Viabilidade da Linguagem Python no Desenvolvimento de Aplicações de Alto Desempenho

Matheus Bernardelli de Moraes, André Leon S. Gradvohl

Faculdade de Tecnologia – Universidade Estadual de Campinas (UNICAMP)
Limeira – SP - Brasil

m121214@dac.unicamp.br, gradvohl@ft.unicamp.br

Abstract. *This paper reports partial results of a project to determine in which situations the Python language offers advantages over traditional languages in the development of High Performance Applications.*

Resumo. *Este artigo relata resultados parciais de um projeto para determinar em que situações a linguagem Python oferece vantagens em relação às linguagens tradicionais no desenvolvimento de Aplicações de Alto Desempenho.*

1. Introdução

Atualmente, devido ao aumento na demanda de aplicações de alto desempenho, motivado pelo crescente desenvolvimento e utilização de computadores cada vez mais robustos [Rhaghunathan 2006], mostram-se necessárias soluções para os problemas que surgem com esse aumento na demanda. Podemos citar, por exemplo, que a maioria das aplicações de alto desempenho são desenvolvidas utilizando-se linguagens de programação como C e Fortran. Contudo, embora eficientes, essas linguagens oferecem dificuldades na programação de sistemas de alto desempenho, principalmente por pesquisadores que não são da área de Computação. Sendo assim, são necessárias pesquisas que avaliem linguagens de programação alternativas mais flexíveis, menos complexas, mas que mantenham ou superem o desempenho dos métodos hoje utilizados [Unpingco 2009].

Uma opção que vem surgindo como uma possível escolha é a linguagem Python, pois possui sintaxe simples e fácil de utilizar, suporte a orientação a objetos e uma eficiente aritmética com arranjos multidimensionais [Sala et.al. 2008], através de bibliotecas como a *Numpy* e a *Scipy*. Além disso, ela é portável, dinamicamente tipada e permite a integração com bibliotecas de outras linguagens.

1.1. Objetivos

No projeto que este texto relata, duas frentes são abordadas: o uso de Python como uma linguagem base para o desenvolvimento de aplicações de alto desempenho; e o uso de Python como uma linguagem para o envelopamento (*wrapping*) de bibliotecas implementada nas linguagens tradicionais, especialmente as bibliotecas baseadas em POSIX threads [IEEE 1996], OpenMP [Chandra, Menon, et al. 2000] e Message Passing Interface [MPI Forum 1993].

2. Metodologia

Foram selecionados os métodos que seriam utilizados no desenvolvimento das aplicações que seriam testadas futuramente. Definimos a utilização de dois métodos para decomposição de matrizes e soluções lineares de problemas do tipo Ax=b: a Decomposição LU e o Método Cholesky. Essas rotinas foram executadas de forma serial e em paralelo em um computador de 32 Gigabytes de memória RAM com processador AMD FX, com 8 núcleos lógicos (4 físicos), 4.2 GHz, com Linux Fedora 16. Utilizou-se Python na versão 2.7.3, com o interpretador IPython 0.12.1 e o ambiente de desenvolvimento IDLE, versão 2.7.3. Para a execução paralela dos métodos escolhidos, foi definido o uso do próprio módulo paralelo do interpretador IPython, o IPython.parallel.

3. Resultados Parciais

Nesta seção apresentamos os resultados das paralelizações dos métodos Decomposição LU e Cholesky.

3.1. Decomposição LU

Inicialmente, foi desenvolvido o código serial para análise do método Decomposição LU. A implementação baseou-se no armazenamento dos tempos relativos de cada tamanho de matriz após a execução das operações necessárias. A matriz variou de 1000 em 1000 números reais.

O sistema inicia a contagem de tempo, gera uma matriz aleatória não singular, uma matriz de resultados, calcula a decomposição a partir dessas matrizes e armazena o tempo final. A Figura 1 ilustra os tempos de processamento serial e paralelo da decomposição LU.

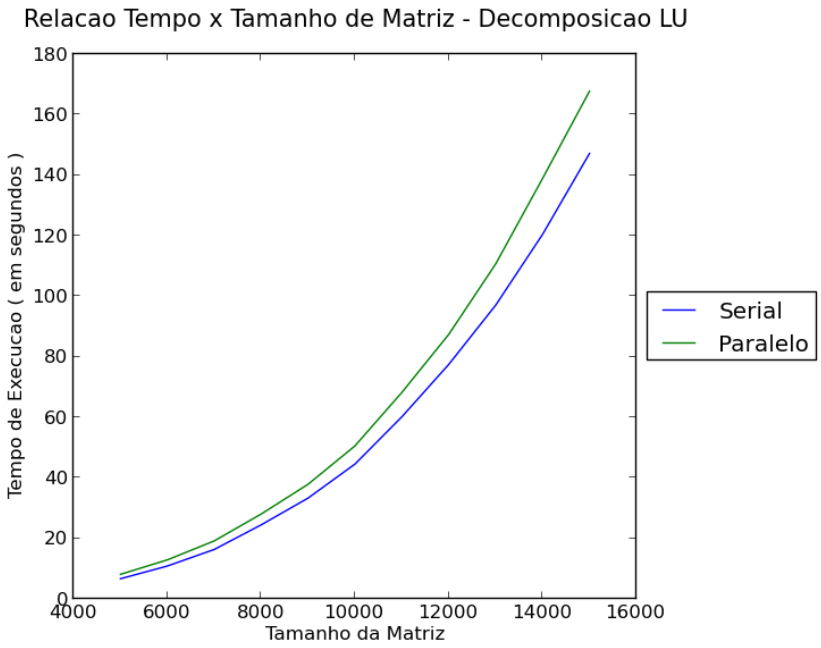

Figura 1 – Tempos de processamento da Decomposição LU serial e paralelo.

Através da análise do gráfico podemos perceber que a paralelização não surtiu efeito e que o tempo de processamento aumentou, inclusive.

3.2. Método Cholesky

No desenvolvimento do código para o método Cholesky, buscou-se a implementação de uma forma similar à decomposição LU. Novamente, a matriz variou de 1000 em 1000 números reais. A Figura 2 ilustra os tempos de processamento serial e paralelo do método Cholesky.

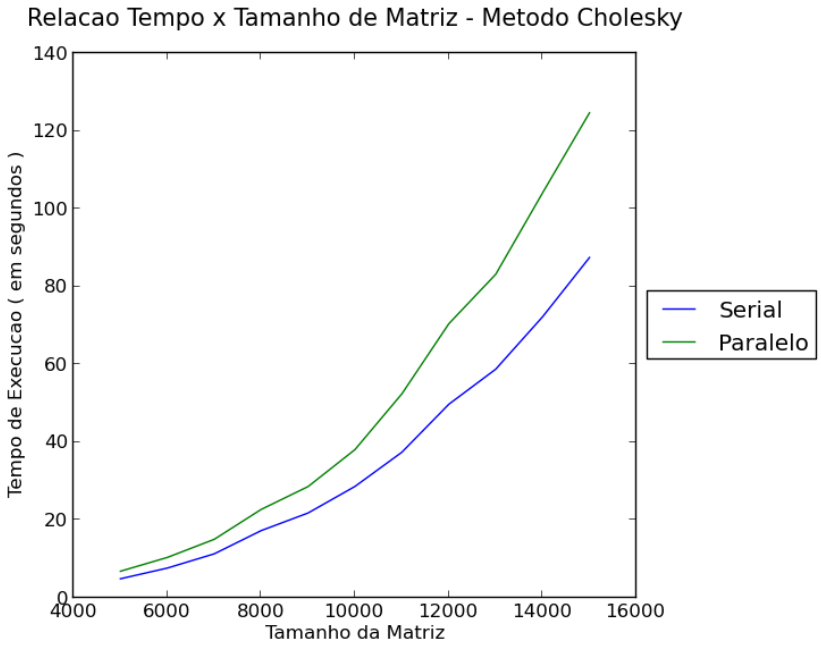

Figura 2 – Tempos de processamento do método Cholesky serial e paralelo.

Nesse caso, o sistema inicia a contagem de tempo, gera uma matriz aleatória simétrica e positiva definida, utiliza-a para o cálculo da decomposição Cholesky e finaliza a contagem de tempo. Analisando o gráfico, percebemos que, novamente, a paralelização não surtiu o efeito desejado.

4. Conclusão

Concluímos que a linguagem Python não se mostrou eficiente na utilização de suas próprias bibliotecas para paralelização. As rotinas desenvolvidas para execução de forma paralela aumentaram consideravelmente o tempo de execução. Isso ocorre por causa do tempo gasto na chamada de primitivas utilizadas na paralelização do código. Mesmo variando o tamanho dos problemas, essas instruções continuam sendo mais custosas computacionalmente do que a execução dos métodos em serial.

Como o projeto está em andamento, resta agora verificar se a linguagem é eficiente no *wrapping* de rotinas em bibliotecas tradicionais para a paralelização de sistemas que demandem um processamento alto, e assim, se tornar uma possível opção para desenvolvimento desse tipo de aplicação.

5. Referências

Raghunathan, S. Making a Supercomputer Do What You Want: High-Level Tools for Parallel Programming. Computing in Science & Engineering, 8, n. 5, Outubro 2006. 70-80.

Unpingco, J. Visual Parallel Computing Using Python-based VISION/HPC. DoD High Performance Computing Modernization Program Users Group Conference. Junho 2009. 392-394.

Sala, M.; Zurich, E.; Spotz, F. W.; Heroux, A. M. PyTrilinos: High Performance Distributed-Memory Solvers for Python. ACM Transactions on Mathematical Software, Vol. 34, No. 2, Article 7, March 2008.

Numpy. Disponível em: http://www.numpy.org/. Acesso em: 16 de janeiro de 2013.

Scipy. Disponível em: http://www.scipy.org/SciPy. Acesso em: 16 de janeiro de 2013.

Using IPython for Parallel Computing. Disponível em: http://ipython.org/ipython-doc/dev/parallel/. Acesso em: 16 de janeiro de 2013.

Overview and getting started. Disponível em: http://ipython.org/ipython-doc/dev/parallel/parallel_intro.html. Acesso em: 21 de janeiro de 2013.

Chandra, et al. Parallel Programming in OpenMP. 1a. ed. [S.l.]: Morgan Kaufmann, v. 1, 2000.

IEEE. Information Technology-Portable Operating System Interface (POSIX®)-Part 1: System Application: Program Interface (API) [C Language]. IEEE/ANSI Std.1003.1. [S.l.]. 1996.4

MPI Forum. MPI: A Message Passing Interface. Proceedings of 1993 Supercomputing Conference. Portland, Washington: [s.n.]. 1993. p. 467-474.