Comparação de rastros para análise de desempenho de aplicações paralelas

Alef Farah, Lucas Mello Schnorr

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

afarah@inf.ufrgs.br, scnhorr@inf.ufrgs.br

Resumo. *Existem atualmente diversas ferramentas para a visualização de rastros de aplicações paralelas, porém poucas oferecem uma comparação automática e escalável de dois arquivos de rastro. Neste artigo apresentamos uma estratégia para realizar a comparação de rastros através de um algoritmo de alinhamento de sequências, e alguns resultados preliminares de nosso trabalho que indicam a possibilidade de desenvolver uma ferramenta escalável.*

1. Introdução

A análise de desempenho de aplicações geralmente passa por uma fase de comparação entre diferentes versões ou execuções de uma mesma aplicação, o que torna desejável também a comparação de seus rastros de execução. Tal análise, quando manual, torna-se complicada à medida em que o número de eventos registrados aumenta. Portanto, é desejável uma ferramenta que faça a comparação automática entre os rastros, apresentando as diferenças através de técnicas de visualização.

A caracterização automática de eventos comparáveis entre dois arquivos de rastro e a reconstrução cronologicamente fiel às execuções originais é não trivial. A escalabilidade é outro problema na comparação, dado a complexidade de identificar regiões semelhantes automaticamente. Existem ferramentas que realizam tal comparação, mas com automação insuficiente [Weber et al. 2013]. Tendo como objetivo superar tais dificuldades, estudamos soluções propostas e apresentamos resultados preliminares a seguir.

1.1. Trabalhos relacionados

[Velho et al. 2013] utilizam marcadores especiais para identificar regiões comparáveis dos arquivos de rastro. Tal processo, apesar de funcional, apresenta um caráter manual indesejável, possuindo baixa escalabilidade.

[Weber et al. 2012] alinham as sequências de eventos presentes nos arquivos de rastro de acordo com sua similaridade utilizando algoritmos clássicos usados em bioinformática. Acomplam a isso métricas de similaridade e de diferenças temporais [Weber et al. 2013] para fornecer uma análise automática ao usuário, através de uma ferramenta proprietária

1.2. Objetivos

Nosso objetivo é desenvolver uma ferramenta de software livre capaz de comparar arquivos de rastro de maneira automática e escalável, fornecendo ao usuário uma representação gráfica das diferenças. Também pretendemos validar nosso trabalho utilizando rastros de aplicações paralelas reais de larga escala.

Para tal, começamos estudando as estratégias utilizadas até então (vide seção 1.1). A estratégia de marcadores definidos pelo usuário [Velho et al. 2013] foi descartada por seu caráter manual. Em seu estágio inicial, nosso projeto atualmente realiza um estudo crítico da estratégia de alinhamento de sequências aliado a métricas de similaridade [Weber et al. 2012], procurando identificar possíveis problemas e sugerindo alternativas. Partimos de tais estudos para buscar novas soluções, e apresentamos alguns resultados preliminares à seguir. Desenvolvemos nossa ferramenta a partir da biblioteca PajeNG [Schnorr and Farah 2015].

2. Implementação

Como no trabalho em estudo, nossa implementação compara os eventos registrados nos arquivos de rastro pelo nome, atribuindo uma pontuação de acordo com o caso:

α, caso os eventos sejam iguais
β, caso os eventos difiram
γ, caso haja uma lacuna em uma das sequências de eventos

A pontuação pode ser definida pelo usuário, sendo a padrão $\alpha = 2, \beta = -1, \gamma = -1$, penalizando diferenças e recompensando semelhanças. Registramos as pontuações recursivamente em uma matriz D. Sendo $\theta(i, j)$ a função que retorna α ou β comparando dois elementos:

$$D(i,j) = \begin{cases} j * \gamma & i = 0 \\ i * \gamma & j = 0 \\ max \begin{cases} D(i-1, j-1) + \theta(i,j) \\ D(i, j-1) + \gamma \\ D(i-1, j) + \gamma \end{cases} & i \neq 0, j \neq 0 \end{cases}$$

Percorremos então a matriz, a partir do fim, indo para cima, para a esquerda, ou ambos (diagonal), selecionando sempre o maior valor até a primeira linha ou coluna, afim de maximizar a soma das pontuações percorridas. Passamos assim uma vez por cada elemento selecionando sempre a comparação que levou à maior pontuação global, dessa forma alinhando regiões semelhantes das sequências. Esse é o algoritmo de Needleman-Wunsch, na prática utilizamos uma versão optimizada com espaço linear e tempo quadrático [Hirschberg 1975].

Dividir o arquivo de rastro de acordo com seu grafo de chamadas e seus ranks permite que executemos o algoritmo em paralelo para múltiplos fragmentos do arquivo, que são posteriormente reunidos, possibilitando um tempo melhor, dependendo da subdivisão do arquivo.

3. Resultados preliminares

Definimos "número de subníveis" de um arquivo de rastro como o número de ranks (para uma aplicação utilizando Message Passing Interface) mais o nível de chamada máximo do grafo de chamadas do rastro, menos dois, pelo fato de o rank0 e a função main ou equivalente estarem sempre presentes.

Os diagramas de caixa à seguir apresentam os resultados para testes de tempo total de execução. Todos testes consistem de trinta execuções para cada um dos dez arquivos de rastro testados, cujo tamanho da sequência de eventos está distribuído no eixo horizontal. Os arquivos foram gerados por um script desenvolvido pelos autores, que gera arquivos com tamanho de sequência e número de subníveis pré-determinado. Os testes foram realizados em um computador pessoal moderno com quatro processadores.

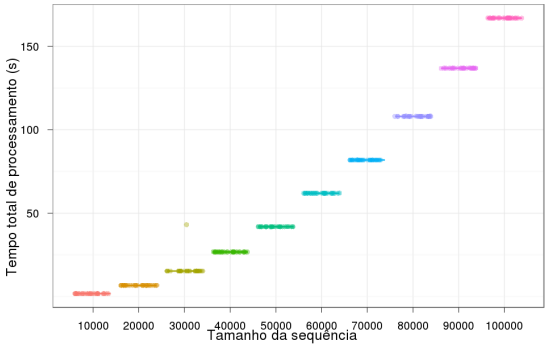

Figura 1. Zero subníveis

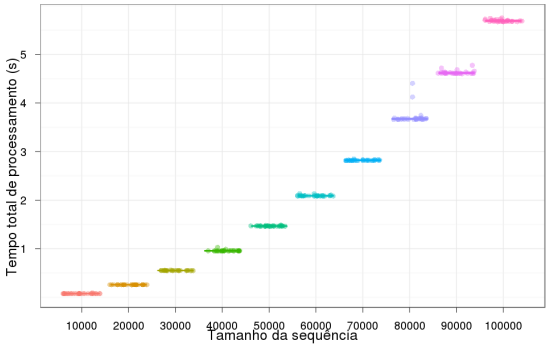

Figura 2. Dez subníveis

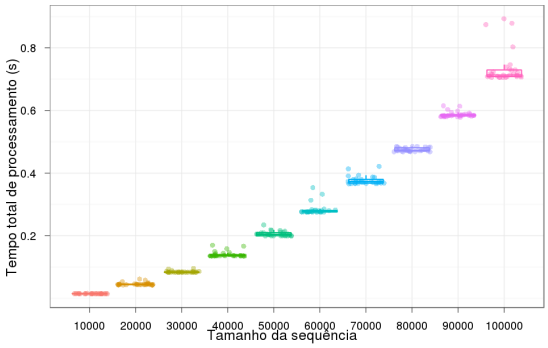

Figura 3. Cem subníveis

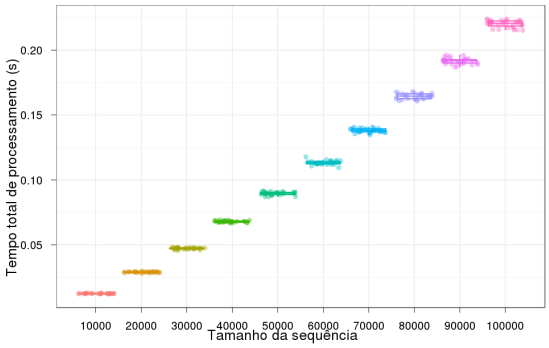

Figura 4. Mil subníveis

O diagrama do primeiro teste (vide Figura 1) mostra tempos extremamente altos, se comparados com os demais testes. O motivo disso é que nossa aplicação trabalha de maneira sequencial em tal cenário - não havendo subníveis nos quais dividir o arquivo, não há a possibilidade de paralelização. O desvio maior nas figuras 3 e 4 deve-se ao fato de que quanto menor o tempo de execução, maior a significância da interferência de tarefas do sistema operacional.

Para melhor caracterizar a tendência de melhora no tempo previamente mencionada, foi realizada uma regressão polinomial de segundo grau sobre os dados apresentados acima, via quadrados mínimos. A tabela à seguir apresenta o coeficiente de x^2 para cada teste, bem como o resíduo da regressão. O coeficiente diminui à medida em que a subdivisão aumenta, sugerindo um caráter cada vez mais longe do quadrático. O resíduo alto no primeiro teste deve-se ao único valor atípico presente em tal teste, bem fora do esperado.

Subníveis	x^2	R^2
0	$1.64373308e-08$	880.30314192
10	$5.51077027e-10$	0.81290156
100	$6.42698034e-11$	0.11137104
1000	$7.42562414e-12$	0.0007698

Tabela 1. Valores da regressão sobre os dados de tempo de execução

Uma possibilidade atualmente sendo explorada é a realização de um alinhamento semi-global [Eddy 2008] das sequências, ao contrário do alinhamento global realizado no trabalho em estudo. Tal estratégia é interessante em casos onde se possuí diferentes versões do mesmo arquivo, em que uma é uma versão reduzida da outra, pois o princípio é o mesmo que no alinhamento de sequências genéticas - a significância pode ser maior utilizando-se tal alinhamento. Tal sugestão ainda está em fase de estudo, pretendemos realizar testes com aplicações reais no futuro para validá-la.

4. Conclusão

Em relação à estratégia adotada no momento, concluímos que é plausível o desenvolvimento de uma aplicação de comparação de rastros automática e escalável. Como o número de sub-níveis em aplicações reais é, em geral, maior que zero, o baixo tempo e o caráter linear do uso de memória apresentados na seção 2 demonstram ser possível pensar no desenvolvimento de tal ferramenta, apesar de mais testes serem necessários.

No futuro pretendemos revalidar tal estratégia com estudos de caso utilizando aplicações reais, com enfoque em aplicações paralelas utilizando as bibliotecas OpenMP e MPI, bem como validar as ideias que temos a acrescentar, sempre buscando novas estratégias.

Referências

Eddy, S. R. (2008). A probabilistic model of local sequence alignment that simplifies statistical significance estimation. *PLoS computational biology*, 4(5):e1000069.

Hirschberg, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM 18 (6): 341–343*.

Schnorr, L. M. and Farah, A. (2015). Pajeng (fork). https://github.com/a442/pajeng. Accessed: Jan 20 2015.

Velho, P., Schnorr, L. M., Casanova, H., and Legrand, A. (2013). On the validity of flow-level tcp network models for grid and cloud simulations. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*.

Weber, M., Brendel, R., and Brunst, H. (2012). Trace file comparison with a hierarchical sequence alignment algorithm. In *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pages 247–254.

Weber, M., Mohror, K., Schulz, M., Supinski, B., Brunst, H., and Nagel, W. (2013). Alignment-based metrics for trace comparison. In Wolf, F., Mohr, B., and Mey, D., editors, *Euro-Par 2013 Parallel Processing*, volume 8097 of *Lecture Notes in Computer Science*, pages 29–40. Springer Berlin Heidelberg.