Pesquisa e Tecnologia
MINERVA

# BIBLIOTECA CIENTÍFICA DE PROCESSAMENTO DE SINAIS DISTRIBUÍDA APLICADA EM ULTRA-SOM

**Felipe Sanches Zanoni**

Graduando em Engenharia da Computação,
EESC-USP, e-mail: fsz.feliz@gmail.com

**Paulo Rogério Scalassara**

Doutorando em Engenharia Elétrica, EESC-USP

**Carlos Dias Maciel**

Professor-Doutor do Departamento de Engenharia Elétrica,
EESC-USP, e-mail: maciel@sel.eesc.usp.br

**José Carlos Pereira**

Professor Titular do Departamento de Engenharia Elétrica, EESC-USP

## Resumo

Neste trabalho, diversos ensaios são realizados para testar a eficiência do processamento distribuído em um *cluster* de computadores usando a técnica MPI com diversos algoritmos: cálculo de números primos, Transformada de Fourier e sua inversa e cálculo da convolução de dois sinais pelo método direto e pela FFT. O objetivo é começar o desenvolvimento de uma biblioteca para processamento de sinais biológicos em *cluster*. Inicialmente, apresenta-se um *software* para estimar o espectrograma de sinais simulados de ultra-som usando a Transformada de Fourier de Tempo Reduzido. Futuros trabalhos acrescentarão mais funcionalidades à biblioteca, aumentando sua aplicação em processamento de sinais.
*Palavras-chave:* mpi, ultra-som, processamento distribuído, *cluster*.

## Introdução

Os *softwares* de processamento de sinais requerem grande capacidade de processamento. Normalmente, essa capacidade está associada a *hardwares* cada vez mais rápidos usando sistemas operacionais com características como: simplicidade, desempenho e disponibilidade.

Com a evolução dos computadores, essa capacidade aumentou radicalmente, porém, em muitos casos, ainda não é satisfatória para um grande volume de dados. Programas com longo tempo de execução, em muitas situações práticas, não são tolerados quando é necessária maior interatividade com o usuário.

Visando a melhor desempenho, pode-se organizar o *software* e/ou *hardware* dos computadores para realizarem as tarefas em conjunto, assim, a capacidade de processamento de cada um é somada, formando os chamados *clusters* (Bader & Pennington, 2001). Essa formação resulta em grande capacidade de processamento preservando o mesmo *hardware* (tanto dos computadores quanto das instalações de rede). Então, pode-se processar grande quantidade de informações em menos tempo e a um custo mais baixo se comparado aos custos de supercomputadores capazes de realizar o mesmo cálculo.

Os *clusters* de computadores são cada vez mais utilizados em pesquisas científicas e em qualquer outro tipo de atividade que necessite de grande processamento de dados (Pacheco, 1997). Para utilizar um *cluster* é necessário conhecimentos de computação paralela, a qual consiste em subdividir os problemas computacionais em outros menores, podendo executá-los em vários processadores ao mesmo tempo, com aproveitamento máximo de cada um deles. Ou seja, somam-se seus poderes computacionais simulando um supercomputador.

A técnica de processamento paralelo adotada neste trabalho é o MPI (*Message-Passing Interface*, ou seja, interface de passagem de mensagem), definida em MPI Forum, 1995. Essa abordagem não é uma nova linguagem de programação, mas uma biblioteca de funções que podem ser usadas pela linguagem C. As funções buscam o paralelismo do programa através de passagem de mensagens entre as máquinas que fazem parte do *cluster*, ou seja, a transmissão de dados entre processos.

De acordo com Pacheco (1997), com a passagem de mensagens, pode-se criar programas paralelos muito eficientes, entretanto, esse é um método difícil de ser usado, pela quantidade de detalhes a serem observados. Felizmente, dependendo da capacidade de modularização do algoritmo, ou seja, subdividi-lo em partes menores, o processo de programação não é tão complexo. Além disso, graças ao grande uso atual de MPI, já existem algoritmos encapsulados com soluções prontas para problemas tradicionais.

A utilização de MPI também é possível em DSPs (*Digital Signal Processors* – processadores digitais de sinal), de acordo com Martín *et al.* (2004). Nesse trabalho, usam-se DSPs da *Texas Instruments* (http://www.ti.com) para montar um *cluster* para processamento de sinais. Já no trabalho de Gallego *et al.* (2005), busca-se uma alternativa para as bibliotecas com direitos reservados de DSPs utilizando MPI, obtendo os resultados desejados com portabilidade, tamanho reduzido e rapidez.

Processamento paralelo é extremamente vantajoso na resolução de problemas que dependem do resultado de muitas equações, como é o caso de sistemas de convecção-difusão, os quais são comuns nas áreas de hidráulica e finanças. Chau (2007) apresenta uma abordagem para esse tipo de problema com base em técnicas de decomposição junto com um método de sobreposição de domínios na resolução das equações algébricas envolvidas. Isso é implementado em um sistema distribuído usando MPI, o qual apresentou bons resultados.

O uso de computação paralela está aumentando muito em processamento de sinais biológicos, pois, geralmente, o volume de dados para analisar é muito grande. Como exemplo, tem-se o trabalho de Tillett & Phillips (2004), o qual, ao analisar sinais de eletroencefalograma (EEG) usando um *cluster*, objetiva melhorar e acelerar o acesso dos profissionais de saúde aos resultados de exames de pacientes. Os sinais desse trabalho citado foram obtidos do Programa de Compreensão de Epilepsia (*Comprehensive Epilepsy Program*) da Universidade de Rochester, nos Estados Unidos. Os resultados foram satisfatórios e motivaram a continuação dos estudos.

Outro caso em que é necessário grande volume de processamento é a procura de informações em bancos de dados biológicos que armazenam seqüências de DNA. Esse é um problema comum ultimamente devido ao avanço das pesquisas ligadas a seqüenciamento de DNA. O trabalho apresentado por Battré & Angulo (2006) mostra claramente como o emprego de processamento distribuído usando MPI pode ser eficiente nesse tipo de pesquisa.

Como apresentado, a área de processamento de sinais precisa analisar um grande volume de dados. Assim, este trabalho visa desenvolver uma biblioteca para processamento distribuído em um *cluster* usando MPI aplicada a sinais de ultra-som. A pesquisa seguiu a mesma linha apresentada no trabalho de Shinoda (2005).

## Materiais e Métodos

Os algoritmos foram desenvolvidos usando a linguagem C++ em ambiente Linux, distribuição Gentoo (http://www.gentoo.org). A implementação MPI utilizada foi o LAM-MPI (*Local Área Multicomputer*), o qual está disponível para a distribuição Gentoo. O LAM-MPI é uma implementação em código livre (http://www.lam-mpi.org) a qual é mantida pelo Laboratório de Sistemas Abertos (*Open Systems Lab*) na Universidade de Indiana nos Estados Unidos. Além de fornecer as bibliotecas necessárias para a implementação dos programas distribuídos, o LAM também é um ambiente para a execução dos programas em forma de serviço (*daemon*), o que facilita o desenvolvimento (LAM/MPI Team, 2007). Outra vantagem dessa implementação é o suporte às versões MPI-1.2, a grande parte da MPI-2 e também a *clusters* heterogêneos, ou seja, formados por máquinas de diferentes tipos de processadores (32 e 64 bits).

O sistema Gentoo Linux foi escolhido, primeiramente, pela possibilidade de otimização do *kernel* (o núcleo do sistema operacional), mas também pela ferramenta de instalação de pacotes *Portage*. Essa ferramenta possibilita a fácil instalação de novos aplicativos no sistema sem que grandes configurações sejam necessárias. Para isso ela guarda no computador uma árvore com os dados dos possíveis aplicativos que podem ser instalados, a qual deve ser sempre sincronizada com o servidor da base de dados.

Uma ferramenta muito útil em processamento de sinais de ultra-som é a Transformada de Fourier (FT – *Fourier Transform*), especificamente a Transformada de Fourier de Tempo Reduzido (STFT – *Short Time Fourier Transform*) (Fish, 1990). O melhor algoritmo para implementar a FT é a Transformada Rápida de Fourier (FFT – *Fast Fourier Transform*), conforme Oppenheim *et al.* (1999). Assim, para a sua implementação em C++, utilizou-se o *software* FFTW (*Fastest Fourier Transform in the West*), disponível no *website* http://www.fftw.org/, versão 3.1.

Para avaliar o desempenho do *cluster*, foram feitos quatro testes: 1. para um teste inicial, criou-se um programa simples para cálculo de números primos; 2. usou-se um algoritmo para a realização da FT e sua inversa de um sinal; 3. estimou-se convolução direta de dois sinais; e 4. a convolução usando a FT. O algoritmo utilizado para o cálculo de números primos é baseado no fato de que todo número é primo se ele não for divisível por nenhum outro número entre 2 e a metade dele mesmo.

O objetivo da avaliação é determinar sob qual configuração o *cluster* consegue melhor desempenho executando os testes apresentados. Assim, foi medido o tempo de execução de cada programa: quanto menor esse tempo, melhor o desempenho do *cluster*.

Os computadores utilizados no processamento dos dados fazem parte do *cluster* do Laboratório de

Processamento de Sinais Biológicos (LPSB) do Departamento de Engenharia Elétrica da Escola de Engenharia de São Carlos (EESC) da Universidade de São Paulo (USP). Esse *cluster* é composto por quatro máquinas: dois AMD Athlon 64 3000+, processador de 64 bits, com freqüência de *clock* igual a 2 GHz e 1024 MB de memória RAM DDR com freqüência de 400 MHz canal duplo, chamadas Máquinas 1 e 2; um AMD Athlon XP 2800+, processador de 32 bits, com freqüência de *clock* de 2,08 GHz e 1024 MB de memória RAM DDR com freqüência de 400 MHz canal duplo, chamada Máquina 3; e um AMD Turion 64 X2 TL-50, 2 processadores de 64 bits, com freqüência de *clock* igual a 1,6 GHz por núcleo e 1024 MB de memória RAM DDR2 com freqüência de 533 MHz canal duplo, chamada Máquina 4.

Os fatores a serem analisados nos testes são três: o número de processadores, a faixa de números que são processados no servidor antes de analisar os clientes e a faixa de números entregue pelo servidor, para cada cliente, em cada requisição. Para o primeiro fator, têm-se a variação de 1 a 5 processadores; para o segundo, têm-se três faixas: 4000, 8000 e 16000; e para o terceiro, também se têm três faixas: 5000, 10000 e 15000. A configuração das máquinas usadas no *cluster* será mantida fixa, assim, fatores como memória e velocidade individual dos processadores não serão levados em consideração nesses testes.

O sistema de avaliação é do tipo fatorial parcial. Foram realizados vários testes, com quase todas as possíveis combinações dos fatores. Entretanto, não se configura como fatorial completo, pois, com o uso de apenas um processador, não é necessário fazer os testes com as três faixas de números entregues ao cliente, uma vez que, nesse caso, haverá apenas um. Nos demais casos, todas as possíveis combinações serão testadas, para verificar qual a melhor combinação e evitar erros que poderiam ser causados pela variação de apenas um fator isoladamente. A Figura 1 apresenta um resumo das variações desses fatores.

Para mais de dois processadores, todas as combinações entre os valores dos fatores serão testadas. Em cada um dos casos, serão realizados cinco testes, obtendo-se a média destes, resultando uma resposta mais confiável. Dessa forma, ao todo, serão realizados 195 testes para determinar a combinação de fatores que leva o *cluster* ao melhor resultado, de acordo com a quantidade de processadores utilizada.

***Teste com números primos***

A variável de resposta do *cluster* é o tempo de execução do *software* que faz o cálculo de números primos no intervalo de 0 a 1300000. Esse tempo varia de acordo com as distribuições das faixas de números do servidor aos clientes, e conforme a quantidade de mensagens trocadas entre as máquinas. Quanto mais números o servidor processa, menos mensagens ele irá trocar com os clientes. Entretanto, se ele processar uma faixa grande, os clientes vão ficar ociosos por muito tempo, esperando até que ele termine o processamento e lhes dê uma nova faixa de valores. Cria-se, assim, uma dependência entre a quantidade de mensagens trocadas e o tamanho do intervalo de números a serem calculados pelo servidor e pelos clientes.

O programa é bastante simples, ele calcula quantos números primos existem em uma faixa de números de 0 (zero) a X, sendo X qualquer número do tipo inteiro de 32 bits. O *software* servidor é executado em uma das máquinas e os outros processos, chamados de clientes, são executados em zero ou mais máquinas.

A divisão do processamento foi feita de forma que o servidor envie aos clientes, quando requisitado, o intervalo de números a serem calculados. Quando acabam esses cálculos, o cliente envia um sinal para o servidor indicando o término do processamento e requisitando novos dados. O servidor, por sua vez, executa também cálculos para intervalos de números, os quais são diferentes dos tamanhos dos intervalos dos clientes, conforme a Figura 1. Cada vez que o intervalo do servidor é finalizado, ele verifica se há mensagens de término e requisições dos clientes. Caso haja alguma mensagem, ele libera um novo intervalo de números para esse cliente.

| Número de processadores | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Faixa de números do servidor | 4.000<br>8.000<br>16.000 | 4.000<br>8.000<br>16.000 | 4.000<br>8.000<br>16.000 | 4.000<br>8.000<br>16.000 | 4.000<br>8.000<br>16.000 |
| Faixa de números do cliente | 10.000 | 5.000<br>10.000<br>15.000 | 5.000<br>10.000<br>15.000 | 5.000<br>10.000<br>15.000 | 5.000<br>10.000<br>15.000 |

**Figura 1** Variações dos fatores que serão usados nos testes de resposta do *cluster*.

***Transformada de Fourier e inversa***

Neste teste, objetivou-se verificar o desempenho do *cluster* com dados um pouco mais reais. O sinal processado, uma senóide com 8192 amostras, é analisado 400 vezes para aumentar a carga do sistema. O processo consiste em estimar a Transformada Rápida de Fourier desse sinal e, depois, obtê-lo novamente usando a Transformada Inversa de Fourier.

No teste anterior, já foi analisado o custo-benefício entre processamento do servidor e dos clientes, assim, neste teste, buscou-se analisar a influência dos tipos de processadores do *cluster*. Cinco configurações foram usadas, considerando as máquinas apresentadas anteriormente: a) Máquinas 1, 2 e 3; b) Máquinas 1 e 3; c) Máquina 1 somente; d) Máquina 3 somente; e e) Máquinas 1 e 2. Lembrando que as Máquinas 1 e 2 possuem processadores de 64 bits e a Máquina 3, processador de 32 bits. Um resumo dessas configurações é apresentado na Tabela 1.

O servidor envia para os clientes quais intervalos dentre os 400 sinais devem ser processados, utilizando um intervalo de 50 sinais. Os clientes apenas recebem esses intervalos e fazem os cálculos, sendo descartadas as trocas de mensagens para unirem novamente os resultados, pois o objetivo é apenas observar o ganho de desempenho no processamento desses dados. Foram realizadas dez execuções de cada configuração, para obter um valor médio.

***Convolução usando Transformada de Fourier***

A convolução entre dois sinais de tempo discreto, x[n] e h[n], é dada pela Equação (1). Essa equação fornece a forma direta de calcular a convolução de dois sinais, mas outra forma de calcular é pela FT, como mostra a Equação (2), sendo $\Im\{.\}$ a FT (Oppenheim *et al.,* 1999; Donnelly & Rust, 2005). Essa equação mostra que a convolução pode ser calculada como a FT inversa da multiplicação das FT dos sinais.

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \quad (1)$$

$$x[n] * h[n] = \Im^{-1}\{\Im\{x[n]\} \cdot \Im\{h[n]\}\} \quad (2)$$

Então, neste teste, o qual é bem parecido com o anterior, a idéia é utilizar a FFT para realizar a operação de convolução entre dois sinais: uma senóide e um trem de pulsos retangulares. Como anteriormente, são realizadas 400 operações com sinais de 8192 pontos. Para cada configuração são realizadas 10 execuções do programa, resultando em 50 no total. As configurações utilizadas do *cluster* foram as mesmas do teste anterior.

***Convolução usando forma direta***

Este teste tem o propósito de comparar os métodos de cálculo da convolução entre dois sinais. Mas como este método de cálculo utiliza multiplicações matriciais, foi necessário diminuir o tamanho dos sinais para 4096 pontos, a fim de evitar problemas de alocação excessiva de memória. As configurações utilizadas do *cluster* foram as mesmas dos testes anteriores.

Para este teste, foram realizadas 5 execuções do *software* para cada configuração do *cluster*, sendo, ao todo, 20 execuções. A diminuição do número de execuções ocorreu porque o tempo de resposta é muito lento para esse tipo de processamento, o qual envolve muitas operações matriciais.

***Transformada de Fourier de Tempo Reduzido (STFT)***

Depois desses testes iniciais, partiu-se para o projeto de um aplicativo capaz de gerar diagramas tempo/freqüência de sinais de ultra-som. Esse *software* suporta *cluster* de computadores totalmente heterogêneo com o apoio da plataforma LAM-MPI. A aplicação se baseia na Transformada de Fourier de Tempo Reduzido (STFT). Para usar essa transformada, é necessário definir a janela utilizada na transformação e o tamanho do deslocamento para um novo cálculo. Para que o resultado fosse um gráfico automaticamente gerado pelo programa, utilizou-se a ferramenta Gnuplot (Williams *et al.*, 2007).

Para validar o *software* foi necessário um sinal conhecido, sendo escolhido um sinal de freqüência variável, iniciando por alguns instantes em 20 Hz e aumentando linearmente até 20 kHz. A estimativa da STFT é apresentada na Figura 2. Foi utilizada uma janela quadrada de 1024 pontos e deslocamento entre janelas de 1024 pontos também.

**Tabela 1** Resumo das cinco configurações utilizadas nos testes.

| Condição | Descrição |
|---|---|
| 1 | 2 Athlon 64 3000+ e 1 Athlon XP 2800+ |
| 2 | 1 Athlon 64 3000+ e 1 Athlon XP 2800+ |
| 3 | 1 Athlon 64 3000+ |
| 4 | 1 Athlon XP 2800+ |
| 5 | 2 Athlon 64 3000+ |

Para testar o desempenho do algoritmo no *cluster* foi calculada a STFT de um arquivo de áudio com 7155333 pontos e usou-se uma janela quadrada de 1024 pontos com deslocamento de 1024 entre as janelas, resultando em 6987 janelas. O resultado desse teste em três condições (somente a Máquina 1, Máquinas 1 e 2 e Máquinas 1, 2 e 3, isto é, 1, 2 e 3 processadores) foi 172,2; 101,4; e 62,6 segundos, respectivamente. Assim, percebe-se que o tempo de resposta melhora com o aumento do número de processadores do *cluster*.

## Resultados e Discussões

### *Teste com números primos*

Foram feitos vários testes com cálculo de números primos de 1 a 1300000, como mostra a Figura 1. Para consolidar os resultados, foram criados dois gráficos de tempo de resposta do *cluster versus* o número de processadores usados (Figuras 3a e b). No primeiro gráfico foi mantida constante a faixa de números passada aos clientes, 10000, variando-se a faixa processada pelo servidor: 4000, 8000 e 16000 números. No segundo gráfico, ao contrário, manteve-se constante a faixa do servidor, 4000, e variou-se a dos clientes: 5000, 10000 e 15000 números.

Pode-se concluir por esse experimento que, com apenas um processador (Figura 3a), quanto maior a carga no servidor, mais rápida é a análise dos números, pois serão feitas menos trocas de mensagens, liberando o processador para realizar os cálculos. Com relação ao uso de 2 até 4 processadores (com máquinas heterogêneas, 32 e 64 bits), a eficiência aumenta conforme aumenta o número de processadores. Com relação à faixa de números em cada máquina, quanto maior as cargas nos clientes, Figura 3b, mais eficiente será o desempenho do *cluster*. Porém, se o intervalo calculado pelo servidor for maior do que o intervalo dos clientes, o desempenho será menor, pois o *cluster* fica menos eficiente pela espera do servidor em processar seu intervalo para então passar o novo intervalo de números aos clientes.

### *Transformada de Fourier e inversa*

Para o teste da FT e FT inversa, os tempos de resposta do *cluster* para as diferentes configurações de máquinas são apresentados na Figura 4.

Pode-se perceber que, ao comparar o sistema executando apenas em uma das máquinas (condições 3 e 4), a Máquina 1 (Athlon 64, condição 3) teve desempenho bastante superior. Isso se deve ao fato de esse processador possuir um conjunto de instruções mais novas que a versão Athlon XP (Máquina 3, condição 4), além de ter processamento de 64 bits. O FFTW utiliza funções do tipo *SSE, MMX*, nas quais o Athlon 64 é mais eficiente.

Outro fato interessante foi que, no caso de mistura das duas arquiteturas diferentes, percebeu-se que as máquinas de 32 bits processavam muito menos dados do que as de 64 bits. Isso se deve ao fato de que para a comunicação entre uma máquina de 64 bits com uma de 32 bits é preciso fazer a conversão dos dados. Essa conversão pode levar um tempo razoável para o sistema em questão.

O melhor desempenho do *cluster* foi para a configuração com a combinação de duas máquinas Athlon 64, condição 5. Apesar de os intervalos de confiança estarem sobrepostos, notou-se que, se fossem colocados mais computadores Athlon 64 no *cluster*, o desempenho tenderia a melhorar.

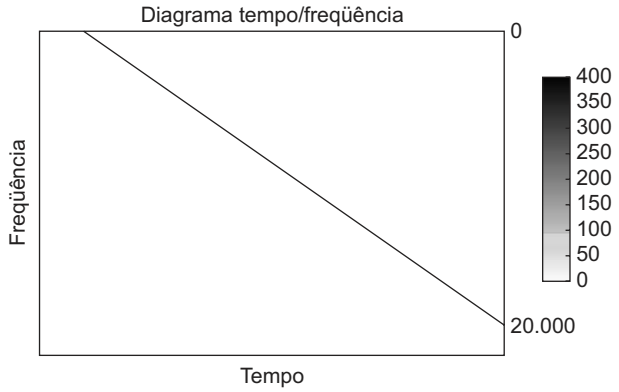


**Figura 2** Resultado da estimativa da STFT de um sinal de freqüência variável, de 20 Hz a 20 kHz, usando uma janela de 1024 pontos e deslocamento de 1024 pontos também.

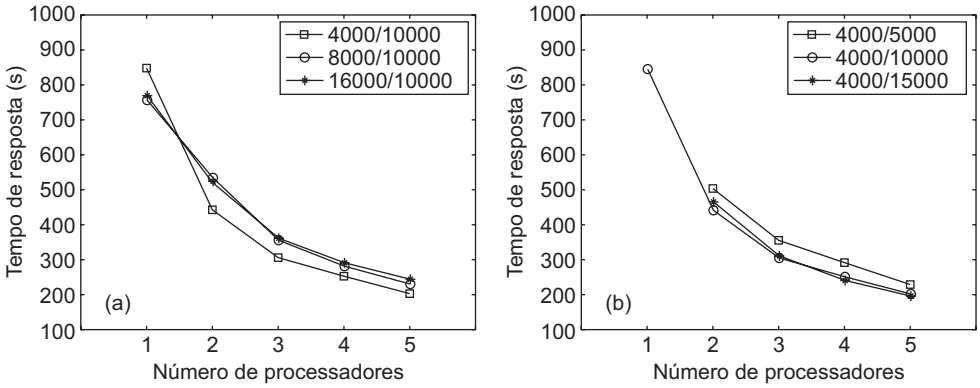


**Figura 3** Tempo de resposta do *cluster* para cálculo de números primos de 1 a 1300000. (a) Variação da faixa de números processada pelo servidor, mantendo a faixa dos clientes em 10000 números. (b) Variação da faixa de números processada pelos clientes, mantendo a faixa do servidor em 4000 números.

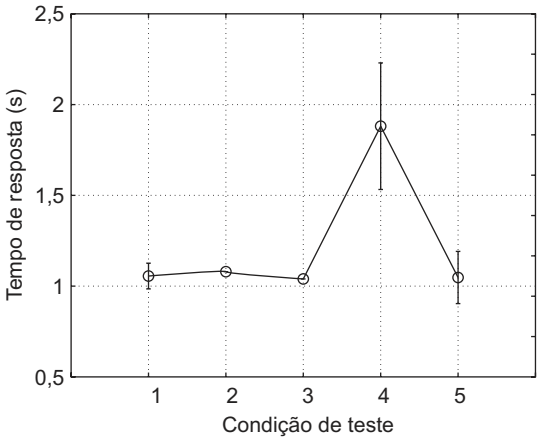


**Figura 4** Tempos de resposta do *cluster* para o teste do cálculo da FT e FT inversa para os 400 sinais senoidais de 8192 pontos.

***Convolução usando Transformada de Fourier***

Para o teste da estimativa da convolução de dois sinais usando a FT, a Figura 5 apresenta os resultados. Observou-se que os melhores resultados foram para as configurações 1, 2 e 5. Com um computador apenas, o Athlon 64 é mais eficiente, o que, como já comentado, se deve às instruções mais modernas desse tipo de máquina.

***Convolução usando forma direta***

Os resultados para o cálculo das 5 execuções de 400 convoluções de dois sinais de 4096 pontos para as 5 configurações apresentadas na seção Materiais e Métodos são mostrados na Figura 6.

Pelos resultados apresentados na figura, percebe-se que esse tipo de algoritmo não é eficiente. O tempo de execução é muito superior se comparado ao teste anterior. Dividindo a tarefa entre vários computadores, nota-se que o desempenho aumenta consideravelmente, porém não o suficiente para ser mais rápido que a convolução por FFT.

Um fato interessante foi que, nesse caso, os computadores com processadores de 32 bits foram mais rápidos. Isso pode ser explicado pelo fato de todos os cálculos não fazerem uso das instruções adicionais do Athlon 64 e o processamento dos produtos matriciais dependerem muito mais da freqüência de *clock* efetiva do computador do que do seu conjunto de instruções. A

freqüência de *clock* efetiva do Athlon XP é maior que a do Athlon 64 no *cluster* apresentado.

***Transformada de Fourier de Tempo Reduzido (STFT)***

Alguns sinais de ultra-som gerados por um simulador apresentado em Scalassara (2005) foram usados para testar o algoritmo de STFT no *cluster*. A Tabela 2 apresenta as características desses sinais, considerando o nível de ruído branco adicionado (em decibéis) e o *jitter* em porcentagem. Mais informações sobre essas características podem ser obtidas em Scalassara (2005). Os sinais possuem 8192 pontos, sendo usada janela de 1024 pontos e deslocamento de 1024 pontos também.

As STFT desses sinais foram geradas usando o *software* desenvolvido; os gráficos do primeiro e último sinais são apresentados nas Figuras 7a e b. Esses gráficos são interessantes porque mostram como o *jitter* dificulta a análise precisa do sinal, em decorrência do espalhamento da energia causado pela variação da freqüência (Scalassara *et al.*, 2007).

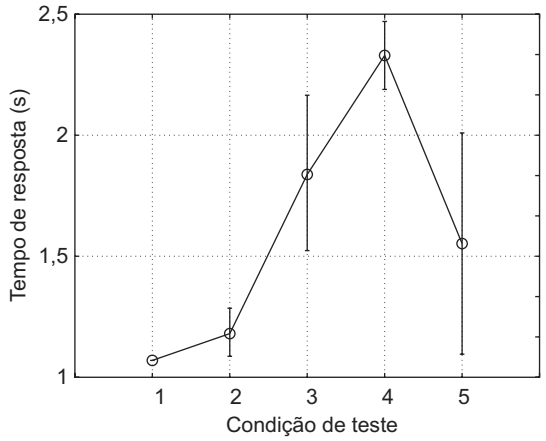


**Figura 5** Tempos de resposta médios do *cluster* para o teste de 10 execuções do programa de cálculo de 400 convoluções entre dois sinais de 8192 pontos usando a FFT.

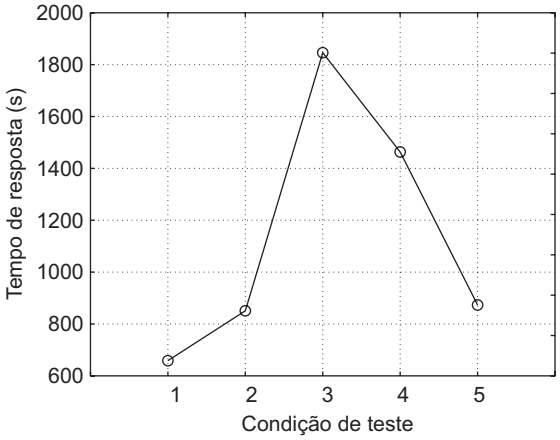


**Figura 6** Tempos de resposta médios do *cluster* para o teste de 5 execuções do programa de cálculo de 400 convoluções entre dois sinais de 4096 pontos usando o método direto.

**Tabela 2** Características dos sinais simulados de ultra-som usados para testar o algoritmo de STFT no *cluster*.

| Sinal | Jitter | Ruído |
|---|---|---|
| 1 | 1% | 15,72 dB |
| 2 | 3% | 15,59 dB |
| 3 | 5% | 15,87 dB |
| 4 | 10% | 15,27 dB |
| 5 | 20% | 15,45 dB |

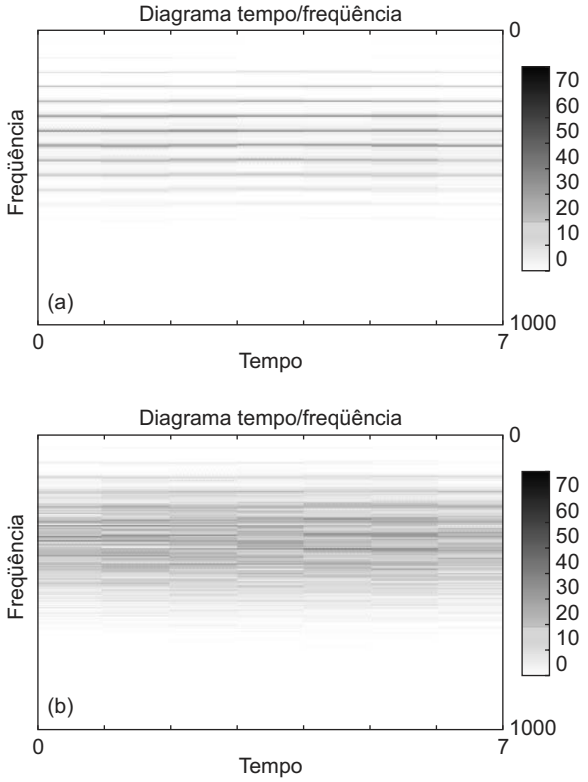


**Figura 7** (a) STFT do sinal de ultra-som simulado com jitter 1% e nível de ruído branco em 15,72 dB obtido usando o algoritmo no cluster. (b) STFT do sinal simulado com *jitter* 20% e nível de ruído igual a 15,45 dB.

## Conclusão

É perceptível o ganho de desempenho obtido com o uso de *cluster* de computadores. Esse ganho não é diretamente proporcional ao número de computadores usados. A experiência do programador é bastante relevante, visto que programação concorrente depende muito de quem escreve o código. Um aplicativo que faz muita leitura e escrita de arquivos também terá seu desempenho reduzido. A divisão de processamento em *cluster* diminuirá o tempo de resposta apenas para processamento, lembrando também que as trocas de mensagens entre os computadores é outro fator a se considerar quando se compara desempenho em cluster.

Sobre a aplicação desenvolvida para cálculo de STFT de sinais de ultra-som, os resultados foram o esperado. Ela é capaz de processar sinais em ambientes de *cluster* heterogêneo de computadores com ganho de performance satisfatório. Utilizando apenas dois computadores foi possível processar um sinal com 10 segundos de duração, um total de 7155333, com uma janela de 1024 pontos e um total de 6987 janelas, em aproximadamente um minuto. Enquanto que com apenas um computador esse tempo era de quase três minutos.

Foram também implementadas rotinas de FFT, com o uso da biblioteca FFTW versão 3.1, e convolução direta e indireta (com uso da FFT) de sinais, obtendo resultados satisfatórios nos testes. Todas as rotinas são compatíveis com *cluster* de computadores com o uso da biblioteca LAM-MPI, que implementa as funcionalidades do *Message Passing Interface* (MPI).

## Agradecimentos

Os autores agradecem os auxílios recebidos da FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) para a realização desta pesquisa, os quais são relacionados ao processo número 06/51484-4.

## Referências Bibliográficas

BADER, D. A.; PENNINGTON, R. Cluster computing: applications. *The International Journal of High Performance Computing*, v. 15, n. 2, p. 181-185, 2001.

BATTRÉ, D.; ÂNGULO, D. S. MPI framework for parallel searching in large biological databases. *Journal of Parallel and Distributed Computing*, v. 66, n. 12, p. 1503-1511, 2006.

CHAU, M. et al. MPI implementation of parallel subdomain methods for linear and nonlinear convection-diffusion problems. *Journal of Parallel and Distributed Computing*, v. 67, n. 5, p. 581-591, 2007.

DONNELLY, D.; RUST, B. The fast Fourier transform for experimentalists. Part II. Convolutions. *Computing in Science & Engineering*, v. 7, n. 4, p. 92-95, 2005.

FISH, P. *Physics and instrumentation of diagnostical medical ultrasound*. New York: John Wiley & Sons, 1990.

GALLEGO, J. A. R.; MARTÍN, J. C. D.; LLORENTE, J. M. A. An MPI implementation for distributed signal processing. In: 12th EUROPEAN PVM/MPI USERS' GROUP MEETING (EuroPVM/MPI2005). *Lecture Notes in Computer Science (LNCS)*. Springer-Verlag, 2005. v. 3666/2005, p. 475-483.

LAM/MPI Team. *LAM/MPI User's Guide* – Version 7.1.4. Open Systems Lab, 2007. Disponível em http://www.lam-mpi.org.

MARTÍN, J. C. D. et al. On interface design for distributed signal processing. In: 12th EUROPEAN SIGNAL PROCESSING CONFERENCE, EUSIPCO, 2004, *Proceedings...* 2004. v. 1, p. 1365-1368.

MPI Forum. *MPI:* a message-passing interface standard. Technical Report. Knoxville: University of Tennessee, 1995. Disponível em http://www.mpi-forum.org.

OPPENHEIM, A. V.; SCHAFER, R. W.; BUCK, J. R. *Discrete-time signal processing*. 2nd ed. New York: Prentice-Hall, 1999. 870 p.

PACHECO, P. S. *Parallel programming with MPI*. San Francisco: Morgan Kaufmann Publishers, Inc., 1997.

SCALASSARA, P. R. *Análise de sinais de ultra-som usando decomposição autorregressiva e rastreamento de pólos.* 2005. 137 f. Dissertação (Mestrado) – Departamento de Engenharia Elétrica, Universidade Estadual de Londrina, Londrina.

SCALASSARA, P. R. et al. Autoregressive decomposition and pole tracking applied to vocal fold nodule signals. *Pattern Recognition Letters*, v. 28, n. 11, p. 1360-1367, 2007.

SHINODA, A. A. Biblioteca científica de processamento de sinais distribuída. *Semina: Ciências Exatas e Tecnológica*, v. 26, n. 1, p. 11-16, 2005.

TILLETT, J. C.; PHILLIPS, D. B. Biomedical signal visualization and analysis system development utilizing open source software and a computer cluster. In: 30th ANNUAL NORTHEAST BIOENGINEERING CONFERENCE, 2004, *Proceedings...* 2004. v. 1, p. 83-84.

WILLIAMS, T et al. *Gnuplot*: An interactive plotting program. Version 4.2, 2007. Disponível em http://www.gnuplot.info.