Desenvolvimento multithread do Epileptogenics Foci Visualizator - EFV

Cleber Roberto Milani, Lucas Ferrari de Oliveira, Gerson Geraldo H. Cavalheiro

Departamento de Informática – Instituto de Física e Matemática
Universidade Federal de Pelotas
Pelotas – RS – Brazil

{cmilani.ifm, lucas.ferrari, gerson.cavalheiro}@ufpel.edu.br

Resumo. *Este artigo apresenta a aplicação de técnicas de paralelismo na área médica, as quais foram adotadas no desenvolvimento do Epileptogenics Foci Visualizator (EFV), uma ferramenta para auxílio ao diagnóstico de epilepsia intratável. O paralelismo foi utilizado com o objetivo de reduzir o tempo de processamento e possibilitar uma interatividade satisfatória do EFV em computadores pessoais de baixo custo. Empregou-se multiprogramação leve, através de threads POSIX, em computadores multi-core, o que permitiu uma redução média em torno de 24% no tempo de processamento.*

1. Introdução

O *Epileptogenics Foci Visualizator* (EFV) é uma ferramenta para auxílio ao diagnóstico desenvolvida por Milani (2007). Tem como finalidade auxiliar profissionais da área médica na avaliação de pacientes com epilepsia intratável. Além disso, visando oferecer uma solução computacional sem custos e que possa ser modificada para atender necessidades locais, o EFV é distribuído na forma de software livre, podendo ser obtido em EFV (2007).

Dentre as características do EFV destaca-se o elevado custo computacional envolvido na execução do algoritmo de reconstrução volumétrica utilizado, o qual é responsável por permitir a visualização interativa em três dimensões (3D) de imagens originalmente 2D (bi-dimensionais). De modo a amenizar o impacto do custo computacional na interatividade do software, utilizou-se programação *multithread*, através da biblioteca *pthreads* (POSIX *Threads*), para exploração do paralelismo em potencial do algoritmo de reconstrução volumétrica. Outra característica importante do EFV é a portabilidade de hardware e software. A ferramenta e as bibliotecas por ela utilizadas estão disponíveis para ambientes MS *Windows*, *Linux* e MAC OS.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os requisitos identificados em relação ao paralelismo e aplicação do mesmo no EFV. As seções 3 e 4 apresentam, respectivamente, os resultados obtidos e as conclusões.

2. Requisitos e Aplicação do Paralelismo

Em geral, a aquisição de dados na medicina produz imagens compostas por fatias paralelas e uniformemente espaçadas. Os volumes para visualização 3D podem, então, ser gerados através do empilhamento dessas fatias, mantendo o espaço original entre elas e mapeando os *pixels* em *voxels*. Isso é feito através da técnica de reconstrução

volumétrica, a qual utiliza um algoritmo de *Ray-casting*, cuja complexidade, em geral, é de $O(n^3)$ [Schroeder et al. 2002].

Aplicações de visualização interativa, ou seja, em que o usuário manipula dinamicamente imagens na tela, exigem uma taxa de resposta e repetição de 10 a 20 *frames* (quadros) por segundo para uma navegação satisfatória [Robb 1999]. Com isso, o custo computacional do *Ray-casting* passa a ser um problema chave no desenvolvimento da aplicação. Entretanto, apesar do custo computacional ser elevado, o mesmo é também paralelizável, pois os valores dos *pixels* são determinados por cálculos independentes entre si. Logo, o algoritmo pode ser explorado em arquiteturas paralelas com ganho de desempenho [Santos 1994; Paiva et al. 1999].

Aplicações semelhantes presentes na literatura, como Manssour et al. (2005), utilizam abordagens do paralelismo voltadas a aglomerados de computadores (*clusters*). No desenvolvimento do EFV, entretanto, optou-se por trabalhar com multiprogramação leve em processadores multi-core. Isso porque, atualmente, os processadores *multi-core* vêm adquirindo *status* de *commodity* e, com isso, computadores pessoais paralelos tornaram-se acessíveis a hospitais, clínicas e mesmo aos médicos. Além disso, diversas bibliotecas de programação *multithread* estão disponíveis gratuitamente para facilitar a exploração do paralelismo em arquiteturas *multi-core*. Logo, explorar o paralelismo inerente ao algoritmo de *Ray-casting* via programação *multithread* em arquiteturas *multi-core*, apresentou-se como uma solução viável para o problema apresentado.

O EFV foi escrito em linguagem C++ e o padrão POSIX *Threads* (*pthreads*) foi adotado como suporte à programação *multithread*. Maiores detalhes a respeito da implementação podem ser obtidos em Milani (2007). O número de *threads* utilizado pela aplicação pode ser alterado, dinamicamente em tempo de execução, através da interface com o usuário, o que garante à ferramenta escalabilidade de software.

3. Avaliação e Resultados

A avaliação de desempenho do EFV considerou o ganho obtido com o uso de recursos de *threads*. Dada a portabilidade do software, utilizou-se na avaliação diferentes configurações de hardware e software. As arquiteturas de hardware utilizadas são apresentadas na Tabela 1. O sistema operacional (SO) em execução durante os testes nas arquiteturas de modelo 1 a 7 é o SO *Windows* XP e, na arquitetura 8, *Mac* OS X 10.4.10. A avaliação completa dos resultados, bem como a descrição dos testes realizados, encontra-se documentada em Milani (2007).

A Tabela 2 apresenta o tempo total gasto na reconstrução dos três volumes que são exibidos na ferramenta, variando o número de *threads* em cada reconstrução. As colunas V1, V2 e V3 representam o número de *threads* utilizados na reconstrução de cada um dos três volumes, onde V1=V2=V3=1 significa que foram executados três threads, um após o outro, enquanto V1=V2=V3=2 indica a execução de seis threads, duas por vez. As colunas A1, A2 etc., referem-se às configurações definidas na Tabela 1. O tempo apresentado é aquele que, efetivamente, o usuário teria de aguardar ao solicitar o carregamento dos exames para visualização na ferramenta, incluindo o consumido por operações de entrada e saída. Embora essas operações não sejam constantemente realizadas durante o processo interativo (como a leitura dos arquivos), todas elas são obrigatoriamente executadas no momento da primeira reconstrução.

Tabela 1. Configurações das arquiteturas utilizadas na fase de testes

Arquitetura	Processador	Cache	Mem. RAM	VGA	Tecnologia
A1	AMD Semprom 2600+ @ 1.84 GHz	256 KB L2	1 x 512 MB 333 MHz	Cirrus Logic 2 MB	Single-core
A2	AMD AthlonXP 2000+ @ 1.67 GHz	256 KB L2	2 x 256 MB 266 MHz	GeForce 5200 128MB	Single-core
A3	Intel Pentium 4 HT @ 3.00 GHz	1 MB L2	1 x 480 MB 333 MHz	Onboard 32MB	Hyper Threading
A4	Intel Pentium D 805 @ 2.66 GHz	2 x 1 MB L2	1 x 496 MB 333 MHz	Onboard 16MB	Dual-core
A5	Intel Core 2 Duo E6300 @ 1.86 GHz	2 MB L2	2 x 1 GB 667 MHz	GF 7600GS 256 MB	Dual-core
A6	Intel Core Duo T2300 @ 1.66 GHz	2 MB L2	2 x 448 MB 667 MHz	Onboard 128MB	Dual-core
A7	AMD Turion 64 X2 TL-50 @ 1.60 GHz	2 x 256 KB L2	1 x 896 MB 667 MHz	Onboard 128MB	Dual-core
A8	PowerPC G4 @ 1.5 GHz	512 KB L2	2 x 512 MB 333 MHz	ATI Mob. Radeon	Single-core

Tabela 2. Tempo médio gasto na reconstrução dos três volumes em arquiteturas variadas, incluindo operações de E/S.

Threads			Tempo total de reconstrução em segundos							
V1	V2	V3	A1	A2	A3	A4	A5	A6	A7	A8
1	1	1	6,557	7,182	5,475	5,308	3,037	3,919	5,237	21,201
1	1	2	6,550	7,196	5,447	5,052	2,861	3,662	4,897	21,298
1	2	1	6,528	7,220	5,316	4,721	2,689	3,426	4,600	21,363
1	2	2	6,534	7,224	5,284	4,467	2,510	3,175	4,248	21,336
2	1	1	6,524	7,231	5,378	5,030	2,891	3,709	4,911	21,355
2	1	2	6,542	7,239	5,356	4,785	2,707	3,457	4,592	21,345
2	2	1	6,554	7,278	5,232	4,452	2,536	3,222	4,287	21,357
2	2	2	6,553	7,310	5,191	4,205	2,361	2,980	3,965	21,003
4	4	4	6,580	7,314	5,206	4,187	2,413	3,052	4,038	21,061

Nas arquiteturas *multi-core* utilizadas, em geral, o resultado mais satisfatório foi obtido ao reconstruir os volumes com 2 *threads* cada. Apenas a arquitetura A4 não seguiu o padrão, obtendo resultados melhores com o uso de 4 *threads* por volume. A redução aproximada do tempo de processamento nesse caso foi de 21% e o *speedup* de 1,27. Entretanto, observa-se que os ganhos com o uso de 2 *threads* por volume foram bastante semelhantes. A arquitetura A8 apresentou o desempenho menos satisfatório do grupo, porém, até o momento, não foram analisados os motivos que levaram a esse

resultado. Na configuração A6, a redução no tempo de processamento, comparando-se a execução seqüencial (execução com 1 *thread*) com a *multithread,* foi de 24% com *speedup* igual a 1.31 e eficiência de 65%. A arquitetura A7, por sua vez, apresentou redução de 25% no tempo de processamento, *speedup* de 1.32 e eficiência igual a 66%.

A comparação entre o desempenho das arquiteturas A1 e A2 comprova que não há necessidade de hardware gráfico especial para execução do EFV. Realizou-se ainda a compilação da aplicação em sistemas operacionais variados para verificação da portabilidade, a qual foi atestada nos sistemas MS *Windows* XP, *Debian Linux* e MAC OS X 10.4.10.

4. Conclusão

A exploração do paralelismo inerente ao algoritmo de *Ray-casting,* dividindo-o em fluxos independentes de execução, trouxe ganho de desempenho em arquiteturas dotadas de processadores com tecnologia *Hyper Threading* e processadores *multi-core*, enquanto naquelas com processador *single-core* a variação no tempo de execução foi desprezível. Assim, a solução apresentada demonstra-se viável, pois atende o requisito em relação à interatividade. Além disso, pode ser vista como vantajosa em relação àquelas que utilizam aglomerados de computadores, pois executa satisfatoriamente em computadores de uso geral, ou seja, não precisa de hardware específico para utilização.

Outras características apresentadas pelo EFV são a escalabilidade de software e a portabilidade. Por permitir manipular, dinamicamente em tempo de execução, o número de *threads,* caso o número de processadores das arquiteturas seja incrementado, pode-se configurar o software para explorá-los eficientemente através da divisão em um número maior de fluxos de execução. Já a portabilidade do software foi atestada através da sua execução em ambientes MAC OS X, *Debian Linux* e MS *Windows* XP.

Referências

EFV (2007) “EFV” http://epileptogenicsf.sourceforge.net/, Acesso em 10 dez.

Manssour, I., Fernandes, L., Freitas, C., Serra, G., Nunes, T. (2005) “High performance approach for inner structures visualisation in medical data”, *IJCAT*, v. 22, p.23-33.

Milani, C. (2007) “Ferramenta multithread para reconstrução volumétrica de imagens em medicina nuclear para auxílio na detecção de zonas epileptogênicas”. 129f. Trabalho acadêmico (Graduação em Ciência da Computação) – Instituto de Física e Matemática, Universidade Federal de Pelotas, Pelotas.

Paiva, A., Seixas, R. e Gattass, M. (1999) “Introdução à Visualização Volumétrica”. 107f. Trabalho acadêmico (Graduação em Ciência da Computação) - Departamento de Ciência da Computação, PUC-Rio, Rio de Janeiro.

Robb, R. (1999) “Visualization in biomedical computing”, *Parallel Computing,* v.25, Issue 13-14, p.2067-2110.

Santos, E. (1994) “Avaliação do algoritmo de Ray Tracing em Multicomputadores”, 181p. Dissertação (Mestrado) – Universidade de São Paulo, São Paulo.

Schroeder, W., Martin, K., and Lorensen, B. (2002) “The Visualization Toolkit - An Object Oriented Approach to 3D Graphics”, Kitware Inc., 3rd ed.