VIABILIDADE DA LINGUAGEM PYTHON NO DESENVOLVIMENTO DE APLICAÇÕES DE ALTO DESEMPENHO

Matheus Bernardelli de Moraes e André Leon Sampaio Gradvohl, Dr. (orientador)
matheuzmoraes@gmail.com ; gradvohl@ft.unicamp.br
Faculdade de Tecnologia - FT/UNICAMP
Agência Financiadora: SAE/UNICAMP
Palavras-chave: Linguagem de Programação Python, Computação de Alto Desempenho.

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 crescente mercado. 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.

Este trabalho utilizou métodos matemáticos para verificar a viabilidade da Linguagem Python no desenvolvimento de aplicações de alto desempenho.

Metodologia

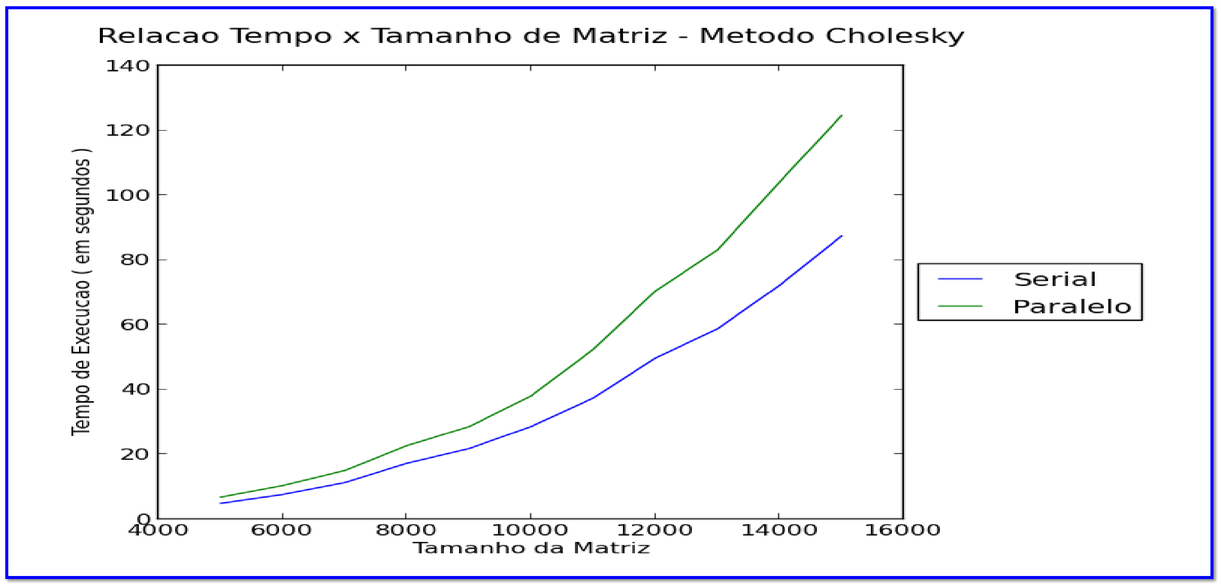
Durante o desenvolvimento deste projeto definiu-se que os métodos matemáticos utilizados para testar a linguagem seriam duas técnicas para decomposição de matrizes e soluções lineares de problemas do tipo Ax = B: a Decomposição LU e o Método Cholesky, ambas implementados nas bibliotecas Numpy e Scipy.

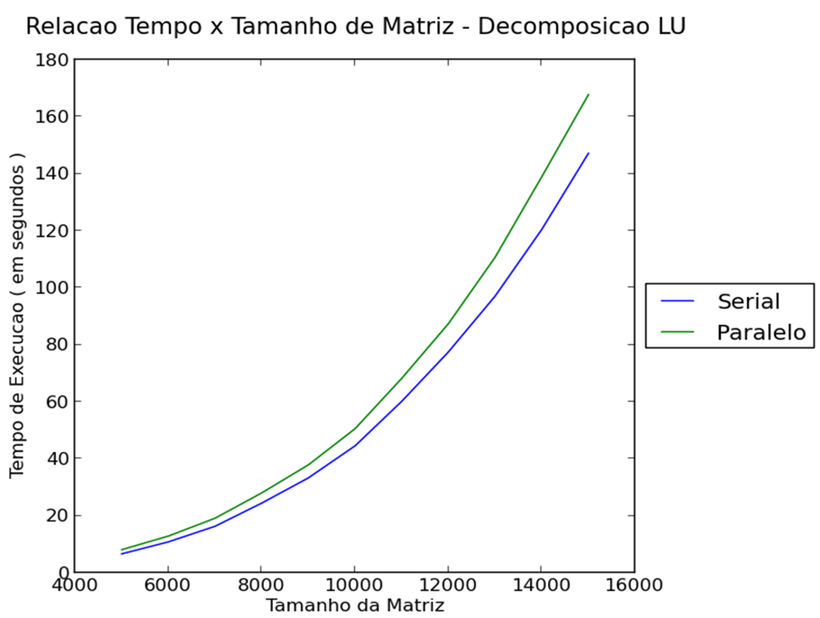
Para o desenvolvimento deste trabalho utilizou-se um computador com as seguintes configurações: 32 Gigabytes de memória RAM, processador AMD FX, com 8 núcleos lógicos (4 físicos), 4.2 GHz, com sistema operacional Linux Fedora 16. Utilizou-se Python na versão 2.7.3, com o interpretador IPython 0.12.1 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.

Resultados e Discussão

As primeiras implementações desenvolvidas visaram descobrir o tempo de execução dos métodos selecionados de forma serial. Os métodos desenvolvidos consistiam na criação de matrizes de números reais que atendessem as necessidades dos problemas , que variassem de 1000 em 1000 e que fossem decompostas através dos métodos de Decomposição LU e Cholesky. Para essa última etapa, utilizou-se as rotinas já implementadas nas bibliotecas Numpy e Scipy.

Posteriormente, houve a adaptação dessas rotinas para execução em paralelo. Para essa adaptação, foram necessárias implementações de rotinas extras que pudessem armazenar os tempos relativos de cada execução. O módulo escolhido para a paralelização dos métodos foi o do próprio interpretador utilizado, o IPython.parallel. Contudo, o resultado da execução em paralelo não foi eficiente, pois houve um aumento de tempo considerável.





Conclusões

Através das pesquisas realizadas por esse projeto e da comparação dos resultados, percebe-se que 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.

Embora a linguagem não tenha sido eficiente na paralelização utilizando suas próprias bibliotecas, resta saber se ela é viável no *wrapping*(envelopamento) de bibliotecas tradicionais para a paralelização de sistemas que demandem um processamento muito alto, e assim, se tornar uma possível opção para o desenvolvimento desse tipo de aplicação.

Referências bibliográficas

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.

SALA, M. et al. - PyTrilinos: High Performance Distributed-Memory Solvers for Python. ACM Transactions on Mathematical Software, Vol. 34, No. 2, Article 7, March 2008.

LANGTANGEN, P. H. - A Primer on Scientific Programming with Python, 2ª ed. 2011