SPMD – Simulador de Propagação de Mensagens Digitais

Esteban W. G. Clua Rainier A. F. Sales

Universidade Federal Fluminense, Instituto de ciência da computação, Brasil.

Resumo

O uso de simuladores vem cada dia sendo mais adotado na tentativa de prever ou mesmo entender os efeitos sociais envolvidos nas relações entre pessoas no mundo globalizado. Em outra vertente temos o uso de processamento paralelo via placas gráficas, sendo uma interessante possibilidade de uso em sistemas de larga escala de processamento de dados. Por tanto, este artigo tem como proposta o desenvolvimento e análise do simulador denominado SPMD – Simulador de Propagação de Mensagens Digitais, buscando assim levantar estudo a respeito das variáveis globais que envolvem o envio de mensagens de usuários conectados ou não em uma rede virtual e validar sua utilização com o uso comparativo de processamento em GPU (Unidade de Processamento Gráfico).

Keywords: Palavra chaves: Grafos, redes, variáveis aleatórias, simulação, algoritmos, GPU, CUDA.

Authors' contact:
esteban@ic.uff.br
rainier_augusto@yahoo.com.br

1. Introdução

O surgimento da internet aconteceu sem grandes pretensões e planejamentos, sendo principalmente voltada para o ensino e a pesquisa. No entanto, na atualidade, temos uma rede altamente conectada em grande parte do mundo, sendo usada para fins bem diferentes dos inicialmente pensados por uma gama de usuários cada vez mais heterogêneos.

O processo de democratização da internet veio acompanhado de alguns aspectos negativos, dentre eles o uso de mecanismos para propagação e difusão de uma mensagem na rede. Um dos mecanismos mais utilizados na propagação de mensagens na internet é o uso de "spam", conforme caracteriza Teixeira (2001), No ambiente da Internet, spam é considerado um abuso e se refere ao envio de um grande volume de mensagens não solicitadas, ou seja, o envio de mensagens indiscriminadamente a vários usuários. Com a evolução natural da internet, os spams evoluíram para diversos novos mecanismos criados na rede, o que antes acontecia exclusivamente por e-mail agora busca outras ferramentas como redes sociais, blogs, chats, etc. Devido a gama de mecanismos devidamente conectados (notebook, pager, tablet, smartphone etc.), a internet tornou-se mais disponível que no passado, sendo a propagação de spams muito mais rápida e eficiente no que tange o alcance aos usuários.

Para entendermos como é o comportamento de uma mensagem se propagando na rede podemos adotar duas perspectivas, a de análise e rateio de uma gama de mensagens se propagando na rede ou fazendo a construção e uso de um simulador adotando as características desse cenário. No entanto, a análise da propagação de uma ou mais mensagens na internet é praticamente inviável devido ao pouco conhecimento e controle da conectividade e demais aspectos intangíveis provenientes da mensagem, o que torna a primeira opção inviável.

A proposta desse artigo é, portanto, a construção, validação e análise do SPMD e seus resultados, para se ter assim uma maior compreensão de como acontece a propagação de mensagens em um ambiente no qual seja possível controlar o nível de conectividade dos usuários e a propensão de que a mesma seja compartilhada.

2. SPMD

O SPMD tem como objetivo principal simular a propagação de mensagens genéricas em diversos grupos de redes da internet, como redes sociais, blogs etc.

Basicamente, na construção do SPMD, foram identificadas duas variáveis que influenciam diretamente na quantidade de usuário infectados (infectados será o termo utilizado no artigo para representar um usuário que recebeu a mensagem), sendo elas P e Q: A variável P representa a taxa de conectividade entre os usuários, ou seja, P é um valor percentual que determina o quão conectado os usuários de uma rede estão. Suponha um grafo G com o valor de P = 0%, ou seja, nenhum dos usuários representados está conectado com os demais. Suponha agora o grafo G com o valor de P = 100%, ou seja, todos os usuários representados estão conectados entre si. A representação de dos extremos é bem clara, no entanto, como realizar a representação para valores entre 0% e 100% conectados já que temos um número relevante de possibilidades de representação? A estratégia adotada no SPMD se baseia na representação desta relação a partir de variáveis aleatórias.

Uma variável aleatória para Kallenberg [1986, p. 79] "pode ser considerada como o resultado numérico de operar um mecanismo não determinístico, ou de fazer uma experiência não determinística, para gerar

resultados aleatórios", ou seja, uma variável aleatória pode ser definida como uma variável quantitativa cujo resultado (valor) depende de valores aleatórios.

O uso de variáveis aleatórias na variável P é realizado pela determinação de conectividade usuário a usuário, baseado em um número randômico gerado instantaneamente para determinar a relação ou não de conectividade entre dois indivíduos, conforme exemplifica o pseudo-algoritmo:

```
Seja:
1. V {A, B} (vértices avaliados)
2. P ← 50% de taxa de conectividade
3. R ← Random(0, 100)
4. Se: R ≤ P
5.    Cria conexão entre A,B
6.    Cria conexão entre B,A
7. Se não:
8.    Não cria conexão entre A,B
9.    Não cria conexão entre B,A
```

Assim temos no primeiro ponto a determinação das variáveis na qual será analisada a conexão (ou seja, A e B), sendo P a taxa de conectividade definida pelo usuário, variando de 0% a 100% (nesse exemplo definido como 50%) e R um número randômico entre 0 e 100 (com estes inclusos). Temos que, se o número gerado for menor ou igual a taxa de conectividade, teremos ligação entre A e B e consequentemente entre B e A (o SPMD considera que se B conhece A, então A conhece B). No entanto, se o número gerado for maior que a taxa de conectividade, então não temos conectividade entre A e B (e consequentemente entre B e A). Como temos uma variável aleatória de mesma probabilidade para todos os valores entre 0 e 100 (com 0 e 100 contido) temos que a conectividade do grafo será portanto a taxa de P.

Para auxiliar o processo de construção da conectividade da rede de modo menos exato e mais parecido com o real foi utilizada após a construção exata da matriz de conectividade a técnica de recombinação baseada em algoritmo genético (crossing over). Essa técnica foi adota buscando encontrar soluções mais aproximadas na otimização da taxa de conectividade como uma abstração da real conectividade existente na relação entre os usuários envolvidos. Assim, vamos supor uma matriz de adjacência G qualquer que representa a conectividade de cinco usuários. Temos que, quando existe conectividade entre dois vértices a interseção da linha pela coluna recebe o valor de 1, enquanto quando não houver recebe o valor de 0. 1 Neste caso foi adotada uma conectividade de 12%. A técnica de recombinação gerada a partir da taxa de crossing over definida pelo usuário produz uma nova matriz que será a real utilizada para transmitir a mensagem entre os usuários.

Neste caso, como temos a possibilidade de 25 modificações na matriz, menos 5 modificações bloqueadas (ligações do vértice com ele mesmo), aplicando a taxa de 10% tem-se:

```
1. Crossing Over (n)=Taxa (Y)*⟦(n⟧^2-n)
2. Crossing Over (5)=Taxa (10%)*⟦(5⟧^2-5)
3. Crossing Over (5)=2
```

Logo, realiza-se duas modificações aleatórias na matriz original, conforme demonstra o algoritmo abaixo:

```
1. V ← Random(1 até n)
2. U ← Random(1 até n)
3. Se (V ≠ U)
4.  Se (Vetor de verificação [v,u] == 0)
5.    Vetor de verificação [v,u] == 1
6.    Vetor de verificação [u,v] == 1
7.  Se não:
8.    Vetor de verificação [v,u] == 0
9.    Vetor de verificação [u,v] == 0
10. Se não:
11.  Chama recursivamente.
```

Está metodologia foi adotada buscando modificar a matriz inicial gerada para uma mais parecida com um ambiente real de conectividade entre usuários. Esta se torna interessante quando se pensa em conectividades de usuários relativamente próximas, mas não exatas a valores matemáticos reais, portanto a matriz gerada inicialmente ira deter uma taxa de conectividade aproximada de 12%, sendo a taxa do crossing over a responsável pelo grau de variação para mais ou para menos.

A variável Q, por sua vez, representa a propensão em transmitir uma mensagem, ou seja, é a taxa que determina quantos e quais filhos um usuário W ira infectar baseado na conectividade que o mesmo possui. Seja o grafo G o grafo de conectividade do vértice W. Suponha que W foi contaminado. Logo, se W tem n filhos, em uma primeira situação pode-se contaminar n + 1 usuários (n que são os filhos de W + o próprio W que já foi contaminado). Porem cada filho de W também pode ter filhos. Como consequência, cada filho descendente de um filho pode ter filhos, maximizando o número máximo de infectados, sendo o limite de pessoas infectadas o valor total de pessoas pertencentes ao grupo analisado. Assim, suponhamos um grafo G qualquer de conectividade igual a 0%, suponha que seja infectado um usuário qualquer (sem perda de generalidade), por consequência teremos sempre 1 infectado somente, independente do valor de Q. Já que a conectividade P é igual a 0%. No entanto suponha agora que haja conectividade completa no mesmo grafo G, neste caso o número de infectados varia de acordo com o valor de Q, ou seja, a quantidade de infectados do grafo G será determinada pelo valor da variável Q.

A representação dos extremos é bem clara; no entanto, como realizar a representação para valores entre 0% e 100% conectados já que temos um número relevante de possibilidades possíveis de representação? A estratégia adotada no SPMD se baseia na representação desta relação a partir de variáveis

aleatórias do mesmo modo aplicado a variável P. Basicamente a diferença do tratamento acontece nas saídas de resultado, conforme exemplifica o pseudo-algoritmo:

```
Seja:
1. V {A, B} (vértices avaliados)
2. Q ← 50% de taxa de contaminação
3. R ← Random(0, 100)
4. Se: R ≤ P
5.    A infecta B
6. Se não:
7.    A não infecta B
```

Assim temos no primeiro ponto a determinação das variáveis na qual será analisado a relação e o sentido da contaminação, ou seja de A para B, sendo Q a taxa de contaminação definida pelo usuário, variando de 0% a 100% (nesse exemplo definido como 50%) e R um número randômico entre 0 e 100 (com 0 e 100 contido). Temos que, se o número gerado for menor ou igual á taxa de infecção, teremos B infectado por A. No entanto, se o número gerado for maior que a taxa de infecção então não temos contaminação de A para B. Como temos uma variável aleatória de mesma probabilidade para todos os valores temos que a taxa de infecção do grafo será, portanto a taxa de Q.
Seja G o grafo anteriormente visto 100% conectado (P = 100%), e seja a taxa de infecção Q = 0%, suponha que seja infectado um vértice qualquer (sem perda de generalidade), teremos sempre o número de infectados = 1, pois mesmo estando totalmente conectado, o primeiro infectado não tem intenção nenhuma (Q = 0%) de transmitir a mensagem. Se aumentarmos o número de Q iremos aumentar o número de usuários infectados, já que ao aumentarmos a propensão do usuário W infectar seus filhos, e os filhos de W também infectarem seus filhos na mesma proporção e assim por diante.

Para efeito de linearidade, foi considerado que uma vez infectado, o usuário não pode ser infectado novamente, já que por estar infectado não pode tentar novamente infectar seus filhos, para que assim não ocorram loops. Assim, seja a o vértice raiz, quando vértice infecta seus filhos, infecta somente b, este por sua vez infecta c, no entanto c apesar de estar ligado em a não pode tentar infecta-lo, já que a já está infectado. Assim c infecta d.

3. Processamento via GPU

Durante a análise de desenvolvimento do SPMD, avaliou-se a possibilidade de implementação do mesmo utilizando de processamento via GPU.

A indústria de simuladores e a demanda de aplicativos em tempo real, com gráficos 3D de alta definição, fez com que as placas gráficas programáveis (GPUs) evoluíssem para sistemas massivamente paralelos, com alto poder de processamento e grande largura de banda de memória. As GPUs, inicialmente designadas ao processamento gráfico, hoje, contêm o mais poderoso chip disponível em uma estação de trabalho de alto desempenho (Luebke 2008). No entanto, permanece sendo um desafio desenvolver aplicativos que possam distribuir adequadamente as tarefas entre os vários núcleos de processamento para, assim, conseguir um ganho real de desempenho (NVIDIA 2010).

Ciente desse fato, a fabricante de placas gráficas NVIDIA, desenvolveu uma extensão da linguagem C para a arquitetura CUDA que permite utilizar os processadores da placa de vídeo para outros fins, como processamento numérico e computação científica. Além da linguagem C, é possível programar as GPUs que suportam a arquitetura CUDA, utilizando algum dos muitos wrappers disponíveis, por exemplo, para as linguagens Java (jCUDA), C# (CUDA.NET), Python (pyCUDA), entre outros.

O modelo de programação em CUDA define funções em C, chamadas kernels que, quando chamadas, serão executadas N vezes em paralelo por N diferentes threads na GPU. Blocos de threads podem ter uma, duas, ou três dimensões, fornecendo um meio natural para ser em utilizados no cálculo de campos escalares uni, bi ou tridimensionais (NVIDIA 2010). Por serem desenhadas para aumentar o desempenho de aplicações gráficas, as GPUs são capazes de realizar grande quantidade de operações aritméticas simultaneamente, dedicando mais transistores ao processamento de dados, em vez de cache e controle de fluxo, como ocorre nas CPUs. São, portanto, poderosas ferramentas para processar grandes quantidades de dados paralelamente o que é ideal para muitas aplicações como processamento de imagens e dados de tomografia computadorizada. Processar dados na GPU é, primeiramente, uma maneira de minimizar o uso da CPU e, além disso, o aumento de desempenho pode ser significativo por um preço bem menor se comparado a outros sistemas. Diversos pesquisadores já vêm utilizando os benefícios da programação em paralelo oferecido pela GPU, conseguindo ganhos de 130x (Lu, et al. 2009) até 300x (Fang e Boas 2009).

Para análise da viabilidade do uso da GPU no SPMD, foi desenvolvido um projeto em paralelo acompanhando os ganhos de desempenho CPU vs GPU. Desconsiderado o processamento da parte gráfica, que não é o foco do simulador, temos basicamente 2 processo macroscópicos acontecendo:

- Construção da Matriz: Construção da matriz de conectividade na qual armazena os inter-relacionamentos dos vetores (linha x coluna).
- Espalhar infecção: Chamada recursiva de pai para filho na qual pode contaminar ou não seus descendentes e assim por diante.

Todos os teste de CPU vs GPU foram trabalhados utilizando médias, sendo o total de simulações realizadas por avaliação de 100 unidades. Os valores

de p e q em todos os testes foi mantido como máximo para assim garantir a mesma avaliação para todos os valores de unidades de usuários.

O primeiro teste realizado iniciou com o número de usuários igual a 1.028. Os tempos registrados nos dois processos descritos (em média) foram:

	CPU	GPU
Construção da matriz	47ms	1232ms
Espalhar infecção	0ms	94ms

Tabela 3 – CPU vs GPU parte I

No segundo teste realizado o número de usuários foi ampliado para 10.000. Os tempos registrados nos dois processos descritos (em média) foram:

	CPU	GPU
Construção da matriz	6146ms	1491ms
Espalhar infecção	94ms	125ms

Tabela 4 – CPU vs GPU parte II

No terceiro teste realizado o número de usuários foi ampliado para 30.000. Os tempos registrados nos dois processos descritos (em média) foram:

	CPU	GPU
Construção da matriz	49733ms	1294ms
Espalhar infecção	749ms	873ms

Tabela 5 – CPU vs GPU parte V

Analisando os resultados obtidos, temos como gargalo no SPMD a geração da matriz. No caso de espalhar a infecção, como o código é exatamente o mesmo, detecta-se uma maior demora no tempo devido a necessidade de copiar a memoria da GPU para a CPU, sendo assim avaliado como desnecessário o uso da mesma neste processo.

Assim tem-se um limiar desconhecido onde o uso da GPU na construção da matriz supera o uso pela CPU. Buscando encontrar o limiar onde o modo utilizando a GPU se torna mais eficiente que a CPU, alteram-se os valores para faixas menores de usuários, sendo o limiar encontrado de aproximadamente 3.500 usuários. Assim, tem-se que para valores grandes de usuários o uso do processamento via GPU torna-se necessário para atender os requisitos do SPMD, enquanto para pequenos valores o uso via CPU é mais interessante, no entanto sendo praticamente imperceptível ao usuário neste caso.

3. Conclusão

Este artigo apresentou um estudo a respeito do uso de simuladores para entendimento e previsão de contaminação de usuários em uma rede, baseando-se nas características de conectividade da rede e na propensão do usuário retransmitir ou não a mensagem recebida.

A partir deste estudo, pode-se validar a relação eminente entre a conectividade e retransmissão da mensagem, sendo ambos diretamente responsáveis pela maximização da quantidade de usuários atingidos. Portanto, para aumentar valores de usuários infectados podemos maximizar a conectividade da rede para valores de retransmissão fixos, ou maximizar valores de retransmissão para valores de conectividade fixo.
Partindo do pressuposto de que a conectividade P e a retransmissão Q não são fixos, pode-se compreender que a maximização de valores altos de P e Q tornam-se desnecessários já que para valores aproximados de 20% em ambos teremos basicamente o mesmo alcance, que é o alcance máximo.

Por último foi possível analisar e viabilizar o uso do processamento via GPU no SPMD, sendo interessante principalmente quando pensamos em um grande grupos de dados a ser analisado em um espaço menor de tempo.

Referências

FANG, QIANQIAN, E DAVID A. BOAS. "MONTE CARLO SIMULATION OF PHOTON MIGRATION IN 3D TURBID MEDIA ACCELERATED BY GRAPHICS PROCESSING UNITS." OPTICS EXPRESS 17, N. 22 (2009): 20187-20190.WOODCOCK, S., 2001. Game AI: the state of the art industry 2000-2001. *Game Developer*, 8 (8), 36-44.

KALLENBERG, O., RANDOM MEASURES, 4ª EDIÇÃO. ACADEMIC PRESS, NEW YORK, LONDON; AKADEMIE-VERLAG, BERLIN (1986). MR0854102 ISBN 0123949602

KOZA, J.R.. GENETIC PROGRAMMING: ON THE PROGRAMMING OF COMPUTERS BY MEANS OF NATURAL SELECTION. MIT PRESS, 1992.

LU, PETER J., HIDENKAZU OKI, CATHERINE A. FREY, GREGORY E. CHAMITOFF, E ET AL. "ORDERS-OF-MAGNITUDE PERFORMANCE INCREASES IN GPU-ACCELERATED CORRELATION OF IMAGES FROM THE INTERNATIONAL SPACE STATION." JOURNAL OF REAL-TIME IMAGING PROCESSING, 2009.

NVIDIA. "CUDA C PROGRAMMING GUIDE VERSION 3.1.1." NVIDIA CORPORATION. 2010. WWW.NVIDIA.COM.

TEIXEIRA, RENATA CICILINI. BOLETIM BIMESTRAL SOBRE TECNOLOGIA DE REDES. PRODUZIDO E PUBLICADO PELA RNP: HTTP://WWW.RNP.BR/NEWSGEN/0101/SPAM.HTML 19 DE JANEIRO DE 2001 | VOLUME 5, NÚMERO 1. ISSN 1518-5974.