Paralelização da Simulação da Trajetória de Elétrons em um Dispositivo FED

Márcio Castro e Ricardo Piccoli

20 de outubro de 2006

Resumo

A necessidade por soluções computacionais vem crescendo cada vez mais. Diversas áreas científicas como Matemática, Física e Biologia necessitam de extremo poder computacional para resolução de problemas e simulações. Neste contexto, arquiteturas paralelas de baixo custo (*clusters*) mostram-se uma alternativa viável para reduzir o tempo de execução destas aplicações. Neste trabalho será apresentada uma solução de paralelização de uma ferramenta de simulação de trajetória de elétrons em um dispositivo Mostrador de Emissão de Campo. Esta ferramenta tem como principal objetivo auxiliar na construção deste novo tipo de dispositivo, afim de prever seu funcionamento antes de construí-lo. Porém, todo o processo de simulação de trajetórias necessita alto desempenho. Este fato impulsiona o estudo de uma solução paralela visto que o tempo de resposta da aplicação não é aceitável em arquiteturas monoprocessadas.

1 Introdução

Com o crescente desenvolvimento das linhas de *hardware* e *software*, o suporte da computação aplicada às mais diversas áreas como Física, Biologia ou Matemática tem se tornado indispensável na resolução de problemas. Há necessidade de desenvolver técnicas de programação e arquiteturas de computadores que forneçam embasamento para o desenvolvimento de ferramentas adequadas à complexidade dos problemas atuais. Simulação de sistemas físicos e bioquímicos, resolução de equações e visualização de dados através de imagens de três dimensões, são exemplos de aplicações da computação aplicada. Em casos de simulação [1], o tempo de uma única execução pode ser aceitável, mas, geralmente, inúmeras execuções são necessárias para que se obtenha um resultado preciso, testando diferentes possibilidades. Visualização de dados, como renderização de moléculas ou volumes de dados médicos, exigem grande esforço computacional para a geração de uma imagem em três dimensões. Quando o objetivo é a geração de uma animação [2] ou suporte ao usuário [3], que requisita periodicamente novas perspectivas de um mesmo objeto, o problema se agrava pois inúmeras imagens devem ser geradas para que se obtenha interação em tempo real.

Arquiteturas monoprocessadas, ou que não ofereçam suporte à manipulação de grandes quantias de dados, não são capazes de resolver os problemas anteriormente citados em tempo viável. Processamento de alto desempenho é uma alternativa que visa resolver as limitações impostas por essas arquiteturas convencionais, distribuindo uma execução para mais de uma unidade de processamento. Não somente o tempo de execução pode ser reduzido consideravelmente, mas uma maior quantia de dados pode ser manipulada.

Neste trabalho é proposta uma implementação paralela de uma ferramenta de simulação das trajetórias de elétrons em um dispositivo mostrador de informação (display) de emissão de campo ou FED (Field Emission Display). Mostradores do tipo FED são dispositivos de exibição, como mostradores LCD (Liquid Cristal Display) e CRT (Cathode Ray Tube). Sua tecnologia revolucionária permite que imagens de alta qualidade, como as geradas por um CRT, sejam possíveis com o baixo consumo de energia e redução considerável de tamanho do dispositivo tal como acontece em mostradores LCD.

A aplicação é capaz de simular a trajetória de elétrons dentro de um dispositivo hipotético, permitindo avaliar a qualidade do mesmo sem a necessidade de construí-lo. Para obter resultados de qualidade, é necessário executar a simulação com inúmeras diferentes configurações, o que acarreta em alto custo computacional.

A simulação de um conjunto de elétrons é considerada como o primeiro passo na paralelização da aplicação. Posteriormente, esta solução será aplicada no módulo do algoritmo genético que dispara inúmeras simulações a fim de testar diferentes parâmetros para a execução e determinar a melhor configuração para a simulação.

Este paper está organizado como segue: na seção 2 será feita a apresentação da aplicação. A seção 3 mostra a solução paralela. Os resultados são analisados na seção 4. A seção 5 é dedicada as conclusões e trabalhos futuros.

2 Apresentação da aplicação

Em poucas palavras, os chamados *displays* ou mostradores de informação CRT e FED possuem, em seu mecanismo de funcionamento, canhões de elétrons e uma superfície reagente a eles. Em um mostrador CRT os elétrons são emitidos por três diferentes canhões, um para cada componente de cor RGB (*Red*, *Green* e *Blue*). Basicamente, os elétrons disparados são guiados através de um tubo condutor até serem coletados em uma superfície de fósforo, fazendo com que sua energia seja convertida em luz, gerando a cor que será visualizada na tela. Por sua vez, a geração da imagem é dada pela varredura desses feixes de elétrons em toda a superfície da tela controlados por um campo magnético aplicado logo após a emissão dos elétrons pelo catodo. A figura 1 exibe o funcionamento de um dispositivo CRT.

Dispositivos FED, similarmente ao CRT, também possuem canhões de elétrons. No entanto, cada ponto da tela possui um conjunto dedicado de canhões, em escala micrométrica, tornando desnecessária uma componente que faça a

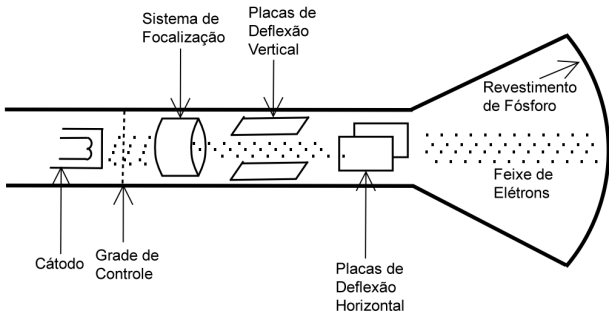

Figura 1: *Cathode Ray Tube* (CRT).

varredura da tela durante a geração da imagem. Isso permite que a distância entre estes canhões e a superfície coletora seja significativamente reduzia, fazendo com que o monitor seja mais fino. A idéia básica é mostrada na figura 2.

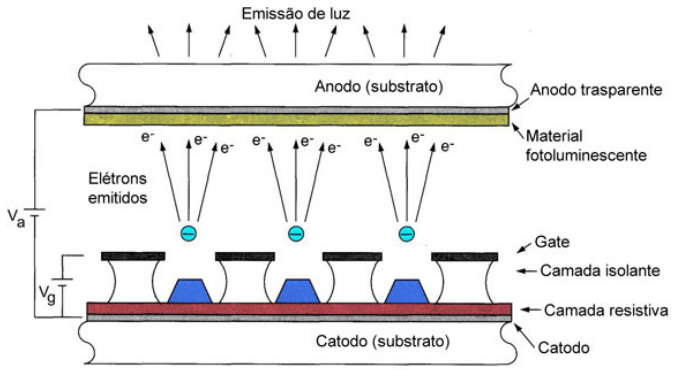

Figura 2: *Field Emission Display* (FED).

Cada elétron emitido pelos canhões percorre o espaço do dispositivo até atingir a superfície de fósforo (anodo). Durante sua trajetória, os elétrons são acelerados por uma força eletrostática e guiados por um campo elétrico devido à uma diferença de potencial aplicada entre o catodo e o anodo. No entanto, devido a diferentes propriedades inerentes ao dispositivo, tanto geométrica quanto elétrica, há dificuldade de se manipular precisamente os elétrons, alguns destes não colidem com o anodo chocando-se com as paredes do dispositivo. Os

elétrons que não atingiram a superfície coletora acarretam em uma perda de eficiência prejudicando assim a qualidade do dispositivo.

Existem diferentes formas de se construir o equipamento. Cada configuração é definida por um conjunto de medidas e atributos físicos que descrevem como o dispositivo será fabricado e afetam, em um nível mais baixo, a trajetória dos elétrons e, em um nível mais alto, seu funcionamento. Por ser uma tecnologia recente, prototipar experimentalmente diferentes configurações do dispositivo é relativamente custoso, o que justifica o uso desta aplicação afim de auxiliar na obtenção da configuração final do dispositivo.

Com a finalidade de obter uma configuração que apresente um bom compromisso entre eficiência e durabilidade, a aplicação conta com dois módulos principais. O primeiro módulo simula as trajetórias dos elétrons para uma configuração. Os resultados da simulação são passados para o segundo módulo. Este segundo módulo avalia os resultados da simulação e, implementando um algoritmo genético de evolução, progressivamente melhora a configuração atual. A seguir, serão apresentados os dois módulos principais.

2.1 Simulação

O simulador implementa o funcionamento de uma modelagem do problema. Existem outras implementações, que não possuem algumas características abordadas neste trabalho. Em [4] o simulador utiliza somente duas dimensões, restringindo as trajetórias dos elétrons a um plano. A modelagem utilizada neste trabalho considera uma terceira dimensão, simulando elétrons em um espaço.

Para executar uma simulação deve ser determinada uma série de parâmetros que especificam a configuração do dispositivo e dos elétrons. Com estes parâmetros o simulador é capaz de calcular as trajetórias para um dado número de elétrons. Ao final, a aplicação gera um conjunto de dados que permitem visualizar graficamente o comportamento dos elétrons dentro do dispositivo e analisar estatisticamente os resultados.

Para entender melhor o funcionamento do simulador, é necessário conhecer a maneira pela qual o dispositivo foi modelado. A modelagem é composta de quatro regiões principais:

- **Catodo**: região de onde os elétrons são lançados.
- **Dielétrico**: material isolante.
- **Grade** (*gate*): região que regula a emissão dos elétrons pelo catodo.
- **Anodo**: área composta por um material fosforescente onde os elétrons devem ser coletados.

A simulação ocorre da o seguinte forma: a grade gera um campo intenso na região do catodo, fazendo com que os elétrons sejam arrancados de sua superfície. Por sua vez, a diferença de potencial dada entre os componentes catodo e anodo fará com que o elétron se movimente através do dispositivo. Por

fim, a função do dielétrico é sustentar a estrutura além de isolar o catodo da grade e a grade do anodo.

A figura 3 demonstra em detalhes como o dispositivo foi implementado. O desenho mostra a geometria do dispositivo em três dimensões na qual é possível visualizar o catodo, a grade e o anodo. Esta é a representação matemática tridimensional do dispositivo real, onde cada eletrodo é modelado como um conjunto de anéis carregados. Neste desenho o dielétrico não foi representado para não dificultar a visualização dos outros componentes.

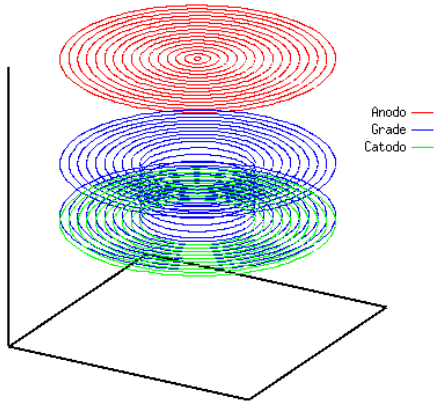

Figura 3: Configuração do dispositivo experimental

A trajetória de cada elétron é calculada com base em um método iterativo. Esta é composta de pontos calculados de forma independente um do outro, sendo o primeiro ponto determinado pelos atributos gerados por um sorteio aleatório (posição inicial do elétron no espaço). A carga de um elétron, utilizada no cálculo da trajetória, não interage com as cargas dos outros elétrons. As trajetórias podem ser calculadas separadamente pois o cálculo de uma não influi no das outras. O cálculo termina quando o elétron atinge a parede ou o anodo. De todas as fases da simulação esta é a que necessita de maior poder computacional.

A cada iteração do cálculo da trajetória de um elétron, o campo elétrico que irá guiá-lo deve ser recalculado para refletir o ponto no espaço da trajetória. O cálculo do campo exige alta precisão numérica, estudos comparativos da aplicação indicaram que a representação numérica nativa da arquitetura utilizada não é capaz de fornecer qualidade. Para lidar com este problema, é utilizada uma biblioteca de manipulação de números em ponto flutuante de tamanho limitado apenas pela quantia de memória disponível denominada GMP (*GNU Multiple-Precision Library*) [5]. O uso desta biblioteca aumenta a qualidade dos resultados mas acarreta em um custo computacional ainda maior.

Para cada elétron que colide com alguma parte do dispositivo, o resultado é computado nas estatísticas finais da simulação. As estatísticas informam o

número de elétrons que colidiram em cada uma das regiões citadas na figura 3. Com base nos dados coletados, é possível determinar a eficiência do dispositivo. Além de analisar o número de elétrons que atingiram o anodo (eficiência), através da observação dos outros dados estatísticos, pode-se ter uma idéia mais precisa do comportamento da simulação, possibilitando assim realizar um ajuste fino dos parâmetros.

2.2 Algoritmo Genético

Como dito anteriormente, o objetivo da simulação é auxiliar na construção e análise de um monitor com tecnologia FED. A configuração correta do dispositivo é necessária para que o máximo de eficiência seja obtida com o menor custo e consumo. Para minimizar o custo deve-se reduzir a voltagem aplicada ao dispositivo, diminuindo a eficiência. Por essas características serem diretamente proporcionais, mas pela necessidade de se reduzir uma e elevar outra, um compromisso deve ser estabelecido. Devido ao número indeterminado de configurações possíveis, não se deseja obter a configuração ótima, mas a que utilize o mínimo de voltagem sem denegrir a eficiência.

Devido ao tempo de execução de uma simulação, a imprevisibilidade do comportamento dos elétrons e o número de possíveis configurações é impraticável ajustar manualmente os parâmetros do dispositivo. Um analista deveria ser capaz de avaliar uma série de resultados e realizar ajustes finos, tentando convergir a eficiência para o máximo. No entanto, uma pequena modificação em um dos parâmetros pode resultar em grande modificação da resposta. Por outro lado, modificações bruscas de um parâmetro podem ser imperceptíveis nas estatísticas. Esse comportamento deve-se, em parte, à alta dependência entre os parâmetros.

Para resolver tal problema, o simulador conta com um módulo adicional, além do módulo de execução, que analisa e gera novos parâmetros realimentando o sistema de simulação automaticamente. Este módulo é chamado de AG (Algoritmo Genético) e é acoplado ao módulo de simulação. Através da idéia de evolução orgânica, em que organismos evoluem e somente os mais fortes sobrevivem, o algoritmo tenta desenvolver e aperfeiçoar uma configuração [6].

Para o funcionamento do AG são necessários dois conjuntos de dados: um conjunto de dispositivos e um conjunto de elétrons. O primeiro é determinado através da alteração aleatória de parâmetros do dispositivo, criando-se assim um conjunto de dispositivos que diferem-se em suas configurações. Este conjunto sofrerá modificações ao longo do tempo, como será explicado posteriormente. Por outro lado, o conjunto de elétrons (com características distintas) é estabelecido no início, mantendo-se o mesmo até o final do AG. O objetivo do AG é simular este conjunto de elétrons em cada um dos dispositivos, coletando as estatísticas da simulação (número de elétrons no anodo, dielétrico, grade e catodo). Após a simulação ter sido feita em todos os dispositivos o AG é capaz de atribuir um grau de evolução para cada um dos dispositivos (considerando as estatísticas coletadas). Um dispositivo que tem agora seu grau de evolução determinado é chamado de **cromossomo**. Ao conjunto de cromossomos é dado

o nome de **população**.

Assim como o número de elementos pertencentes ao conjunto de dispositivos (população), o tamanho do conjunto de elétrons a serem simulados é escolhido pelo usuário. É importante notar que quanto maior for a quantidade de elétrons simulados, melhor será a avaliação do dispositivo. Porém, o aumento da quantidade de elétrons acarreta em um aumento considerável no tempo de execução do AG.

O funcionamento do algoritmo genético consiste em, passando por sucessivas iterações, gerar um cromossomo evoluído. Um grupo de cromossomos é gerado à cada iteração, e deste, apenas o que possui maior evolução, permanece para a próxima iteração. Dessa forma garante-se, no mínimo, o mesmo grau de evolução a cada iteração. Ao definir o tamanho da população possibilita-se que maiores sejam as variedades de cromossomos. Deve-se ter precaução quanto a esse parâmetro do algoritmo genético, pois a análise e evolução ocorrerá somente quando todos os cromossomos tiverem sido processados individualmente. Quanto maior o tamanho da população, mais lenta poderá ser a convergência, mas quanto menor o tamanho da população, menor serão as chances de uma evolução em potencial. Exemplificando, se for escolhida pelo usuário uma população com 8 elementos, em cada iteração serão realizadas 8 simulações (uma para cada dispositivo da população). Cada simulação será responsável por calcular as trajetórias de 50 elétrons. Logo, no total, serão feitos 400 cálculos de trajetória em uma única iteração.

A geração de cromossomos novos a cada iteração baseia-se em dois fenômenos evolutivos: mutação e cruzamento. Mutação é a modificação de um ou mais atributos de um dispositivo, amplificando-o ou reduzindo seu efeito. Como exemplo o número de anéis dentro do gate do dispositivo, que pode aumentar ou diminuir. O cruzamento é a permutação de atributos semelhantes do dispositivo. A figura 4 demonstra graficamente as duas operações.

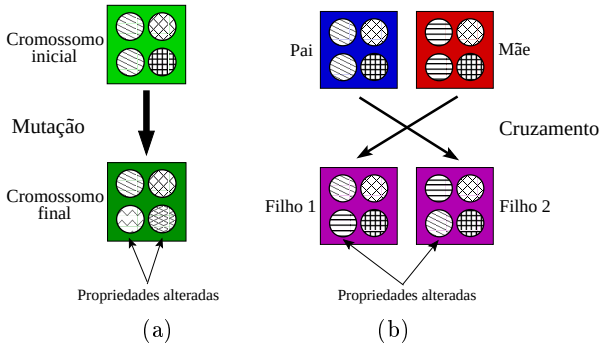

Figura 4: Mutação e cruzamento.

Tanto para uma mutação (a), quanto para o cruzamento (b), deve ser eleito um e dois pais, respectivamente. Durante essas atividades, os pais permanecem

inalterados, originando um filho modificado no caso da mutação ou dois filhos, com características de ambos os pais, no cruzamento. Deve-se considerar que somente são candidatos a pais os cromossomos com grau de evolução calculado. Estes compõem a população antiga, sendo a nova população composta por filhos e o pai mais evoluído da população antiga.

O sorteio dos pais é probabilístico, através da soma consecutiva dos graus de evolução obtém-se uma zona de eleição para cada pai. Se o valor sorteado cair dentro daquela zona, o pai é eleito para participar do processo de mutação ou cruzamento. Dessa forma todos têm chance de serem sorteados, porque mesmo o pior dos cromossomos pode estar próximo de um alto grau de evolução devido à imprevisibilidade de seus atributos.

O algoritmo genético termina após um determinado número de iterações. Como alternativa, pode-se estabelecer uma eficiência mínima, que deve ser atingida dentro das iterações estabelecidas. Para maior clareza sobre o funcionamento, ambos algoritmos, genético e simulação, são exibidos a seguir.

Algorithm 1 Algoritmo genético.

```
1:  for i = 0 até número de iterações do AG do
2:      população = Nova população
3:      for p = 0 até tamanho da população do
4:          dispositivo_atual = população[p]
5:          eficiência[p] = simula_eletrons(dispositivo_atual)
6:      end for
7:      Escolhe pais com base nos valores de eficiência
8:      Gera filhos
9:      for j = 0 até número de filhos do
10:         Realiza crossover nos filhos
11:         Realiza mutação nos filhos
12:     end for
13: end for
```

Na linha 5 do algoritmo genético é invocado o módulo de simulação (Algoritmo 1) e os resultados são armazenados. A configuração atual, criada pelo algoritmo genético, é passada como parâmetro do módulo. Em seguida, os pais são calculados com base na eficiência, dando maior preferência aos mais eficientes, através de um sorteio probabilístico. No processo de mutação e crossover, há uma chance probabilística de que uma dada característica mude, sendo possível que apenas um item do dispositivo sofra modificação.

Algorithm 2 Algoritmo de simulação.

```
1: Sorteia os parâmetros iniciais dos elétrons
2: for e = 0 até número de elétrons do
3:    while elétron e está dentro do dispositivo do
4:       Escreve posição atual em arquivo
5:       Calcula nova posição
6:    end while
7: end for
8: Escreve estatísticas
```

No módulo de simulação (Algoritmo 2), a linha 1 do algoritmo representa o sorteio de valores que dirão como os elétrons serão lançados dentro do dispositivo. O laço na linha 3 realiza o cálculo da trajetória, que tem seu término quando o elétron se choca contra alguma parede do dispositivo.

3 Solução Paralela

Esta seção tem por objetivo apresentar a solução paralela proposta para o problema. Primeiramente será realizada a paralelização do simulador. Posteriormente, esta solução será incluída no algoritmo genético.

3.1 Simulador Paralelo

A visão geral do simulador paralelo será apresentada de acordo com seus dois componentes principais. O primeiro é o processo mestre, que é responsável pelo agrupamento de resultados e distribuição de trabalho. O segundo componente são os escravos, responsáveis por calcular as trajetórias. Os dois componentes principais são vistos na figura 5.

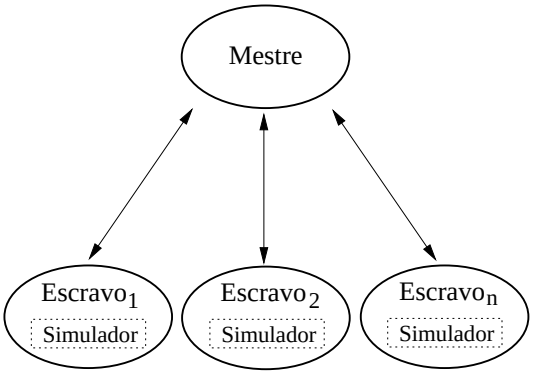

Figura 5: Componentes do simulador paralelo.

O funcionamento da solução paralela proposta segue o modelo mestre-escravo. O conjunto de elétrons é implementado na prática em um vetor, onde cada elemento deste vetor possuirá a posição, velocidade inicial e o ângulo de lançamento de cada elétron. O sorteio destes valores é realizado pelos escravos no inicio da simulação, mantendo-se a mesma semente do gerador de números aleatórios para que os escravos possuam o vetor idêntico aos demais. Portando, o trabalhos distribuídos pelo mestre são compostos somente pelo intervalo dos elétrons a serem simulados. O mestre então aguarda os resultados e envia novos trabalhos (se ainda restarem).

Por tratar-se de uma grande quantidade de informações, as trajetórias não são enviadas para o mestre através de mensagens. Durante uma execução da simulação, um arquivo temporário com as trajetórias é criado para cada escravo. Após calcular as trajetórias de um trabalho, ele as escreve neste arquivo temporário, enviando para o mestre somente as estatísticas (número de elétrons que atingiram anodo, catodo e as paredes do dispositivo). Caso o escravo volte a receber mais trabalho, este adicionará as novas trajetórias ao arquivo anteriormente criado.

Portanto, há a necessidade de utilizar-se NFS (*Network File System*) na solução paralela. Através do compartilhamento de disco, o mestre, ao final da execução, agrupa as trajetórias de calculadas por cada escravo em um único arquivo. Além disso, computa as estatísticas finais, somando as estatísticas parciais de cada escravo.

3.2 Algoritmo Genético Paralelo

Como dito anteriormente, o objetivo do estudo da paralelização do simulador é prover uma base eficiente para a paralelização do algoritmo genético. A maior parte do tempo de processamento deste encontra-se na simulação dos elétrons para cada dispositivo. Isso implica que a simulação afetará significativamente o tempo de execução de cada iteração do algoritmo genético. Este fator é ainda mais influente visto que uma iteração do algoritmo genético realizará a simulação para um mesmo conjunto de elétrons tantas vezes quanto for o tamanho da população. Outra característica é o tamanho do conjunto de elétrons, que deve ser suficiente para prover resultados coerentes, ou seja, que permita classificar os dispositivos e seja realista.

É importante considerar também que o processamento do dispositivo e todas as trajetórias tem como fim as estatísticas associadas aos dispositivos. Como o AG necessita somente das estatísticas para estabelecer a qualidade de cada dispositivo, os resultados intermediários deste processamento, tal como os caminhos das trajetórias, são descartados pois não há necessidade de reutilizá-los.

Visto que o algoritmo genético é composto por um conjunto de iterações e possui um critério de parada definido pelo usuário, é necessário estabelecer um limite para a execução dos testes. Para este trabalho, foi determinado que a primeira iteração é um bom indicador da eficiência do algoritmo paralelo. Neste caso, o tempo de computação é maior que nas iterações posteriores. O motivo principal desta escolha é avaliar o algoritmo em seu pior caso, no qual todos

os elementos da população não possuem seu grau de evolução computado. Nas iterações posteriores, como visto na seção 2.2, será mantido o melhor cromossomo obtido na iteração anterior, não necessitando computar suas trajetórias novamente, reduzindo em um o número de dispositivos a serem simulados.

O mestre envia para os escravos o dispositivo, juntamente com o subconjunto de elétrons que deverão ser simulados, como demonstrado na figura 6. A seta preta contínua indica que o conjunto total de elétrons será dividido entre os escravos. Por outro lado, a seta preta tracejada mostra que o dispositivo é simplesmente copiado para todos os escravos.

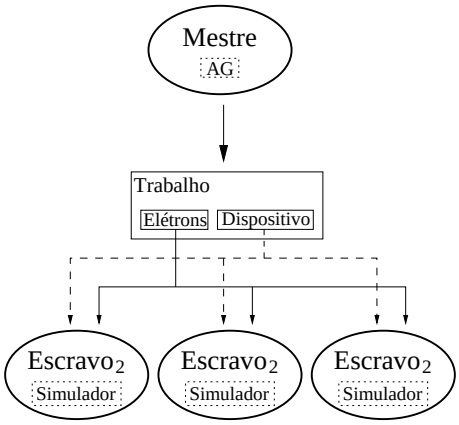

Figura 6: Distribuição do trabalho.

Assim como no simulador paralelo, o escravo enviará as estatísticas ao mestre após calcular as trajetórias de um subconjunto de elétrons. Existindo ainda elétrons a serem simulados para o determinado dispositivo, o mestre enviará um novo trabalho para este escravo. Caso não haja mais elétrons, ou seja, todas as trajetórias do dispositivo já calculadas, o mestre iniciará a distribuição dos trabalhos para o proximo dispositivo da população, até que todos os dispositivos sejam processados. Ao fim de uma iteração, o mestre agrupa os resultados enviados pelos escravos e executa as operações de mutação e cruzamento.

4 Resultados

Como dito no inicio do trabalho, o foco principal é a paralelização da ferramenta em uma arquitetura de baixo custo (*cluster*). Neste trabalho utilizou-se **I-Cluster2** [7] localizado no *Laboratoire ID* em Grenoble, na França. Este possui 100 nós, cada um com dois processadores de 64 bits Itanium-2 de 900 MHz e 3GB de memória principal. A interligação dos nós é feita através de uma rede proprietária chamada *Myrinet* [8].

Primeiramente foram realizados experimentos com o simulador paralelo para

verificar o ganho obtido com a simulação das trajetórias em paralelo. Para tanto, variou-se o número de escravos de 2 até 10. Os resultados podem ser observados na figura 7. O número de elétrons a serem simulados neste experimento é 100.

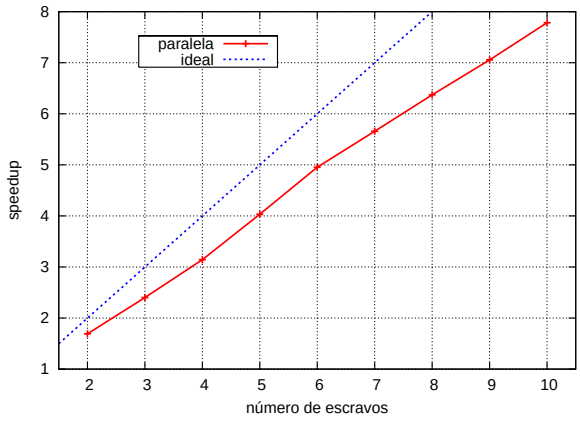

Figura 7: Speedup do simulador paralelo.

Os resultados demonstram que a solução paralela é bem escalável até 10 escravos. De acordo com o comportamento da curva pode-se prever que a utilização de mais escravos reduzirá ainda mais o tempo de execução da aplicação, pois com 10 escravos, a curva ainda está em crescimento, não atingindo o ponto de inflexão. Isto indica que os escravos estão trabalhando na maior parte do tempo, havendo um pequeno tempo desperdiçado com trocas de mensagens entre o mestre e os escravos ou em casos em que os escravos permanecem inativos aguardando um novo trabalho a ser enviado pelo mestre. Este pequeno tempo desperdiçado com troca de mensagens ocorre devido ao uso de uma rede de baixa latência, a *Myrinet*.

Tendo verificado o ganho obtido com a versão paralela do simulador, foram realizados testes com a versão paralela do AG, o qual inclui o simulador paralelo. Como o tempo de execução do simulador paralelo foi muito inferior ao da versão seqüencial, esperava-se que o tempo de execução de cada iteração do AG também o fosse. Como o AG necessita de mais poder computacional que o simulador, devido ao fato de executar várias simulações, uma para cada elemento da população, utilizou-se até 20 escravos. Esta questão é confirmada com os resultados exibidos na figura 8, onde o número de elétrons a serem simulados por dispositivo é 50.

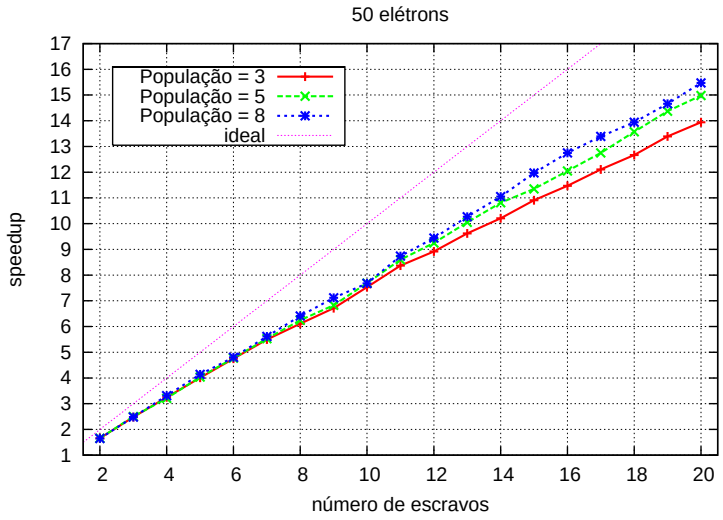

Figura 8: *Speedup* com 50 elétrons por dispositivo da versão paralela do AG.

Analisando os resultados percebe-se que, como na versão paralela do simulador, a solução se mostrou bem escalável. Além disso, o aumento da população gerou *speedups* ainda melhores. Isso é facilmente explicado: o aumento da população gera uma necessidade ainda maior de processamento. Neste contexto, os escravos irão ter mais trabalhos para processar, tornando ínfimo o tempo desperdiçado com a ociosidade dos escravos.

A questão que surge é: se dobrarmos o número de elétrons a serem simulados por dispositivo, o ganho será maior? Como visto nos resultados anteriores, o fato de ter aumentado o tamanho da população fez com que o *speedup* também aumentasse. Por tanto, para responder esta questão foram realizados testes com um conjunto de 100 elétrons. Os resultados são vistos na figura 9.

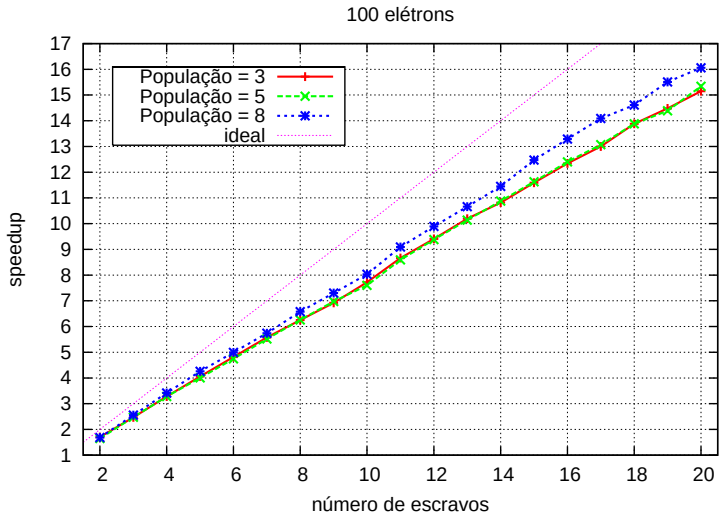

Figura 9: *Speedup* com 100 elétrons por dispositivo da versão paralela do AG.

Como era esperado, os *speedups* obtidos com o processamento de 100 elétrons por dispositivo foram maiores. Novamente, uma quantidade maior de trabalho valorizou ainda mais a solução paralela, obtendo-se resultados ainda melhores. Isto é verificado pelo fato das curvas de resultado estarem ainda mais próximas da ideal.

5 Conclusão

A solução paralela do algoritmo genético, a qual era o objetivo final do estudo, mostrou-se bem escalável. A inclusão de mais indivíduos na população e o aumento do conjunto de elétrons a serem simulados fez com que a aplicação necessitasse de um grande poder de processamento. Logo, a solução paralela comportou-se ainda melhor nestes casos.

Para comprovar a escalabilidade da solução foi realizado um teste pontual. Neste experimento utilizou-se 40 escravos e o maior caso de teste (8 indivíduos na população e 100 elétrons a serem simulados por dispositivo). O *speedup* obtido neste teste foi de 30.1, ficando bem próximo do ideal (40) visto que 40 nós escravos foram utilizados. Em termos de tempo de execução, uma iteração do AG para este caso de teste foi reduzida de **12 horas** (versão seqüencial) para **25 minutos** (versão paralela). Com 40 escravos, a curva de *speedup* ainda não atingiu seu ponto de inflexão, ou seja, a inclusão de mais nós escravos ainda reduziriam o tempo de execução da aplicação.

Referências

[1] PHILLIPS, J. C. et al. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, v. 26, p. 1781–1802, 2005.

[2] CASTRO, M. B. et al. A parallel version for the propagation algorithm. In: MALYSHKIN, V. (Ed.). *Parallel Computing Technologies: 8th International Conference, PaCT 2005*. Krasnoyarsk, Russia: Springer-Verlang Heidelberg, 2005. (Lecture Notes in Computer Science, v. 3606), p. 403–410.

[3] MANSSOUR, I. H. et al. High performance approach for inner structures visualisation in medical data. *International Journal Of Computer Applications In Technology*, v. 22, n. 1, p. 23–33, 2005.

[4] FONSECA, L. R. C.; ALLMEN, P. von; RAMPRASAD, R. Numerical simulation of the tunneling current and ballistic electron effects in field emission devices. *Journal of Applied Physics*, v. 87, n. 5, p. 2533–2541, 2000.

[5] THE GNU MULTIPLE PRECISION BIGNUM LIBRARY. Disponível em: <http://www.swox.com/gmp>. Acesso em: 2 jun. 2006.

[6] MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Londres, Reino Unido: Springer-Verlag, 1996.

[7] Laboratory ID. Disponível em: <http://ita.imag.fr>. Acesso em: 20 out. 2006.

[8] Myrinet. Disponível em: <http://www.myri.com/myrinet>. Acesso em: 20 out. 2006.