Processamento Paralelo Aplicado a um Problema de Engenharia de Materiais

Paulo Jorge Quaresma
U.Aberta & DI-FCT,
Univ. Nova Lisboa
pjq@di.fct.unl.pt

Pedro Duarte Medeiros
D.Informática, FCT,
Univ. Nova Lisboa
pm@di.fct.unl.pt,

Alexandre Velhinho
D.Ciênc.Materiais, FCT,
Univ. Nova Lisboa
ajv@fct.unl.pt

Resumo

A Tomografia de raios X é uma técnica que tem sido utilizada em muitas áreas. Os aparelhos usados produzem uma cada vez maior quantidade de dados em bruto e os investigadores têm vindo a aumentar a diversidade de informações que pretendem extrair deles. Neste trabalho, partiu-se de um programa sequencial de processamento de dados tomográficos desenvolvido pelo investigador G. Vignoles, da Universidade de Bordéus 1, que aplica aos dados em bruto vários tipos de processamento. Com o objectivo de diminuir o tempo de execução dos passos mais longos, começou-se pela optimização de determinadas partes do código, em particular, diminuindo o número de chamadas às bibliotecas de sistema e melhorando o aproveitamento da cache. Seguidamente partiu-se para a paralelização das operações mais demoradas, em que se usou um esquema de paralelização geométrica dos dados, o que permitiu ganhos significativos de desempenho. Neste esforço a abordagem mais escalável foi aquela em que se usou um multiprocessador de memória distribuída, tendo sido usada a biblioteca de troca de mensagens MPI-2, incluindo as primitivas de entrada e saída paralelas.

Palavras-chave: tomografia, multiprocessadores de memória distribuída, biblioteca MPI-2, sistema de ficheiros em paralelo.

Abstract

X-ray tomography is a technique that is being widely used in many areas. In one hand, the available tomographic devices produce massive amounts of data; on the other hand the researchers continually demand more information to be extracted from the raw data. In this work, the starting point was a sequential program developed by G. Vignoles of Bordeaux 1 University that includes several processing steps of the raw tomographic data. In order to reduce the execution time of the longer processing steps, different approaches have been tried like the optimization of some parts of sequential code (mainly those related with I/O operations), and the use of distributed-memory multiprocessors (clusters), where the use of a parallel file system for data sharing between the nodes was combined with a geometrical parallelization approach. In the last approach the MPI-2 library (including the parallel I/O primitives) was used.

Keywords: tomography, distributed-memory multiprocessors, MPI-2 library, parallel file system.

1. Microtomografia de Raios X

A microtomografia de raios-X com radiação de sincrotrão (SXMT) constitui um desenvolvimento recente da tomografia computorizada (CT), técnica bem estabelecida no

domínio da imagiologia médica, mas que apenas num passado relativamente recente foi aplicada com sucesso ao estudo dos materiais.

O princípio da tomografia computorizada consiste na medição da distribuição espacial de uma determinada grandeza física do objecto estudado, examinada a partir de diferentes direcções, e no cálculo, a partir desses dados, de imagens livres de sobreposições. Neste caso é utilizado o bombardeamento por radiação X de alta intensidade – radiação de sincrotrão - correspondendo a grandeza física medida à sua intensidade após absorção parcial pelo objecto de estudo.

Os dados decorrentes da tomografia consistem em ficheiros em bruto de grandes dimensões. É frequente que sejam gerados ficheiros com muitos megas ou mesmo gigabytes. Devido ao tamanho destes ficheiros, os requisitos a nível de poder computacional para o processamento destes dados, tendo por objectivos o melhoramento qualitativo das imagens (redução de ruído, optimização de contraste) e a quantificação de diferentes parâmetros estereológicos, são muito elevados. O processamento de uma imagem com cerca de 600 MB pode demorar mais de 24 horas.

No caso em análise, o objecto de estudo corresponde a materiais compósitos com uma matriz de alumínio com reforço de partículas de carboneto de silício. O objectivo específico da análise das imagens tomográficas consiste na determinação de parâmetros de distribuição espacial dos reforços (número, forma dimensão e localização das partículas). A aplicabilidade destas ligas na indústria automóvel, aeronáutica e aeroespacial tem crescido bastante nos últimos anos, devido à sua boa relação dureza-peso. A determinação da morfologia interna da liga, assim como da distribuição espacial dos reforços e respectiva orientação, é de extrema importância nesta área.

As imagens tomográficas são constituídas por um conjunto espacialmente coordenado de pontos elementares (*voxel*), correspondendo cada um no espaço tridimensional a cerca de 1 μm^3; a cor de cada *voxel,* um valor inteiro entre 0 e 255, constitui o atributo registado pelo ficheiro representativo da imagem. O processamento completo da imagem envolve a análise de muitos milhões de *voxeis*, o que se torna muito pesado em sistemas monoprocessador. Com o advento da computação paralela e das técnicas de programação paralela é possível paralelizar o algoritmo que processa as imagens de forma a aumentar significativamente o seu desempenho.

No que concerne à aplicação da microtomografia ao estudo de materiais, foram anteriormente desenvolvidos alguns trabalhos. Um destes trabalhos foi efectuado por G. Vignolles [1] e consistia na caracterização da população de reforços na forma de fibras de carbono dispersos numa matriz de carbono amorfo. Neste trabalho foi desenvolvido um programa que permite obter uma ideia da estrutura interna da liga e dos reforços. Este software pegava no volume de dados produzido à saída do sincrotrão e fazia um processamento do mesmo, com o objectivo primordial de obter o índice granulométrico da amostra.

Um trabalho que envolvia a paralelização da tomografia foi realizado por um conjunto de investigadores do Departamento de Computação da Universidade da Califórnia [2]. A execução da tomografia paralela pode ocorrer em dois cenários, um *offline* e outro *online*. No primeiro o utilizador recorre a dados que se encontram em memória secundária (foi esta também a abordagem seguida neste trabalho). Na tomografia *online* os dados são processados à medida que saem do tomógrafo. Esta opção permite trabalhar quase em tempo real. Neste

trabalho os professores Henry Casanova, Shava Smallen e Francine Berman apresentam um cenário de tomografia paralela *online* recorrendo para o efeito aos recursos da *Grid*.

Para os autores existem dois parâmetros que condicionam a qualidade da tomografia *online*. O primeiro denominado (f) é um escalar que especifica a redução do tamanho de uma projecção em cada dimensão. O segundo denominado (r) corresponde ao número de novas projecções processadas por cada refrescamento do tomograma. O trabalho desenvolvido por esta equipa de investigadores consistia em determinar um par (f,r) ideal para os recursos computacionais disponíveis.

2. Metodologia Utilizada

No trabalho apresentado neste artigo partiu-se de um programa denominado *tritom*, desenvolvido por G. Vignolles [1] da Universidade de Bordéus. O *tritom* é um programa sequencial escrito em linguagem C, que suporta vários tipos de tratamentos dos dados tomográficos. Seguidamente, descreve-se de forma resumida o seu funcionamento.

No início do processamento, o ficheiro produzido pelo tomógrafo é carregado para memória central, passando a ser visto pelo programa como uma matriz a três dimensões, em que cada elemento da matriz é um valor entre 0 e 255, sendo o 0 interpretado como preto e o 255 como branco; os valores intermédios correspondem a níveis de cinzento. Esta imagem não pode ser usada directamente pois existem problemas relacionados com fenómenos de interferência e indefinição de fronteiras. Após o processamento bem sucedido, a matriz ficará branca (255) e os reforços pretos (0).

O programa *tritom* disponibiliza um menu de opções, que permitem ao utilizador especificar o processamento que deseja ver executado. Das principais opções disponibilizadas podemos destacar as seguintes, usando-se na designação das fases a terminologia introduzida no artigo de G. Vignolles [1]:

- *A bisegmentação da imagem*: A conversão da imagem para três níveis de cor apenas: branco (255), preto (0) e cinzento (127). A aplicabilidade desta função corresponde a situações em que o contraste entre as fases que se pretendem identificar é reduzido, como resultado da ocorrência de valores pouco diferenciados para as densidades da matriz e do reforço, como efectivamente sucede nos sistemas carbono/carbono e alumínio/carboneto de silício. As regiões assim identificadas deverão em principio corresponder a zonas de incerteza na vizinhança imediata das fronteiras matriz/reforço e serão posteriormente sujeitas a uma operação de *histerese*.
- *A limpeza da imagem*: Neste processamento procura-se eliminar ruído, correspondendo à ocorrência de *voxeis* pretos isolados em regiões brancas e vice-versa.
- *A histerese*: Nesta fase toma-se uma decisão sobre os *voxeis* cinzentos que possuem vizinhos pretos ou brancos (i.e., aqueles que foram identificados durante a operação de bisegmentação), decidindo-se se os *voxeis* cinzentos pertencem a uma partícula (cinzento passa a preto), ou se pertencem à matriz (cinzento passa a branco). Esta decisão é feita analisando o número de vizinhos de cada classe e pintando de acordo com a maioria; no caso de os vizinhos serem todos cinzentos o *voxel* não é alterado. A *histerese* pode ser aplicada repetidamente até todos os *voxeis* cinzentos se terem tornado pretos ou brancos.
- *A percolação da imagem*: Esta fase de processamento centra-se numa entidade que o *tritom* chama *blob*. Define-se por *blob* um conjunto contíguo de *voxeis* da mesma cor. Um *blob* tem um interior e uma pele, sendo esta última o conjunto dos *voxeis* exteriores. Esta

função vai marcar o interior e a pele dos blobs cinzentos. Inicialmente esta função determina uma caixa para delimitação do *blob*. Numa segunda fase, os *voxeis* que estão na pele são pintados de preto ou branco de acordo com a cor maioritária na vizinhança.

- *As estatísticas de granulometria*: Nesta operação determina-se o número, posição e dimensão dos reforços na matriz de base do compósito. Esta operação identifica *blobs* pretos, determinando para cada um, o seu volume, centro de massa, superfície e as dimensões de um paralelepípedo rectângulo que o contém.

Para permitir uma visualização dos resultados obtidos por cada fase optou-se por tornar cada fase isolada das restantes. Cada fase usa um ficheiro de entrada com a organização atrás descrita e produz uma nova matriz que é guardada num ficheiro de saída. A Figura 1 ilustra esta abordagem.

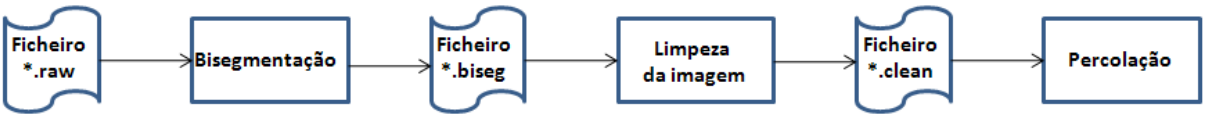

Figura 1 - Interligação entre as várias fases usando ficheiros

Com o objectivo de diminuir o tempo de execução de algumas das fases, procurou-se, neste trabalho, optimizar partes do programa. As principais optimizações realizadas corresponderam a:

- Mantendo o programa sequencial, optimizar algumas operações.
- Paralelização dos tratamentos mais demorados.

A nível da versão sequencial foram realizadas algumas alterações, nomeadamente:

- A forma de percorrer a matriz em memória principal foi reformulada de forma a diminuir o número de falhas (*misses*) no acesso à *cache* do CPU.
- Foram igualmente realizadas alterações na operação de escrita e leitura da imagem de forma a reduzir o número de chamadas ao sistema.

Na parte da paralelização, foram escolhidas as duas operações mais demoradas: a limpeza da imagem, e as estatísticas de granulometria. Foram exploradas arquitecturas de memória partilhada e de memória distribuída. Em ambos os casos optou-se por uma estratégia de divisão de trabalho entre os vários processadores baseada na geometria do problema (paralelização geométrica).

Isto significa que cada processador tem a seu cargo o processamento de uma região diferente da amostra, isto é, da matriz a 3 dimensões (ver Figura 3). Para a exploração de multiprocessadores de memória partilhada recorreu-se à biblioteca das *Posix Threads* [3]; no caso dos multiprocessadores de memória distribuída, foi usada a biblioteca de troca de mensagens *Message Passing Interface* (MPI) [4].

3. Optimização da versão sequencial

O algoritmo de tomografia atrás descrito não foi totalmente paralelizado, visto que algumas das funções têm um tempo de execução relativamente curto, e não era justificável a paralelização de todas as funções. Nesta secção são descritas as optimizações feitas em algumas das operações de modo a aumentar o seu desempenho sem no entanto se ter recorrido à paralelização da mesma.

3.1. Leitura e escrita da imagem

Tomando como exemplo uma imagem de 700*700*700, verifica-se que o tempo de leitura é superior a 1 minuto e o tempo de escrita é cerca de 3,5 minutos. Após alterar a forma de leitura e escrita das imagens, diminuindo o número de chamadas à biblioteca standard do C, obtêm-se tempos de leitura/escrita muito mais reduzidos - por exemplo, a mesma imagem de 700*700*700 consome cerca de 4 segundos na leitura e cerca de 11 segundos na escrita. Estes resultados foram obtidos no sistema Ext2 do Linux e os resultados obtidos no sistema PVFS2 são muito semelhantes a estes, sendo a variação inferior a 1 segundo.

3.2. Percurso da imagem

Como atrás foi referido a imagem da amostra é uma matriz tridimensional e os vários tratamentos percorrem todos os *voxeis* da imagem. Tirando partido do conhecimento da forma como os dados estão organizados em memória, é possível modificar a estratégia de percurso da imagem de forma a garantir que acessos consecutivos à imagem correspondem a um percurso sequencial da memória física, assegurando assim uma muito melhor utilização da *cache*.

A Figura 2 ilustra os resultados obtidos para a operação de bisegmentação da imagem, antes e depois da optimização do acesso à memória realizada.

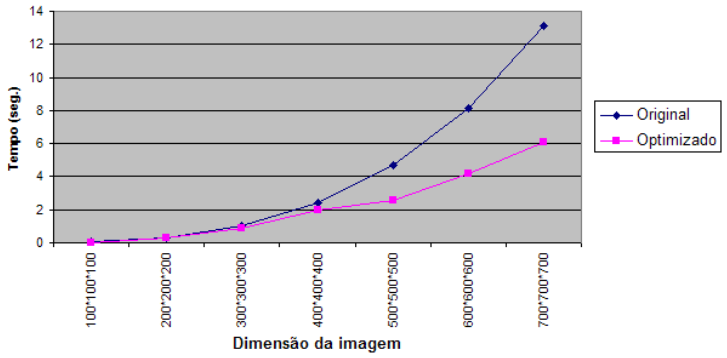

Figura 2 - Tempos da operação de bisegmentação

4. Paralelização Geométrica

A estratégia usada para dividir o trabalho entre os vários processadores disponíveis foi a paralelização geométrica. O paralelepípedo que representa a imagem foi dividido em tantas partes quanto o número de processos, ficando cada processo responsável pelo tratamento de um volume separado. A Figura 3 ilustra a abordagem seguida.

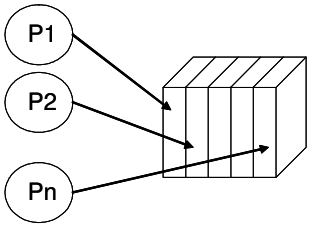

Figura 3 - Arquitectura da solução implementada

No caso da tomografia, alguns dos passos identificam e processam partículas isoladas. Quando uma partícula se situa em duas fatias distintas da imagem, os processos que tratam cada fatia necessitam de comunicar entre si. Este problema tende a agudizar-se à medida que aumentamos o número de processos. A Figura 7 ilustra esta situação, onde existem três partículas cortadas pelas fronteiras.

Da análise anterior pode-se concluir, que em termos de paralelização há dois tipos de tratamento:

- Nenhuma comunicação entre os processos: Cada processo trata a sua parte sem necessidade de trocar informação com os restantes. É uma situação do tipo "embaraçosamente paralelo", que acontece na fase de limpeza da imagem. Neste caso será de esperar uma aceleração linear.

- Necessidade de comunicação entre processos: Nas estatísticas de granulometria os vários processos trocam mensagens com o objectivo de identificar partículas seccionadas pelas fronteiras. A experiência mostrou que esta fase pesa relativamente pouco no tempo total da operação, pelo que as acelerações conseguidas se aproximam da linearidade.

Claro que para um valor de N muito grande, se pode colocar a questão da fatia atribuída a cada processo ser demasiado pequena, o que faz com que os custos de arranque de cada processo pesem significativamente no tempo total.

5. Paralelização usando Pthreads

Para mostrar as potencialidades do uso de multiprocessadores de memória partilhada, foi desenvolvida uma versão da operação de limpeza da imagem, em que esta é partida em varias zonas, sendo cada zona da responsabilidade de um processo leve. Estava disponível uma máquina com as seguintes características:

- Hardware: 2 CPUs Intel Xeon, os quais possuem a tecnologia *simultaneous hyperthreading (SMT)*. Como é sabido, isto corresponde a cada CPU ter um certo número de componentes duplicado, nomeadamente os registos.

- Software: Sistema Linux 2.4.21 ELsmp com suporte de multiprocessador indicando o núcleo que estão disponíveis 4 processadores.

O gráfico da Figura 4 mostra as acelerações conseguidas para uma imagem de 600*600*600.

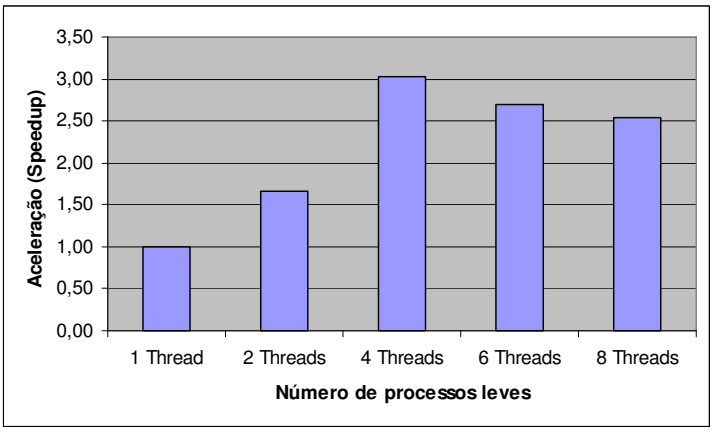

Figura 4 – Spedup's da operação de limpeza da imagem (threads)

A aceleração máxima consegue-se usando 4 processos leves. Isto não é surpreendente, uma vez que corresponde ao número de CPUs disponíveis. Como se trata de uma situação em que não há perdas de tempo relacionadas com a sincronização de processos, seria de esperar uma aceleração linear. Tal não acontece uma vez que, com a tecnologia *hyperthreading* num CPU com 2 fluxos de execução, a maior parte dos componentes (Unidade Aritmética e Lógica, *Cache* …) não são duplicados. Esta experiência mostra que só com hardware com mais CPUs (bastante mais dispendioso) seria possível obter melhores acelerações. Isto sugere que uma arquitectura de memória distribuída tem potencial para conseguir maiores acelerações com uma melhor relação preço-desempenho.

6. Paralelização usando MPI

Nesta secção é indicado o trabalho desenvolvido na paralelização das duas fases do algoritmo de maior duração (limpeza da imagem e a operação de estatísticas de granulometria) usando um multiprocessador de memória distribuída. Em ambos os casos, a paralelização foi feita usado o MPI e a divisão de trabalho pelos vários processadores baseou-se na geometria do problema a resolver. Nestas experiências foi usado um agregado (*cluster*) composto por 4 nós interligados por interfaces *Gigabit Ethernet*. Em cada nó, existe um processador AMD de 2.2 GHZ e o sistema de operação é o Linux (Scientific Linux 4.2 – Kernel 2.6.9). Foi utilizada a implementação *openmpi* [8].

Na estratégia de paralelização geométrica, quase todos os processamentos correspondem ao seguinte esquema simplificado:

```
Para cada processo fazer
    (1) Ler para RAM uma fatia da matriz que resultou do tratamento
    anterior
    (2) Aplicar o processamento a esta fatia
    (3) Escrever num ficheiro a fatia da matriz processada
Fim-fazer
```

Como atrás foi explicado, cada fase do processamento recebe um ficheiro de grandes dimensões como entrada e produz um ficheiro de igual dimensão como saída. Quando a paralelização envolve vários nós que não têm acesso a discos partilhados é preciso assegurar um acesso eficiente aos ficheiros e evitar a sua replicação nos vários nós.

Como o MPI tem primitivas para acesso coordenado por múltiplos processos ao mesmo ficheiro, decidiu-se que as fases (1) e (3) utilizariam essas primitivas. Como a configuração hardware usada não tem discos partilhados, foi necessário simular essa partilha através de um sistema de ficheiros paralelo/distribuído. Uma discussão das diferenças entre estes dois tipos de sistemas de ficheiros não pode ser feita aqui por limitações de espaço, mas pode-se com alguma simplificação dizer que os primeiros se focam mais nos aspectos relacionados com a transparência da localização dos dados, e com questões como a tolerância a falhas, e a coerência das *caches*, enquanto os segundos estão mais vocacionados para o aumento da velocidade no acesso aos dados.

As opções possíveis para suportar o MPI-IO, na configuração hardware disponível e na versão do MPI usada limitavam-se ao sistema de ficheiros distribuído *Network File System* (NFS) [5], e ao sistema de ficheiros paralelo *Parallel Virtual File System* (PVFS) [6]. As experiências feitas e a consulta de diversas referências [10] levaram-nos a optar por usar o sistema de ficheiros paralelo PVFS2 [7].

6.1. Limpeza da imagem

A estratégia usada nesta fase já foi atrás descrita. Seguiu-se a tradicional organização em *mestre/escravos*: Os processos *escravos* são responsáveis pelas operações de processamento de cada parte da imagem e existe um processo *mestre* que coordena as operações. Seguindo a abordagem SPMD (*Single Program Multiple Data*) subjacente ao MPI, o processo com *rank* 0 é o *mestre* e os restantes são *escravos*. A Figura 5 ilustra a abordagem seguida.

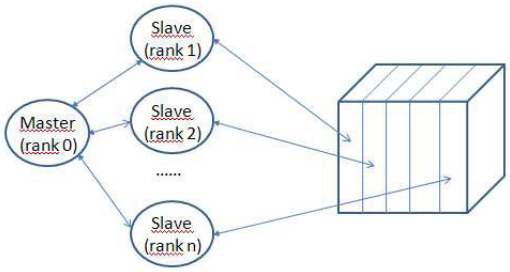

Figura 5 - Arquitectura da solução implementada

Note-se que, nesta operação, como atrás foi dito, não há comunicação entre os processos que tratam de cada fatia, pelo que é de esperar uma aceleração quase linear.

A Figura 6 ilustra os resultados obtidos para várias dimensões de imagens e para a configuração de hardware descrita acima; dadas as características de CPU intensivo do processamento, executa-se um processo por CPU. Numa imagem de grandes dimensões como a de 700*700*700, a versão sequencial demorou cerca de 62 horas (quase 3 dias a limpar a imagem), enquanto na versão paralelizada esse tempo foi reduzido para cerca de 20 horas, ou seja menos de 1 dia. A análise do gráfico demonstra que se obtiveram valores da aceleração (*speedup*) na casa dos 3,5.

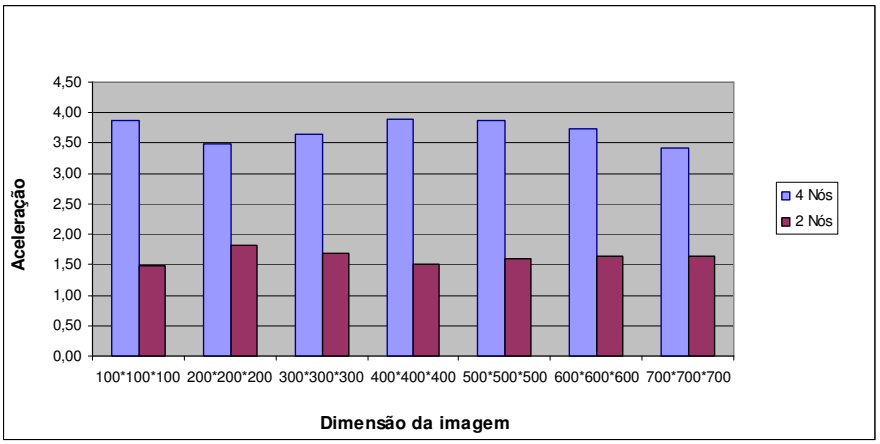

Figura 6 – Speedup's da operação de limpeza da imagem (MPI)

O hardware disponível não permitiu estudar a escalabilidade para mais nós, mas dada a natureza do problema é de esperar uma aceleração quase linear, uma vez que não há necessidades de comunicação entre os processos, nem desequilíbrios de carga.

6.2. Estatísticas de Granulometria

A operação de estatísticas de granulometria é a mais demorada de todo o processo e como tal foi paralelizada, de forma a reduzir os seus tempos de execução. Esta operação foi descrita atrás e o seu resultado final é um ficheiro de texto, onde, para cada partícula detectada são calculadas: as coordenadas do centro de massa, volume, superfície e as coordenadas dos vértices e o tamanho das arestas de um paralelepípedo que delimita a partícula no espaço tridimensional.

É mais uma vez usada uma estratégia de paralelização em que cada processo trata uma fatia separada. Como atrás foi assinalado, há uma dificuldade se uma partícula é cortada pela fronteira entre duas fatias. Imaginemos uma partícula que se situa em duas fatias distintas da imagem, logo cada um dos processos verá a mesma partícula como se fossem duas partículas diferentes, e de dimensão mais reduzida (ver Figura 7).

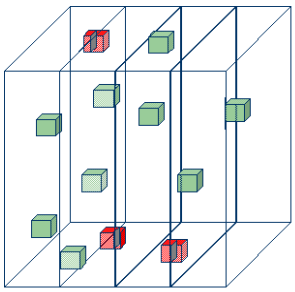

Figura 7 - Partículas cortadas pela fronteira

Uma forma de dar resposta a este problema é, após o processamento da granulometria terminar, analisar e fundir os vários ficheiros produzidos procurando ligar partículas que foram fraccionadas pela paralelização. Para tal foi desenvolvida uma solução que, a partir da dimensão da caixa delimitadora de cada partícula e do seu ponto de partida, verifica se a partícula foi cortada pela fronteira e, caso isso tenha ocorrido, vai analisar o ficheiro de *blobs* seguinte de forma a determinar a caixa delimitadora que daria continuidade à anterior. Esta

segunda fase do processamento tem uma duração da ordem dos segundos, o que é completamente desprezável face à duração da fase anterior (horas). A tabela seguinte apresenta os tempos totais e as acelerações obtidas na operação de granulometria para dois ficheiros de imagem.

Tabela 1- Tempos Totais e Aceleração

	Sequencial Tempo (horas)	**2 Nós** Tempo (horas) Aceleração		**4 Nós** Tempo (horas) Aceleração	
Imagem de 300*300*300	1,81	1,11	1,63	0,59	3,06
Imagem de 400*400*400	9,24	4,94	1,87	2,83	3,26

7. Conclusões

Com este trabalho conseguiu-se melhorar significativamente os tempos de processamento de dados resultantes da tomografia, quer com multiprocessadores de memória partilhada, quer com multiprocessadores de memória distribuída.

Os processamentos efectuados são do tipo CPU-intensivo, tendo as operações de entrada/saída um peso muito reduzido no tempo de processamento. Por outro lado, usando uma estratégia de paralelização geométrica, conseguem-se acelerações próximas da linearidade, o que sugere o uso do maior número de CPUs possível. Com estas premissas, o uso de agregados de computadores (*clusters*) parece a solução com melhor relação preço-desempenho.

Ao usar o MPI para a exploração de um *cluster* a programação das aplicações fica simplificada ao serem utilizadas as primitivas de I/O paralelo do MPI; o suporte destas primitivas sobre o sistema de ficheiros PVFS2 permite uma solução eficiente evitando a duplicação dos ficheiros de dados. Por outro lado, o PVFS2 tem o inconveniente de não ter qualquer suporte para tolerância a falhas, isto é, se um dos nós falhar, toda a informação fica indisponível, mesmo que se situe em nós que continuam a funcionar.

8. Trabalho futuro

As previsíveis evoluções deste trabalho situam-se em duas áreas:

- *Introdução de novos tipos de processamento*: na área dos materiais compósitos estão permanentemente a ser introduzidos novos tipos de reforços e de técnicas para os difundir; daqui resulta, a necessidade de novos tipos de processamento a fazer à informação recolhida pelos tomógrafos.
- *O melhoramento da interface de utilização*: seria muito útil associar a este programa facilidades que permitissem a um engenheiro de materiais configurar uma sequência de tratamentos a fazer. A escolha deverá poder ser feita usando uma ferramenta gráfica que permitisse escolher os tratamentos e os seus parâmetros a partir de um menu. Associada a esta ferramenta, deveria haver facilidades de visualização que permitissem observar os resultados de cada um dos processamentos efectuados. Isto corresponderia a um embrião de um ambiente de resolução de problemas (PSE – *Problem Solving Environment*) dedicado ao processamento de imagens tomográficas.

Bibliografia

[1] G.L. Vignoles (2000), *Image segmentation for phase-contrast hard X-ray CMT of C/C composites*, Carbon **43**, no. 2.

[2] F.Berman, S. Smallen e H. Casanova (2001), *Applying scheduling and tuning to on-line parallel tomography*, Actas da conferência "Supercomputing", ACMPress.

[3] D. Towsley, T. Wagner (1995), *Getting started with POSIX threads*, Relatório Técnico, Department of Computer Science - University of Massachusetts.

[4] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra (1996), *MPI: The Complete Reference vol. 1,* MIT Press.

[5] RFC 1094 - NFS: Network File System Protocol specification.

[6] W.B. Ligon III, R. Latham, R.B. Ross, P.H. Carns (2002), *Using the Parallel Virtual File System*, Relatório Técnico, Graduate School of Clemson University.

[7] H. Ramachandran (2002), *Design and Implementation of the System Interface for PVFS2*, Tese de Mestrado, Graduate School of Clemson University.

[8] R.L. Graham, T.S. Woodall, J.M. Squyres (2005), *Open MPI: A Flexible High Performance MPI,* Actas da conferência "Parallel Processing and Applied Mathematics".

[9] W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir (1998), *MPI: The Complete Reference vol. 2,* MIT Press.

[10] M.C.L. Carri (2004), *Análisis de performance y optimización en cluster Beowulf*, Tese de Mestrado, Universidade de Buenos Aires.