Impacto da variabilidade de tarefas nas distribuições de aplicações baseadas em tarefas

Lucas Leandro Nesi[1]*, **Arnaud Legrand**[2], **Lucas Mello Schnorr**[1]

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

[2]Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG
F-38000, Grenoble – France

{lucas.nesi, schnorr}@inf.ufrgs.br, arnaud.legrand@imag.fr

Resumo. *Algoritmos de distribuição de dados e tarefas em vários nós computacionais podem utilizar o poder de processamento de cada nó como parâmetro. O poder de cada nó pode ser calculado utilizando uma tarefa predominante. Entretanto, estas tarefas estão sujeitas a variabilidade. Este trabalho investiga o desempenho das distribuições estáticas quando as tarefas sofrem variabilidade.*

1. Introdução

Aplicações paralelas distribuídas requerem um mapeamento de dados e processamento entre as máquinas, de forma a distribuir a carga de trabalho e minimizar o tempo total de execução. A programação de distribuições não cíclicas [Nesi et al. 2020] é mais trabalhosa em paradigmas tradicionais como MPI. Entretanto, a programação baseada em tarefas apresenta uma flexibilidade, já que pode usar um runtime para controlar a memória de uma distribuição informada. Aplicações baseadas em tarefas são organizadas em um DAG (Grafo Acíclico Dirigido). Um runtime, como StarPU [Augonnet et al. 2011], recebe a submissão de tarefas e escalona dinamicamente para os recursos. No caso do StarPU, a distribuição dos dados em diferentes nós computacionais é estática e informada pelo programador, para melhorar a escalabilidade. Exemplos de distribuição para a álgebra linear incluindo a fatoração LU, usada neste trabalho, é o algoritmo 1D-1D para recursos heterogêneos [Beaumont et al. 2001] e extensões [Nesi et al. 2020]. A distribuição 1D-1D usa o poder de processamentos das máquinas que pode ser calculada utilizando o tempo da tarefa predominante na operação. No LU é a tarefa de multiplicação de matrizes (dgemm). Entretanto, o tempo de execução das tarefas pode sofrer variabilidade.

A variabilidade das tarefas pode ter várias causas: A estocasticidade da plataforma, acessos a memória e estados diferentes de cache podem causar uma variabilidade de curto prazo, onde durante uma operação de álgebra linear, a fatoração LU por exemplo, os kernels podem sofrer variação. Entretanto, o escalonamento dinâmico intra nó do StarPU cuida deste problema. Uma outra situação é a variabilidade de longo prazo, quando os kernels apresentam variabilidade entre as operações de álgebra linear ou em dias (tempo) de execução distintos. Isto pode acontecer por aquecimento do hardware e sua má instalação [Cornebize and Legrand 2019]. Este trabalho visa investigar se uma distribuição dinâmica é necessária por causa da variabilidade de longo prazo. Ou se distribuições estáticas que são computadas com o tempo médio das tarefas são suficientes.

*Bolsista do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

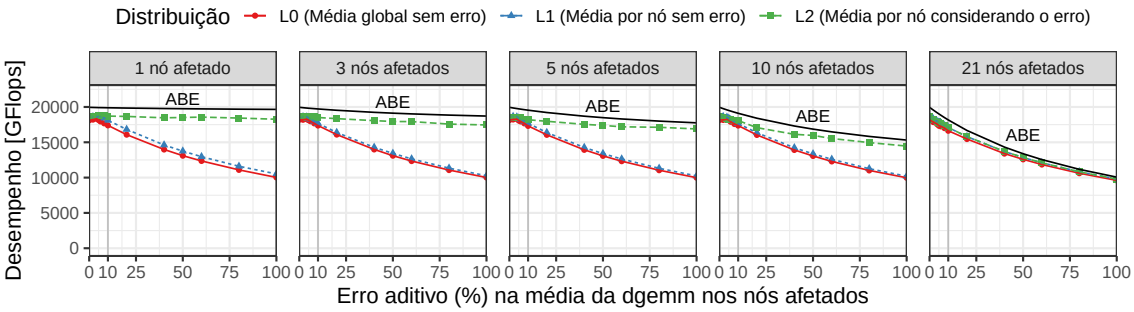

Figura 1. Desempenho das distribuições em função do erro aditivo na tarefa

Para analisar a interferência desta variabilidade foi utilizada a aplicação Chameleon [Agullo et al. 2010] e a operação LU com o StarPU-Simgrid e modelos de desempenho de 21 nós reais. Considerado cinco cenários onde uma certa quantidade de nós (1, 3, 5, 10, 21) apresenta um erro aditivo (0%-100%) na média de tempo de execução da dgemm, criando uma variabilidade sintética. Por exemplo, no cenário com um nó afetado, com um erro de 50%, se a média da execução da dgemm duraria 50ms, ela vai durar 75ms. Para cada cenário e erro aditivo, três distribuições são calculadas: L0 – onde é utilizada a média da dgemms sem erro aditivo de todas as máquinas para calcular a distribuição (representando uma calibragem antiga de apenas uma máquina para gerar a distribuição), L1 – onde é utilizada a média das dgemms sem erro aditivo por máquina para gerar a distribuição (representando uma calibragem antiga por máquina). L2 – onde é utilizada a média da dgemms com o erro aditivo (representado uma calibragem pré operação da distribuição). A Figura 1 apresenta os resultados de desempenho (GFlops) em função do erro aditivo para cada distribuição, com um *upper bound* de desempenho (ABE).

O principal aspecto dos resultados é a distância de L1 para L2. Se a perda de desempenho entre L1 e L2 for muito grande, é necessário a utilização de outras estratégias, como balanceamento dinâmico da distribuição por exemplo. Assim uma distribuição L1 seria dinamicamente balanceada para ter desempenhos semelhantes às distribuições L2. Entretanto, a diferença só acontece de maneira significativa para erros superiores a 10% e a maior variabilidade encontrada de longo prazo na literatura é inferior a 10% [Cornebize and Legrand 2019]. Desta maneira não fica evidenciado a necessidade de balanceadores dinâmicos entre os nós nesta aplicação. A utilização de calibragens antigas ou anteriores a execução da aplicação é suficiente para gerar uma distribuição estática.

Referências

Agullo, E. et al. (2010). Faster, Cheaper, Better – a Hybridization Methodology to Develop Linear Algebra Software for GPUs. In *GPU Computing Gems*. Morgan Kaufmann.

Augonnet, C. et al. (2011). StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures. *Conc. Comp.: Pract. Exp., SI:EuroPar 2009*, 23.

Beaumont, O., Legrand, A., Rastello, F., and Robert, Y. (2001). Static LU decomposition on heterogeneous platforms. *Int. Journal of High Performance Comp. Applications*.

Cornebize, T. and Legrand, A. (2019). DGEMM performance is data-dependent. Research Report RR-9310, Université Grenoble Alpes ; Inria ; CNRS.

Nesi, L. L., Schnorr, L. M., and Legrand, A. (2020). Communication-Aware Load Balancing of the LU Factorization over Heterogeneous Clusters. In *2020 IEEE 26th ICPADS*.