Aplicação de Modelo de Programação Híbrido na Espacialização do Relevo no Submédio do São Francisco

Jesse Nery[1], Diego Lapa[1], Ícaro Gonzalez[1], Murilo Boratto[1], Brauliro Leal[1]

[1] Colegiado de Engenharia da Computação (CECOMP)
Universidade Federal do Vale do São Francisco (UNIVASF)
Av. Antonio Carlos Magalhães, 510, Santo Antonio, 48902-300
Juazeiro – Bahia – Brazil

{jesse, diego, icaro, murilo.boratto, brauliro.leal}@univasf.edu.br

Resumo. *A Espacialização do Relevo é um instrumento utilizado na simplificação da representação do relevo e sua importância se deve ao fato de possibilitar a descrição de fenômenos por meio de modelos matemáticos a partir de uma amostra de dados. Graficamente, equivale a identificar a superfície matemática que melhor se ajusta aos pontos de um diagrama de dispersão. A Computação de Alto Desempenho vem ampliar o seu poder computacional possibilitando o desenvolvimento da representação do relevo na região do Submédio do Vale do São Francisco. Nesse trabalho apresentamos uma metodologia para representação do relevo utilizando o método de regressão polinomial bidimensional otimizado através de um Modelo Paralelo de Computação de Alto Desempenho.*

1. Introdução

Alguns fatos recentes têm proporcionado o desenvolvimento de aplicações que representem recursos geofísicos através de uma visão computacional de forma eficiente, e dentre essas representações destaca-se a Espacialização de Relevo através de polinômios bidimensionais [Nogueira et al. 2008]. O problema de representação do relevo através de um polinômio já havia sido estudado anteriormente [Bajaj et al. 1993], porém sem a abordagem na distribuição de processamento. Este fato limitou que um polinômio de alto grau fosse estimado e o tamanho da área a ser representada fosse limitada, pois quanto maior for a informação a ser representada, maior será o poder computacional exigido, sendo que além disso, uma representação com relativa fidelidade requer um polinômio de alto grau, demandando um grande poder computacional.

Dentre as inúmeras justificativas para a realização da Espacialização do Relevo, focou-se na busca de respostas para mensurações em áreas agrícolas, tendo no dimensionamento de plantações, na otimização de recursos hídricos, na logística de estocagem da produção e minimização dos efeitos erosivos, fatores preponderantes. Logo, o processo de Espacialização do Relevo torna-se uma ferramenta fundamental e de grande relevância na exploração eficiente da agricultura, principalmente, no

polo agrícola situado as margens da região do Submédio do Vale do Rio São Francisco, que se destaca como uma das maiores zonas de vinicultura e fruticultura para exportação do país. E nesta zona um dos principais problemas que dificultam uma maior eficiência dos fatores produtivos agrícolas deve-se aos problemas relacionados à erosão dos solos, associado ao uso da terra de maneira inadequada.

O objetivo deste trabalho é apresentar uma metodologia para representação do relevo da região agrícola do Submédio do Vale do São Francisco utilizando o método de regressão polinomial bidimensional otimizado através do Modelo de Programação Híbrido para Computação de Alto Desempenho. O trabalho está estruturado da seguinte forma: Na Seção 2 será abordado o modelo matemático da Espacialização do Relevo. Na Seções 3, 4 e 5 será apresentado o Modelo de Programação Híbrido para Computação de Alto Desempenho que foi utilizado, no final as Conclusões e Trabalhos Futuros.

2. Modelo Matemático do Relevo

Um Modelo Matemático do Relevo é uma representação matemática computacional da distribuição de um fenômeno espacial que ocorre dentro de uma região da superfície terrestre [Namikawa et al. 2003]. Este modelo pode representar diversas informações geográficas de um terreno como: geológicas, geofísicos, umidade do ar, altitude do terreno, ... Uma da técnicas para realizar essa representação do relevo é através do Modelo de Grade Regular [Rufino et al. 2009], onde o mapeamento da superfície é feita com um ajuste global a uma superfície polinomial através da técnica de regressão polinomial. Esta técnica que ajusta um polinômio bidimensional que melhor descreve a variação dos dados de uma amostra torna-se limitada devido ao alto poder computacional demandado exigido para realizar a regressão em um conjunto de dados muito grande. Quando se utiliza modelos matemáticos de regressão, o método de estimação dos parâmetros mais amplamente utilizado é o método dos mínimos quadrados ordinários [Golub and Loan 1989] que consiste em estimar uma função para representar um conjunto de pontos minimizando o quadrado dos desvios. Considerando um conjunto de coordenadas geográficas (x, y, z), tomando a altitude estimada como função estimadora destes pontos, um polinômio de grau r em x e de grau s em y pode ser dado conforme a Equação 1, com o erro ε_{ij} estimado pela Equação 2.

$$\widehat{z} = f(x_i, y_j) = \sum_{k=0}^{r} \sum_{l=0}^{s} a_{kl} x_i^k y_j^l \tag{1}$$

$$\varepsilon_{ij} = z_{ij} - \widehat{z_{ij}} \tag{2}$$

3. Modelo de Programação utilizando Memória Compartilhada (OpenMP)

O modelo de Programação com Paradigma de Memória Compartilhada [Kumar 2002] caracteriza-se pela existência de uma porção de memória que possa ser acessada diretamente por todos os elementos de um conjunto de processos. Esta memória será utilizada para transferência de informação entre os

mesmos. Este tipo de modelo corresponde a sistemas que possuem um conjunto de memórias compartilhadas com todos os processadores envolvidos, onde a memória estaria distribuída no sistema, entre os distintos processadores. Apesar de ser fácil de usar e a transferência de dados ser rápida, existe uma limitação com relação ao número de processadores, não sendo escalável para as diversas máquinas do cluster. Existem ferramentas específicas de programação em Memória Compartilhada, as mais conhecidas são: OpenMP [Dagum and Menon 1998] e Threading Building Blocks [TBB 2011]. OpenMP(Open Multi-Processing) é um padrão atual para a programação utilizando o modelo de memória compartilhada, que incluem os sistemas *multithreads* e computadores de alto desempenho com memória virtual compartilhada. O OpenMP define uma API para chamadas de funções das bibliotecas que permitem lograr uma grande variedade de funcionalidades. Assim, encontramos funções para averiguar o número de *threads* e processos, para estabelecer o número de *threads* a serem utilizados, funções de âmbito geral que permitem a criação e gestão de semáforos, funções para temporização e medição de tempos, funções para paralelismo e para gestão dinâmica de *threads*.

4. Modelo de Programação utilizando Memória Distribuída (MPI)

No Modelo de Programação com Paradigma de Memória Distribuída [Kumar 2002] cada processador tem associado um bloco de memória. Assim, cada elemento pode acessar indiretamente um dos blocos associados a outros processadores. Desta forma, para conseguir a troca de dados é necessário que cada processador realize explicitamente a solicitação de dados aos processadores disponíveis, que serão os responsáveis pelo envio dos dados. Este modelo se baseia na técnica de passagem de mensagem. Existem vários ambientes de programação para esse modelo e o estándar atual chama-se MPI [MPI]. Apesar de não existir limites para número de processadores e cada processador acessa, sem interferência e rapidamente, sua própria memória, existe um elevado *overhead* devido a comunicação pelo envio e recebimento dos dados. A interface Message Passing Interfaces (MPI) provê uma base poderosa para construir programas com alta escalabilidade. Uma de suas metas de projeto é possibilitar a construção de softwares, que ajudam a resolver problemas aplicados a computação paralela e distribuída. Dentro do padrão MPI, uma aplicação será composta por diferentes processos que irão trocar informações úteis através de envios e recebimentos de mensagens.

5. Modelos de Programação Híbrido (MPI + OpenMP)

As aplicações em *clusters* podem ser programadas para utilizar troca de mensagens entres todos os processadores. Mas a possibilidade de um melhor desempenho utilizando-se um Modelo de Programação Híbrido [Kumar 2002] de comunicação com troca de informações tem como objetivo tirar partido das melhores características de ambos os modelos de programação, mesclando a paralelização explícita de grandes tarefas com o MPI com a paralelização de tarefas simples com o OpenMP. Para o problema proposto mesclou-se os Paradigmas de Memória Distribuída e Compartilhada em um único algoritmo, sendo uma união dos modelos. Adotou-se uma estratégia de paralelismo o qual utiliza um processo MPI por nó e *multithread* OpenMP sobre os *cores*, assim, apenas a *thread* principal faz chamadas MPI,

sendo responsável então, pela comunicação, onde as demais *threads* apenas realizam computação. Constatou-se que estratégias de escalonamento de tarefas inicializando múltiplos processos *multithreading* por nó obtinha-se um baixo desempenho, tendo uma saturação na gestão de tarefas por vários processos no mesmo nó.

6. Resultados Obtidos

Esta seção mostra os resultados experimentais do algoritmo implementado para o Modelo de Computação de Alto Desempenho utilizado na Espacialização do Relevo. Utilizou-se uma rede de máquinas Linux composta por 5 nodos homogêneos totalizando 10 processadores, com arquitetura Intel Xeon, biprocessador dual core, 2 GB de Memória denominado **Cluster SOL**, sol.inf.um.es, estando localizada no Laboratório de Computação Científica na Universidad de Murcia (UM) [LABCOCI 2011], Espanha. Para as experimentações, tentou-se buscar os melhores parâmetros de execução, tendo em vista a obtenção da máximo desempenho frente as características do ambiente de experimentação. elegeu-se de maneira experimental a melhor relação entre o número de *processos* e *threads*, dos quais se obtinha o menor tempo de execução para tamanhos de problemas propostos (Grau do Polinômio).

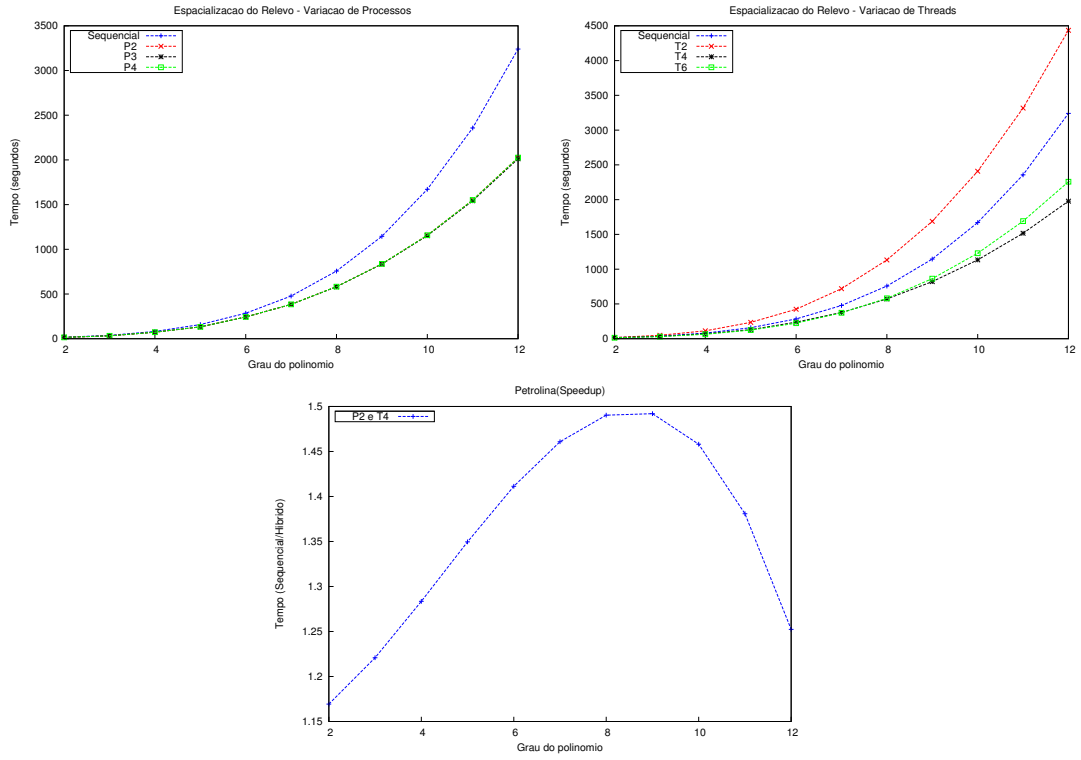

Figura 1. Representação gráfica dos tempos de execução (em segundos) variando o tamanho do problema, o número de processos e o número de threads e speedup variando o tamanho do problema.

Observando as Tabelas 1 e 2 e a Figura 1, percebe-se a busca para encontrar a melhor relação entre o número de *processos* e *threads* experimentado. Observa-se também na Tabela 1 que o tempo para o cálculo das tarefas diminui com o número de *processos*

Tabela 1. Representação tabular dos tempos de execução (segundos) variando o tamanho do problema e o número de processos.

Polinômio	Sequencial	2	3	4
2	16,387	15,187	15,325	15,179
4	81,036	72,692	72,674	72,998
6	285,001	242,92	243,017	242,980
8	757,028	581,848	581,054	582,082
10	1670,725	1157,51	1152,820	1157,710
12	3239,931	2025,796	2012,778	2026,138

Tabela 2. Representação tabular dos tempos de execução (segundos) fixando a quantidade de processos em 2 e variando o tamanho do problema e o número de threads.

Polinômio	Sequencial	2	4	6
2	16,387	17,014	15,224	14,134
4	81,036	112,993	71,644	64,078
6	285,001	424,299	238,91	223,236
8	757,028	1132,056	571,216	581,852
10	1670,725	2408,94	1133,22	1230,06
12	3239,931	4435,024	1978,632	2257,428

utilizados. No entanto, a redução do tempo cessa quando alcança dois *processos*. Caso semelhante ocorre durante a experimentação da busca do melhor número de *threads*, pois a redução do tempo diminui naturalmente até 4 *threads* uma vez que o número de cores no ambiente de experimentação é um biprocessador dual-core. Além disso, a assincronia aporta aos algoritmos uma superescalabilidade frente a memória distribuída, tendo na gerência das tarefas realizadas um equilíbrio das cargas de trabalho entre a execução das *threads*. A Figura 1, apresenta o comportamento no ambiente de experimentação, através do incremento do poder de computação nos resultados experimentais para um índice chamado de *speedup* [Kumar 2002].

7. Conclusões e Trabalhos Futuros

O modelo híbrido MPI + OpenMP apresenta-se sem dúvida como uma excelente ferramenta para o processamento de algoritmos paralelos em clusters. A possibilidade de realizar um bom balanceamento de carga entre as máquinas, somada a liberdade que o programador tem de definir quais trechos de código serão executados de forma paralela, demonstram o excelente potencial que o modelo possui para aplicações que exigem uma computação mais intensa. Com base nos resultados obtidos, verifica-se que a distribuição mais vantajosa consistiu em organizar o processamento de modo que cada nó execute um processo inicializando várias *threads*. A paralelização da etapa de resolução do sistema linear resultou em um ganho de rendimento pequeno, o que indica que também seria necessário paralelizar o trecho de código responsável pela composição das matrizes. É importante destacar que o modelo MPI + OpenMP não pode ser descartado, pois pode ser associado a outros modelos, resultando em uma implementação híbrida com um potencial de execução melhor.

Referências

Bajaj, C., Ihm, I., and Warren, J. (1993). Higher-order interpolation and least-squares approximation using implicit algebraic surfaces. *ACM Trans. Graph.*, 12:327–347.

Dagum, L. and Menon, R. (1998). OpenMP: An industry-standard API for shared-memory programming. *IEEE Comput. Sci. Eng.*, 5(1):46–55.

Golub, G. H. and Loan, C. F. V. (1989). *Matrix Computations*. JohnsHopkinsPress, Baltimore, MD, USA, second edition.

Kumar, V. (2002). *Introduction to Parallel Computing*. Addison-Wesley Longman Publishing Co., Inc.s, Boston, MA, USA, 2nd edition.

LABCOCI (2011). Laboratório de Computación Científica (Universidad de Murcia). Available in (2011) June 16: http://www.um.es/pcgum/.

MPI. Mensage Passing Interface. Available in (2011) June 16: http://www.lammpi.org.

Namikawa, L., Felgueiras, C., Mura, J., and Lopes, E. (2003). Modelagem numérica de terreno e aplicações. *INPE*, 1:158.

Nogueira, L., Abrantes, R. P., and Leal, B. (2008). A methodology of distributed processing using a mathematical model for landform attributes representation. In *Proceeding of the IADIS International Conference on Applied Computing*.

Rufino, I., ao, C. G., Rego, J., and Albuquerque, J. (2009). Water resources and urban planning: the case of a coastal area in brazil. *journal of urban and environmental engineering*, 3:32–42.

TBB (2011). Threading Building Blocks. Available in (2011) June 16: http://www.threadingbuildingblocks.org.