Análise de Performance dos Métodos Implementados

Giuliano Belinassi[1]

[1]USP – Universidade de São Paulo, Instituto de Matemática e Estatística (IME) – São Paulo, Brasil.

giuliano.belinassi@usp.br

Resumo. *Este documento tem como objetivo relatar os testes e conclusões apresentadas pelo programador do Exercício Programa de MAC0300. Conclui-se que implementar um algorítimo orientado a linhas em C possui um desempenho superior que seu equivalente por colunas, assim como a decomposição de Cholesky possui performance superior à decomposição LU.*

1. Informações Gerais

Este documento contém uma analise de certa forma superficial dos algorítimos conhecidos como **Decomposição de Cholesky** e **Eliminação LU com pivoteamento parcial**. Todos estes algorítimos foram implementados em C para o primeiro exercício programa de MAC0300 – Álgebra Linear Numérica.

2. Informações Técnicas

Foram utilizados três computadores para os testes:

- **i5**:
  - Intel Core i5 3210m 2.6GHz, 3M Cache
  - 4GB RAM DDR3
  - Debian 8, com Linux 3.16.0-2-amd64
  - GCC 4.9.2 (64-bits)
- **P4**:
  - Intel Pentium 4 Prescott 3.06GHz, 1M Cache
  - 1GB RAM DDR
  - Debian 8, com Linux 3.16.0-2-i686
  - GCC 4.9.2 (32-bits)
- **K6-2**:
  - AMD K6-2 Mobile 400MHz
  - 160MB RAM DIMM
  - MS-DOS 7.10
  - DJGPP-GCC 4.7.3 (32-bits, modo protegido)

Todos os executáveis foram gerados nas máquinas de teste de dois métodos: com ou sem otimizações de compilação:

- **gcc -Wall -pedantic -ansi -O3 -funroll-loops -march=native -mfpmath=sse -g -o optimized main.c rowimp.c colimp.c matrixio.c -lm**
- **gcc -Wall -pedantic -ansi -g -O0 -o nonoptimized main.c rowimp.c colimp.c matrixio.c -lm**

Porém para o K6-2 em questão, a opção **-mfpmath=sse** foi removida, pois este processador não tem suporte a instruções SSE.

Para o cálculo do tempo gasto pelos testes a função *clock()* da biblioteca *time.h* foi utilizada, que conta o tempo da CPU.

3. O Esperado

Devido ao fato de em C as matrizes serem guardadas por linhas, é de se esperar que ao percorrer uma matriz por linhas providencie uma menor quantidade de *cache-miss*, resultando em menos acessos aleatórios à memória e tornando a performance do programa superior a implementação em colunas.

O suposto problema com implementações que varrem uma matriz por colunas em C seria neste caso um acesso aleatório à memória, resultado em *cache-misses* mais frequentes, o que levaria o programa buscar a informação na memória principal, cuja velocidade de resposta é menor.

Conforme as análises da quantidade de *flops* feitas em aula, também é de se esperar que a decomposição de Cholesky ($n^3/6$) demore, no mínimo, a metade do tempo da decomposição LU ($n^3/3$).

4. Os Resultados

4.1. Cholesky

Foram criados ao todo 10 arquivos para os testes, onde os 9 primeiros seguem o que fora especificado pelo arquivo *genmatsim.c*, e o décimo foi criado usando a especificação do primeiro, porém alterando seu tamanho para 1000.

Os tempos estão na tabela abaixo:

	i5	i5	i5	i5	P4	P4	P4	P4	K6-2	K6-2	K6-2	K6-2
	Linha	Linha	Coluna	Coluna	Linha	Linha	Coluna	Coluna	Linha	Linha	Coluna	Coluna
	Otimiz	Sem ot	Otimiz	Sem ot	Otimiz	Sem ot	Otimiz	Sem ot	Otimiz	Sem ot	Otimiz	Sem ot
Sym_01	0.000446	0.002155	0.001309	0.001269	0.000470	0.001934	0.001257	0.002392	0.000000	0.054945	0.054945	0.054945
Sym_02	0.001473	0.005607	0.011190	0.011582	0.003593	0.014325	0.020773	0.032358	0.109890	0.219780	0.824176	0.879121
Sym_03	0.005285	0.018527	0.037471	0.036288	0.015716	0.050175	0.128663	0.197985	0.274725	0.659341	3.021978	3.186813
Sym_04	0.011795	0.043489	0.095432	0.092387	0.047129	0.121069	0.574884	0.682929	0.714286	1.648352	9.505495	10.109890
Sym_05	0.025473	0.086513	0.208812	0.231493	0.099623	0.244768	1.421025	1.614742	1.318681	3.186813	17.967033	19.175824
Sym_06	0.044158	0.150088	0.381087	0.512409	0.178114	0.432124	2.883153	3.215079	2.417582	5.659341	34.285714	36.318681
Sym_07	0.072098	0.244062	0.623572	0.735858	0.278876	0.685424	5.125897	5.829907	3.846154	9.010989	-	-
Sym_08	0.042681	0.121627	0.365211	0.444160	0.187696	0.379694	3.630022	3.760036	2.032967	4.560440	-	-
Sym_09	0.071276	0.234779	0.625299	0.735489	0.277905	0.681221	5.096305	5.808297	3.846154	8.956044	-	-
Sym_10	0.224673	0.694743	2.105863	3.496420	0.754790	1.940910	18.574272	19.423347	11.263736	26.208791	-	-

Tabela 1: CPU Time dos testes da implementação de Cholesky em seus respectivos processadores

Em todos os testes, o erro foi da ordem de 10^{-11}.

Também é interessante notar que os testes de 8 à 10 simplesmente não demonstraram seu tempo no K6-2 por razões que o autor desconhece, muito embora o arquivo com o resultado correto seja criado.

4.2. Eliminação de Gauss

Foram criados ao todo 10 arquivos para os testes, onde os 9 primeiros seguem o que fora especificado pelo arquivo *genmat.c*, e o décimo foi criado usando a especificação do primeiro, porém alterando seu tamanho para 1000.

Os tempos estão na tabela abaixo.

	i5	i5	i5	i5	P4	P4	P4	P4	K6-2	K6-2	K6-2	K6-2
	Linha	Linha	Coluna	Coluna	Linha	Linha	Coluna	Coluna	Linha	Linha	Coluna	Coluna
	Otimiz	Sem ot	Otimiz	Sem ot	Otimiz	Sem ot	Otimiz	Sem ot	Otimiz	Sem ot	Otimiz	Sem ot
Sym_01	0.000465	0.001770	0.002688	0.002681	0.001133	0.003348	0.003433	0.005924	0.054945	0.054945	0.164835	0.219780
Sym_02	0.002900	0.011294	0.023675	0.021622	0.007883	0.026095	0.057134	0.112211	0.219780	0.439560	1.703297	1.978022
Sym_03	0.008401	0.036616	0.086617	0.074444	0.035417	0.088063	0.473727	0.476663	0.769231	1.318681	6.373626	6.648352
Sym_04	0.019689	0.085500	0.215870	0.223245	0.118851	0.218022	1.829391	1.800508	1.703297	3.131868	16.758242	17.087912
Sym_05	0.039391	0.167269	0.420761	0.551132	0.241453	0.423557	3.897098	4.006284	3.296703	6.318681	35.384615	36.153846
Sym_06	0.072541	0.294402	0.886010	1.254049	0.401206	0.725692	7.050247	7.291011	5.879121	10.769231	64.175824	66.043956
Sym_07	0.122360	0.465343	1.620257	2.078200	0.638721	1.132377	11.142917	11.731669	9.065934	16.923077	104.120879	108.186813
Sym_08	0.124321	0.459857	1.614976	2.105477	0.634518	1.131219	11.127231	11.728306	8.956044	16.868132	103.846154	107.967033
Sym_09	0.122781	0.460378	1.614692	2.070822	0.636860	1.133349	11.123006	11.669063	9.010989	16.868132	103.901099	107.967033
Sym_10	0.417458	1.332812	6.239564	7.584482	1.887248	3.217931	34.757596	37.091612	26.483516	48.901099	304.505495	312.472527

Tabela 2: CPU Time dos testes da implementação de LU em seus respectivos processadores

Em todos os testes, o erro foi da ordem de 10^{-11}.

5. Conclusões

Apesar da tecnologia de processadores e compiladores terem evoluído desde 1998, ainda notamos que a ordem em que as matrizes são percorridas influenciam bastante na performance do programa, e esse resultado é notório quando se compara os resultados do K6-2 ordenados por linhas com os do P4 ordenados por colunas. A orientação de leitura e escrita dos algorítimos devem ser levadas em conta para o desempenho geral da aplicação.

Também é notório que o tempo gasto na decomposição LU é comparável com o dobro do tempo gasto com a decomposição de Cholesky, reforçando a afirmação de que a quantidade de *flops* impacta na perofmance do sistema.

É curioso notar que otimizações do compilador realmente impactam na performance do programa dividindo a complexidade do programa por uma constante.

Referências

Ulrich Drepper, (2007) “What Every Programmer Should Know About Memory”, https://people.freebsd.org/~lstewart/articles/cpumemory.pdf, Agosto.