

# Aplicando Charm++ na Paralelização de um Sistema de Irrigação de Solos \*

Pablo J. Pavan<sup>1</sup>, Edson L. Padoin<sup>1,4</sup>,  
Laércio Pilla<sup>2</sup>, Philippe O. A. Navaux<sup>3</sup>, Jean-François Méhaut<sup>4</sup>

<sup>1</sup>Universidade Reg. do Noroeste do Estado do Rio G. do Sul (UNIJUI) - Ijuí - RS - Brasil

<sup>2</sup>Universidade Federal de Santa Catarina (UFSC) - Florianópolis - SC - Brasil

<sup>3</sup>Universidade Federal do Rio Grande do Sul (UFRGS) - Porto Alegre - RS - Brasil

<sup>4</sup>Laboratoire d'Informatique de Grenoble (LIG), Université de Grenoble - France

{pablo.pavan,padoin}@unijui.edu.br,

laercio.pilla@ufsc.br, navaux@inf.ufrgs.br, Jean-Francois.Mehaut@imag.fr

**Resumo.** Este artigo apresenta a análise de desempenho da paralelização de um sistema de irrigação de solos. O objetivo do trabalho é analisar a viabilidade da utilização de CHARM++ na paralelizações de aplicações reais. Os resultados alcançados apresentaram speed-up de até 11 vezes se comparado com a versão sequencial ainda sem a utilização de balanceamento de carga.

## 1. Introdução

Nas pesquisas realizadas em sistemas agroflorestais e agrícolas o comportamento da água no solo é de grande importância, uma vez que o movimento dos nutrientes depende do movimento da água no solo. Assim a descrição de como a água se desloca é fundamental nos projetos de irrigação, principalmente se considerado o sistema de irrigação por gotejamento, onde se faz necessário determinar a quantidade ótima de água necessária para o desenvolvimento das plantas em cada profundidade do solo [Padoin et al. 2011, Arruda et al. 2015].

Nesse sentido, considerando a necessidade de implementar sistemas que utilizam de maneira eficiente o poder computacional dos sistemas computacionais, novas tecnologias estão sendo desenvolvidas. Dentre elas, destaca-se o modelo de programação CHARM++ que pode ser utilizado tanto para sistemas multicomputadores quanto em sistemas multiprocessadores. Por meio desta ferramenta é possível dividir um problema em diversos componentes migráveis, chamados de *chares*, que são executados paralelamente em vários processadores [Kalé et al. 1998]. Assim, o objetivo deste trabalho é realizar uma análise da viabilidade da utilização de CHARM++ na paralelização de uma aplicação que simula um sistema de irrigação de solos.

O restante do trabalho está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. A Seção 3 apresenta a metodologia utilizada na implementação e o ambiente de execução utilizado na realização dos testes. Resultados são discutidos na Seção 4, seguido das Conclusões e Trabalhos Futuros.

---

\*Trabalho parcialmente apoiado por CNPq, CAPES, FAPERGS e FINEP. Pesquisa realizada no contexto do Laboratório Internacional Associado LICIA e tem recebido recursos do programa EU H2020 e do MCTI/RNP-Brasil sob o projeto HPC4E de número 689772.

## 2. Trabalhos Relacionados

Na literatura diversos trabalhos apresentam estudos sobre a movimentação da água no solo. Borges *et al.* (2005) apresenta uma análise do comportamento da água do solo saturado e não saturado por meio da equação de Richards. Para complementar o referido trabalho traz a implementação de um algoritmo utilizando o método de diferenças finitas. Entretanto, devido ao elevado tempo de processamento, utilizou-se somente simulação com matriz de pequena ordem.

No trabalho de Padoin *et al.* (2006), este sistema foi paralelizado com a biblioteca *PVM*, o que possibilitou a utilização de matrizes maiores gerando assim resultados mais precisos. Neste trabalho alcançou-se speed-up de 1,88 vezes com uma eficiência de até 38%. Em Padoin *et al.* (2008) o sistema foi resolvido pelo método do problema inverso e paralelizado para sistemas de memória distribuída com *MPI*. Já Padoin *et al.* (2011) em utilizou-se o algoritmo sequencial para analisar a eficiência energética de processadores e placas aceleradoras utilizando a paralelização através da plataforma *CUDA*.

Por outro lado, muitas pesquisas buscam balancear as cargas das tarefas entre os processadores almejando melhorar a eficiência da utilização destes sistemas paralelos [Kalé et al. 1998]. Deste modo, neste trabalho foi implementado uma nova versão paralela do algoritmo utilizando *CHARM++*.

## 3. Metodologia

Considerando que o sistema de irrigação de solo já tenha sido modelado em outros trabalhos, para execução paralela, fez-se um estudo do algoritmo sequencial [Padoin et al. 2011] e paralelo utilizando *MPI* [Padoin et al. 2006] para o desenvolvimento desta nova implementação. Este estudo foi realizado almejando determinar os fatores que demandam grande processamento e podem ser divididos entre os chares. Os principais fatores são:

- o número ótimo de células da malha, necessário para obtenção dos resultados com a precisão desejada; e
- o número de execuções do problema inverso (chute).

Para esta nova versão paralela, diferente das demais, optou-se por paralelizar o cálculo do problema inverso em busca do menor erro. Deste modo, a implementação consiste em que cada *chare* execute paralelamente um chute do problema inverso.

Na implementação em *CHARM++*, foi realizada a transformação do código que estava de forma procedural para o paradigma orientado a objeto. Para isso foram criadas duas classes utilizando a linguagem *C++*, uma classe *main* e outra classe *Irrigação*. As mesmas foram declaradas nos arquivos de cabeçalho (extensão *.h*), onde todas as variáveis foram declaradas como privadas e os métodos declarados como público.

Após definir os métodos de cada classe, foi necessário definir quais métodos seriam os *métodos de entradas* de cada classe, em seguida os mesmos foram declarados nos arquivos de extensão *.ci*. Toda a implementação das classes e dos métodos foi realizada nos arquivos com extensão *.C*.

A classe *main*, ficou responsável em realizar o cálculo da evaporação, inicializar o vetor de *chares* da classe *Irrigação*, fazer o *broadcast* para todos os *chares* através do método de entrada da classe *Irrigação*, passando por parâmetro os dados do cálculo da evaporação e

por fim, receber os dados calculados da classe *Irrigação* pelo seu método de entrada e realizar o cálculo do melhor caso. Sendo assim, a classe *Irrigação* recebe os dados de evaporação da classe *main*, realiza o problema inverso e devolve os dados para a classe *main*.

### 3.1. Metodologia de Execução

Para a análise de desempenho buscou-se variar os dois fatores que influenciam a aplicação, o tamanho da malha e o número de chutes do problema inverso. Assim para o primeiro fator buscou-se trabalhar com três tamanhos de malha: 128, 256 e 512. Para o segundo fator trabalhou-se com 100, 250 e 500 chutes, totalizando assim 9 configurações diferentes.

Cada configuração foi testada com diferentes números de cores do ambiente de execução, variando-se a quantidade de cores em 1, 2, 4, 8, 16 e 32 o que resultou em 54 testes diferentes. Cada um destes testes foi realizado 10 vezes para se obter a média aritmética.

### 3.2. Ambiente de Execução

O ambiente de execução possui dois processadores Intel Xeon modelo E5-2640 v2 de micro-arquitetura *Ivy Bridge*. Cada processador possui 8 cores com 2 SMT/core, o que totaliza 32 cores de 2.00 GHz de frequência. Este equipamento também possui 64 GB de memória RAM DDR3 de frequência 1600 MHz. Para os testes, utilizou-se o sistema operacional Ubuntu versão 3.16.0 – 70. A versão do CHARM++ utilizada foi 6.7.0 e do compilador g++ a versão 4.8.4.

## 4. Resultados

O tempo médio de execuções da aplicação sequencial foi de 58,95 s quando utilizado uma matriz de ordem 128 e 100 chutes para o problema inverso (Figura 1(a)). Observa-se um *overhead* de 26% no tempo de execução quando executado a versão paralela em somente 1 core. Este aumento no tempo de execução é resultado da criação do *array* de *chares* pelo CHARM++. Para os demais testes, com a versão paralela executando em apenas 1 core, independente da ordem da matriz e do número de chutes, observa-se o mesmo *overhead*.

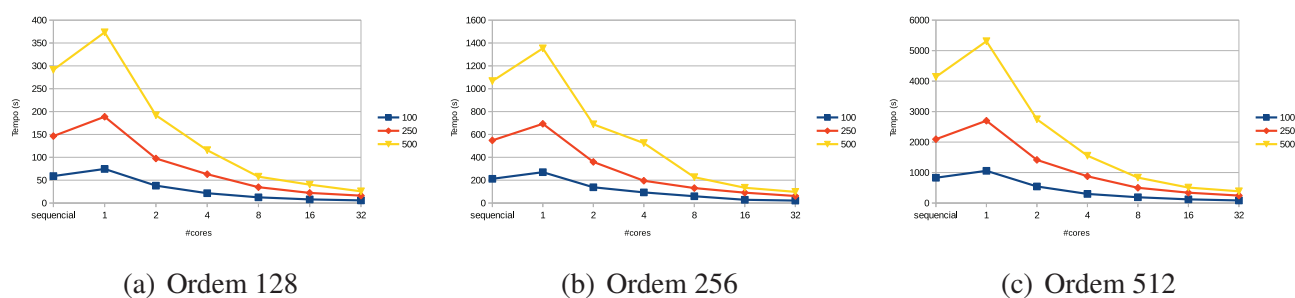


Figura 1. Tempo de execução (s) das simulações com diferentes ordens de matriz.

Percebe-se *speed-up* na versão paralela a partir de 2 cores, onde o *overhead* já é compensado com a redução no tempo total de execução. Para matriz de ordem 128 e 100 chutes, o tempo de execução foi de 37,99 s, o qual representa um *speed-up* de 1,55.

O maior *speed-up* para matriz de ordem 128 foi alcançado quando utilizado 500 chutes em 32 cores do sistema (linha amarela da Figura 1(a)). Neste teste o tempo de execução foi reduzido de 291,30 s para 25,6 s, o qual representa ganhos de até 11,36 vezes sobre o algoritmo sequencial.

Almejando aumentar a precisão utiliza-se matrizes de ordem 256 (Figura 1(b)) e 512 (Figura 1(c)) na simulação. Nestes testes, os tempos de execução da versão sequencial utilizado 500 chutes aumenta de 291,3 s para 1068,4 s e 4138,6 s respectivamente. Entretanto, nestes testes a redução no tempo total de execução foi proporcional, ou seja, para matrizes de ordem 256 tem-se *speed-up* de até 10,96 e para matrizes de ordem 512 tem-se *speed-up* de até 10,74.

## 5. Conclusões

Analisando os resultados, percebe-se que o aumento da ordem do sistema apresenta um aumento equivalente no tempo de execução da aplicação. Deste modo tem-se *speed-up* semelhante nas execuções paralelas independente da ordem de matriz simulada.

Como trabalhos futuros pretende-se executar aplicação de modo a simular um ambiente real com matrizes maiores. Por exemplo, se tomado como base uma matriz de ordem 65.536, a versão sequencial executada com 500 chutes tomaria 547,9 dias para chegar ao resultado. Com a versão paralela implementada em CHARM++, executando em 32 cores, o tempo seria proporcionalmente reduzido para 50 dias, o que ainda é elevado para pesquisas científicas na área de irrigação de solos em sistemas agroflorestais e agrícolas. Nesse sentido, como trabalhos futuros, pretende-se rodar a simulação paralela em sistemas computacionais que disponham de uma quantidade elevada de cores. Também, uma vez tendo a aplicação paralela implementada em CHARM++, pode-se utilizar balanceadores de carga almejando equilibrar a utilização dos recursos computacionais paralelos.

## Referências

- Arruda, G., Padoin, E. L., Pilla, L. L., Navaux, P. O. A., and Mehaut, J.-F. (2015). Proposta de balanceamento de carga para redução de migração de processos em ambientes multi-programados. In *XVI Simpósio de Sistemas Computacionais (WSCAD-WIC)*, pages 1–8, Florianópolis, RJ.
- Borges, P. A., Coelho, G., and Buligon, S. (2005). Análise do comportamento da água em solos saturados e não saturados. In *XVIII Congresso Nacional de Matemática Aplicada e Computacional, CNMAC, Santo Amaro, SP*.
- Kalé, L. V., Bhandarkar, M., and Brunner, R. (1998). Load balancing in parallel molecular dynamics. In *International Symposium on Solving Irregularly Structured Problems in Parallel*, pages 251–261. Springer.
- Padoin, E., Borges, P. A., Dill, S. L., and Padilha, N. R. (2008). Resolução do problema de absorção da água do solo através da paralelização do problema inverso. *VIII Escola Regional de Alto Desempenho*, pages 268–271.
- Padoin, E. L., DILL, S. L., BORGES, P. A., and BORGES, R. S. (2006). Paralelização de métodos computacionais aplicados à análise das variações do teor de umidade de solos saturados e não saturados. In *VII Workshop em Sistemas Computacionais de Alto Desempenho*, pages 97–104.
- Padoin, E. L., Pilla, L. L., Boito, F. Z., Kassick, R. V., and Navaux, P. O. (2011). Análise do consumo energético do algoritmo de irrigação de solos em arquiteturas heterogêneas. *Conferencia Latino Americana de Computación de Alto Rendimiento, Colima, Mexico. CLCAR*, pages 1–7.