa de escalonamento baseado em *List Scheduling* [Hwang et al. 1989]. O objetivo é achar um escalonamento com o menor tempo para executar W alinhamentos de um experimento, supondo que só há uma quantidade de memória M disponível, cada alinhamento i consome $k_i < M$ de memória e leva t_i de tempo de execução. A estrategia aproveita um modelo, elaborado num trabalho anterior [Mills et al. 2023], para estimar os tempos e consumos de memória de alinhamentos de sequências genéticas de tamanhos arbitrários quando a ferramenta de bioinformática MASA [De O. Sandes et al. 2016] é usada para realizar os alinhamentos.

O que torna este problema de escalonamento diferente é o fato que esta aplicação, como outras da área de *big data*, tem um consumo crescente de memória durante sua execução. Como a maioria de algoritmos de escalonamento não considera o consumo dinâmica de memória, não alcançam resultados satisfatórios. Enquanto assintoticamente ótimo em tempo, vale a pena buscar escalonamentos melhores do que o de pipeline?

3. Avaliação Experimental

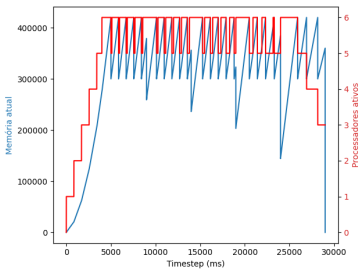
Apresentamos o seguinte exemplo que compara um escalonamento em lista com o algoritmo em pipeline definido em [Sodré et al. 2022]. Nesse experimento utilizou-se um conjunto de 30 alinhamentos similares, cada um com consumo máximo da memória de 120000 *bytes* e um tempo de execução de 5000 ms. Com uma quantidade de memória disponível de 420000 *bytes*, o escalonamento pipeline consegue executar até 6 alinhamentos em paralelo. Pelos tempos de inicio dos alinhamentos nos dois escalonamentos apresentados na tabela de Figura 1(a), percebe que os dois são muito parecidos, o de pipeline sendo 182,6 segundos mais lento do que o tempo do *List Scheduling* de 28,984 segundos. Na figura 1(b), observa-se o consumo de memória não ultrapassa o limite porque um *job* só seria iniciado prevendo memória suficiente para sua execução. Há somente 27 picos porque três *jobs* começam e terminam junto com outros (veja a próxima figura).

Também percebe-se que as distâncias entre os picos não são iguais entre si, o que pode ser justificado pela figura 1(c), que apresenta o intervalo entre o início de *jobs* sucessivos no escalonamento. Nessa figura, observa-se que os atrasos oscilam especialmente no final do escalonamento, mas aparece mais regular no começo (por volta de 848ms) como no caso do pipeline. Tem três *jobs* que foram iniciados sem atrasos.

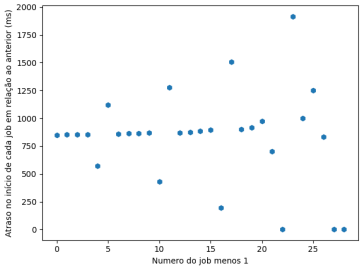
Considerando diferentes quantidades W de *jobs*, foi observado que, mantendo o limite de processadores $P \leq W$ e memória $M = (P + 1) \times (k/2)$, o escalonamento do algoritmo em lista consegue alcançar *makespans* (pouco) menores ou iguais ao do *pipeline*, dependendo do valor de W. A maior diferença entre *makespans* dos dois algoritmos é limitado na ordem de t/P.

List Scheduling		Makespan = 28984			
0	850	1702	2556	3411	3984
5102	5962	6824	7688	8555	8984
10261	11133	12009	12893	13791	13984
15490	16392	17310	18283	18984	18984
20900	21900	23150	23984	23984	23984
Pipeline		Makespan = 29166.6			
0	833.3	1666.6	2500	3333.3	4166.6
5000	5833.3	6666.6	7499.9	8333.3	9166.6
10000	10833.3	11666.6	12500	13333.3	14166.6
15000	15833.3	16666.6	17500	18333.3	19166.6
20000	20833.3	21666.6	22499.9	23333.3	24166.6

(a) Os tempos de inicio, em milissegundos, da mesma sequência de 30 jobs de alinhamento nos seus respectivos escalonamentos.



(b) Representação gráfica do *List Scheduling* da figura 1(a), com makespan de 28984ms, e grau de paralelismo ao longo da execução.



(c) Os tempos de espera para inicio de cada job, em relação ao anterior, no escalonamento da figura 1(b) em milissegundos.

Figura 1. Comparação dos escalonamentos do algoritmo de lista e de pipeline

4. Conclusão e análises futuras

O método em *List Scheduling* apontado neste artigo se mostra promissor, com seus escalonamentos podendo ser melhores do que os em pipeline em situações de restrição de memória e processadores. Como o algoritmo em pipeline é destinado somente a escalonamentos de alinhamentos similares, o algoritmo em *List Scheduling* em conjunto com algoritmo genético se torna a opção mais flexível para experimentos biológicos de alinhamento de sequências variadas. Trabalhos futuros pretendem melhorar a eficiência do algoritmo genético proposto para esse tipo de experimento e avaliar seus escalonamentos em ambientes de nuvem.

Referências

De O. Sandes, E. F., Miranda, G., Martorell, X., Ayguade, E., Teodoro, G., and De Melo, A. C. M. A. (2016). MASA: A multiplatform architecture for sequence aligners with block pruning. *ACM Transactions on Parallel Computing*, 2(4).

Hwang, J.-J., Chow, Y.-C., Anger, F. D., and Lee, C.-Y. (1989). Scheduling precedence graphs in systems with interprocessor communication times. *SIAM Journal on Computing*, 18(2):244–257.

Mills, G., Cavalcante, S., Boeres, C., and Rebello, V. (2023). Estimativa de tempo de alinhamentos biológicos na nuvem. In *Anais da VIII Escola Regional de Alto Desempenho do Rio de Janeiro*, pages 14–16, Porto Alegre, RS, Brasil. SBC.

Soares, D. M. d. C. (2023). Choosing the right cloud configuration for you and your workload. Master's thesis, Universidade Federal Fluminense.

Sodré, D., Boeres, C., and Rebello, V. (2022). Making the most of what you pay for by delaying tasks to improve overall cloud instance performance. In *Anais Estendidos do XXIII Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 9–16. SBC.