SAULO HENRIQUE CABRAL SILVA

Orientador: Joubert de Castro Lima
Co-orientador: Tiago Garcia de Senna Carneiro

TERRAME HPA - UMA ARQUITETURA BASEADA EM MEMÓRIA COMPARTILHADA PARA MODELAGEM E SIMULAÇÃO DE SISTEMAS TERRESTRES

Ouro Preto
Dezembro de 2011

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

TERRAME HPA - UMA ARQUITETURA BASEADA EM MEMÓRIA COMPARTILHADA PARA MODELAGEM E SIMULAÇÃO DE SISTEMAS TERRESTRES

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

SAULO HENRIQUE CABRAL SILVA

Ouro Preto
Dezembro de 2011

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

TerraME HPA - Uma Arquitetura Baseada em Memória Compartilhada para modelagem e simulação de sistemas terrestres

SAULO HENRIQUE CABRAL SILVA

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Joubert de Castro Lima – Orientador
Universidade Federal de Ouro Preto

Dr. Tiago Garcia de Senna Carneiro – Co-orientador
Universidade Federal de Ouro Preto

Dr. Ricardo Augusto Rabelo
Universidade Federal de Ouro Preto

Dr. Fabrício Benevenuto de Souza
Universidade Federal de Ouro Preto

Ouro Preto, Dezembro de 2011

Resumo

O TerraME é uma plataforma de modelagem e simulação destinado ao desenvolvimento de modelos dinâmicos espacialmente explícitos que vem sendo utilizada pelo projeto GEOMA para para a modelagem e simulação de processos de mudança de uso e cobertura do solo (LUCC) para toda região amazônica. Já na FIOCRUZ vem sendo utilizado para o desenvolvimento de modelos epidemiológicos [Saude, 2006], como são os casos do controle da Dengue nas cidades do Rio de Janeiro e Recife. Além do TerraME, temos plataformas como *Swarm* [Minar, 1996], *Stella* [Costanza e Voinov, 2001], *Vensim*[Systems., 2004] como alternativas de plataforma de modelagem. Infelizmente, nenhuma plataforma está desenhada para ser executada em arquitetura de computadores com memória compartilhada ou distribuída. Diante deste cenário, propomos o TerraME HPA (
textitTerraME High Performance Archtecture). Na primeira versão do TerraME HPA é possível calibrar um modelo em paralelo e realizar uma simulação em paralelo. Os resultados mostram que o TerraME HPA possui runtime linear e speedup de 7.62 ao utilizar 8 núcleo de processamento, o que sugere *Amdhal* como sendo cerca de 95% de paralelismo no TerraME HPA. Ainda estamos exaurindo a plataforma com inúmeros experimentos com o intuito de comprovar os resultados alcançados.

Abstract

TerraME is a modeling and simulation platform for the development of spatially explicit dynamic models that has been used for the project GEOMA for modeling and simulation of land use change and land cover (LUCC) for the entire Amazon region. In the FIOCRUZ has been used for the development of epidemiological models, as are the control of Dengue cases in the cities of Rio de Janeiro and Recife. TerraME Besides, we have platforms like Swarm, Stella Stella, Vensim as alternative modeling platform. Unfortunately, no platform is designed to run on computer architecture with shared memory or distributed. In this scenario, we propose the TerraME HPA (High Performance TerraME Archtecture). In the first version of TerraME HPA is possible to calibrate a model in parallel and perform a simulation in parallel. The results show that the HPA has TerraME runtime linear speedup of 7.62 when using 8 core processing, which suggests Amdhal to be about 95% of parallelism in TerraME HPA. We are still draining the platform with numerous experiments in order to verify the results.

Dedico este trabalho a Deus, a minha família e meus amigos por me darem forças para conquistar mais este objetivo traçado em minha vida.

Agradecimentos

Agradeço primeiramente a Deus pelo dom da vida e por ser sempre meu guia em todos os momento.

A toda minha família. Em especial, aos meus pais, Edmar José Silva e Regina Célia Cabral Silva, por não medirem esforços em me ajudar a conquistar mais este objetivo, verdadeiramente são os maiores mestres que tenho, obrigado pela força, dedicação, amor e compreensão que me deram ao longo de minha vida.

Aos meus amigos da República Molotov, peço obrigado a Deus todos os dias, por me ter concedido o privilégio de ter suas amizades.

Aos meus amigos que durante a jornada me acompanharam nos momentos de descontração e sempre torceram pelo meu sucesso.

Aos professores David Menotti,José Maria, Luiz Merschmann, Sica, Ricardo Rabelo, Andre Luiz Lins de Aquino. Em especial ao meu orientador Joubert de Castro Lima e meu Co-Orientador Tiago Garcia de Senna Carneiro, pela dedicação, preocupação e esforço para produzir um bom trabalho. Todos participaram dessa jornada comigo, ou antes dela forneceram-me os meios para a travessia.

"Os homens conseguiriam muito mais coisas se julgassem menos coisas impossíveis!"(Malesherbes)

Sumário

1 Introdução **1**

2 Conceitos Básicos **4**
2.1 Modelagem 4
2.2 TerraME 5
2.3 A Linguagem LUA 7
2.4 TerraLib 9
2.5 TerraView 9
2.6 Computação Paralela 9

3 Calibração Paralela de Modelos no TerraME **12**
3.1 Método Genético 13
3.2 Método de calibração Monte Carlo 17
3.3 Estratégia de paralelização dos algoritmos de calibração no TerraME 18

4 Simulação paralela no TerraME **22**

5 Experimentos **27**
5.1 Experimentos para calibração paralela 27
5.1.1 Testes para o Método de Monte Carlo 27
5.1.2 Testes para o Método Genético 28
5.2 Experimentos para simulação paralela 29

6 Conclusões **32**
6.1 Trabalhos futuros 32

Referências Bibliográficas **34**

Lista de Figuras

2.1 Ambiente de desenvolvimento e suporte a modelagem ambiental TerraME 6
2.2 Arquitetura do Ambiente TerraME 6
2.3 Exemplo de código LUA 7

3.1 Operador cruzamento 14
3.2 Operador mutação 14
3.3 Pseudocódigo do algoritmo Genético 16
3.4 Passos de execução Monte Carlo 18
3.5 Exemplo de uma calibração paralela no TerraME 19
3.6 Instrumentação necessária para utilizar a calibração paralela dos métodos de Monte Carlo e Genético 20
3.7 Interpretador para o módulo de calibração paralela 21

4.1 Interpretador do módulo TerraME HPA 23
4.2 Arquitetura do módulo TerraME HPA 24
4.3 Estrutura de um Worker 25
4.4 Exemplo da instrumentação de um código, para este execute em paralelo 26

5.1 Speedup para o teste de Monte Carlo 28
5.2 Speedup para o teste Genético 29
5.3 Speedup para os testes utilizando o módulo TerraME HPA 30

Lista de Tabelas

3.1 Plataformas vs. métodos de calibração 13
3.2 Parâmetros do algoritmo Genético 15

Lista de Algoritmos

Capítulo 1

Introdução

Os sistemas Terrestres hoje estão submetidos em profundas mudanças cada vez mais intensas que vêm alterando seu funcionamento. Essas alterações têm trazido um forte impacto direto na integridade do meio ambiente e na qualidade de vida das pessoas. Diversos estudos apontam as ações humanas como à principal causa das alterações dos biomas [Salazar L. F. e D., 2007].

A intensificação das mudanças ambientais causadas por processos antrópicos, cujos impactos são quase sempre negativos ao ambiente, exige que pesquisadores, empresários, e governos forneçam respostas aos desafios científicos e tecnológicos ligados ao entendimento do funcionamento dos sistemas terrestres, mas para isso eles precisam de ferramentas de modelagem que sejam confiáveis e capazes de capturar a dinâmica e os resultados das dinâmicas das ações da sociedade humana. Modelos computacionais que reproduzem de uma forma satisfatória o fenômeno sob estudo contribuem para o ganho do conhecimento científico no que diz respeito ao seu funcionamento e este conhecimento pode servir como uma base inicial para o planejamento e definição de políticas públicas através da sua otimização e obtenção de melhores resultados (chamadas políticas saudáveis).

Planejadores, decisores e todos aqueles que se ocupam em projetar e estabelecer políticas públicas precisam de ferramentas de modelagem que sejam confiáveis e capazes de capturar a dinâmica e os resultados das ações humanas [Lambin, 1994].O TerraME é uma plataforma de modelagem e simulação de domínio público para o desenvolvimento de modelos ambientais espacialmente explícitos integrados a um Sistema de Informação Geográfica (SIG), desenvolvido pela parceria TerraLAB-UFOP (Universidade Federal de Ouro Preto) e INPE (Instituto Nacional de Pesquisas Espaciais). Esta plataforma fornece uma linguagem de modelagem de alto nível que é utilizada para a descrição dos modelos/algoritmos e sua posterior integração aos bancos de dados geográficos.

O TerraME possui suporte a modelagem ambiental espacial dinâmica, que suporta modelos de computação baseados em autômatos celulares [Wolfram, 2003] e conceitos de autômatos celulares aninhados (Nested-CA) [Carneiro, 2006]. Um modelo espacial dinâmico é uma representação abstrata de um fenômeno que evolui no tempo e no espaço, baseado em descrições

de entidades, processos e relações entre eles.

O TerraME está associado a um Sistema de Informação Geográfica (SIG) que fornece a localização espacial dos dados. Os resultados destes modelos são mapas que mostram a distribuição espacial de um padrão ou de uma variável contínua.

TerraME permite simulação em duas dimensões de espaços celulares sejam estes regulares e irregulares. Ele ainda suporta desenvolvimento de modelos baseados em agentes, oferece ao modelador funções que auxiliam na criação de modelos que utilizam conceito de multi-escala e multi-paradigma e ainda trabalha com duas importantes inovações que são modelos de autômatos híbridos [Henzinger, 1996] e espaços anisotrópicos [Xiangrong Li, 1998]. Até o fechamento deste trabalho não foi encontrada outra plataforma com características similares ao TerraME. Tais particularidades do TerraME são úteis ao criar modelos que demandam um comportamento contínuo e discreto.

Dentre os principais problemas entre as plataformas de simulação existentes hoje podemos destacar o tempo de execução do modelo e a quantidade de dados utilizada como sendo os mais graves. Com o aumento das mudanças ambientais esses problemas tendem a piorar ainda mais,pois para tentar reproduzir a dinâmica dos fenômenos naturais os modelos agora devem considerar mais variáveis, mais processos passam a interferir na dinâmica do fenômeno e mais dados são necessários. Para que o modelo seja executado em tempo hábil o modelador muitas das vezes tem que omitir alguns pontos do modelo que prejudicam a fidelidade ao tentar expressar a dinâmica do fenômeno.

Atualmente as plataformas mais utilizadas baseadas em Teoria Geral dos Sistemas,Teoria de Agentes e autômatos celulares são: *Swarm* [Minar, 1996], *Stella* [Costanza e Voinov, 2001], TerraME [Carneiro, 2006], *Dinamica-EGO* [Soares, 2003] e *Repast*. Infelizmente, tais plataformas não são desenhadas para arquitetura de computadores com memória compartilhada ou distribuída.

Programar para tais arquiteturas de computadores não é trivial. Na literatura temos a programação baseada no endereçamento compartilhado e troca de mensagens. Explicitar tal complexidade ao modelador não é razoável, portanto é fundamental o desenvolvimento de plataformas de modelagem e simulação que sejam passíveis de serem executas em arquiteturas de computadores de alto desempenho, porém que mantenham os modelos existentes nas versões sequenciais.

Diante deste cenário estendemos a plataforma TerraME inserimos um conjunto de diretivas que possibilitam a execução paralela dos modelos construídos pelo usuário de forma intuitiva e sem modificação no código que ate então era executado de forma sequencial. Utilizamos o conceito de anotações para instrumentar o modelo do usuário.

De uma forma geral o TerraME HPA escala bem em arquiteturas de computadores com memória compartilhada. Em uma máquina com oito núcleos de processamento atingimos um Speedup de até 7.62, o que sugere *Amdhal* como sendo cerca de 95% de paralelismo. Para efe-

tuar os testes, utilizamos modelos complexos, um deles é o Throll [Chave, 1999]. Este modelo representa o crescimento florestal, projetado para funcionar em grandes parcelas de florestas naturais densas, com ênfase para florestas neotropicais. A resolução adotada no modelo é de 1m e a competição por luz é espacialmente explicita, ocorrendo dentro dos denominados voxels de 1m x 1m x 0.5m. A competição por luz é modelada calculando exatamente o campo tridimensional da radiação fotossinteticamente ativa na floresta sub-bosque. Também é considerado no modelo o crescimento e mortalidade das árvores assim como a dispersão das sementes a dormência e sucesso do estabelecimento.

O restante deste trabalho esta organizado da seguinte forma: Capítulo 1 é uma introdução objetivando, principalmente, apresentar as motivações e objetivos dessa monografia. O Capítulo 2 conceitua modelagem ambiental e a plataforma TerraME. Apresentamos também quais os principais problemas que as plataformas Espaço temporais enfrentam e as soluções que adotamos para contorná-los na plataforma TerraME. No Capítulo 3 descreve os testes efetuados e apresenta os resultados do módulo de calibração paralela e para o módulo TerraME HPA. Finalmente no Capítulo 4 apresentamos as conclusões finais e discussões sobre os resultados obtidos.

Capítulo 2

Conceitos Básicos

Vamos identificar neste capítulo principais objetivos e características da modelagem ambiental. Detalharemos também a estrutura e o funcionamento da plataforma TerraME, juntamente com os principais pontos fortes da linguagem LUA. Vamos abordar também dois softwares livres o *TerraLib* e o *TerraView* que podem trabalhar em conjunto com o TerraME e por fim apresentamos o paradigma de computação paralela.

2.1 Modelagem

Computadores vêm sendo utilizados como ferramentas de apoio a procedimentos de estudos, de análises e de simulações em vários campos do conhecimento humano. Sistemas complexos para análises e para modelagens foram desenvolvidos para se trabalhar dados relacionados com áreas específicas como finanças, transportes, geologia, solos, etc.. Nessa mesma tendência, sistemas de armazenamento, manipulação e apresentação de dados espaciais, conhecidos como SIG, foram criados e estão sendo utilizados no campo das ciências ambientais [Felgueiras, 1999].

Modelagem ambiental é qualquer abstração do mundo natural com algum poder de prever eventos com base em princípios gerais. Utilizando ferramentas de SIG, é possível prever o tamanho do impacto que um fenômeno que esta sendo observado causará. Por exemplo, o aumento da temperatura global poderá implicar em explosões populacionais de vetores de doença tropicais e resultar em epidemias globais [Hales S., 2002] [Githeko, 2000] [Patz, 1998] [Martens, 1995]. Essas simulações podem ser feitas em vários tipos de sistemas, como os sistemas ecológicos, sistemas climáticos, sistema hidrológicos,sistemas de uso do solo. Com o objetivo de representar propriedades e facilitar o estudo e compreensão, modelos matemáticos, aritméticos e lógicos vem sendo implementados [Felgueiras, 1999].

As ferramentas de modelagem devem ser capazes de simular os fenômenos escolhidos para estudo levando em consideração a dinâmica e os resultados das ações humanas. De acordo com os resultados obtidos na modelagem aqueles que são responsáveis por projetar e estabelecer políticas públicas tem condições de analisar os impactos das suas decisões [Lambin, 1994]

[Kaimowitz e Angelsen, 1998] [Turner II, 1995]. A simulação de processos naturais ou a simulação das interações humano-ambiente são instrumentos de pesquisa de impactos e previsão [Batistella, 2005] que têm potencial como meio de detectar, identificar, mapear e monitorar alterações nos ecossistemas, independentemente dos seus agentes causais [Coppin, 2004]. Desta forma, é clara a necessidade do desenvolvimento de técnicas e tecnologias que permitam o melhor entendimento do funcionamento dos sistemas Terrestres e a simulação de seu comportamento. Tais artefatos deverão ser utilizados para a análise de cenários alternativos gerados por simulações computacionais que servirão de base para o planejamento e definição de políticas públicas [Lana, 2009].

2.2 TerraME

TerraME é uma plataforma de desenvolvimento e suporte a modelagem ambiental espacial dinâmica que suporta modelos de computação baseados em autômatos celulares [Wolfram, 2003] e conceitos de autômatos celulares aninhados [Carneiro, 2006]. Um modelo espacial dinâmico é uma representação abstrata de um fenômeno que evolui no tempo e no espaço, baseado em descrições de entidades, processos e relações entre eles. Para que, o TerraME possa executar modelos espaciais, ele foi associado a um Sistema de Informação Geográfica (SIG) que fornece a localização espacial dos dados que são utilizados no modelo. Os resultados destes modelos são mapas que mostram a distribuição espacial de um padrão ou de uma variável contínua. Maiores detalhes sobre o TerraME podem ser encontrados em [Carneiro e Câmara, 2007].

O TerraME permite simulação em duas dimensões de espaços celulares regulares e irregulares. Entre as aplicações típicas do TerraME podemos citar a modelagem e simulação de processos de mudança de uso e cobertura do solo para toda a região Amazônica no âmbito do projeto GEOMA, desenvolvimento de modelos de drenagem da água das chuvas, modelos de propagação de incêndios em florestas, e modelos epidemiológicos junto à FIOCRUZ, como no caso do controle da Dengue nas cidades do Rio de Janeiro (RJ) e Recife (PE).

A parte fundamental do desenvolvimento do ambiente TerraME é seu interpretador, o ilustramos na figura 2.1. Ele lê um modelo descrito na linguagem de modelagem TerraME, que por sua vez é uma linguagem estendida da linguagem LUA, interpreta o código fonte, e chama as funções do plataforma TerraME. Esta plataforma é um conjunto de módulos escritos em C++ que oferecem funções e classes para modelagem espacial dinâmica e acesso ao banco de dados geográficos *TerraLib*. Os resultados da simulação dos modelos desenvolvidos podem ser vistos com a aplicação *TerraView*.

A Figura 2.2 descreve o arquitetura da plataforma de modelagem TerraME. Na primeira camada, a TerraLib oferece serviços típicos de sistemas de informação geográfica (SIG), gestão de dados espaciais, serviços de análise e funções extras para a manipulação de dados tempo-

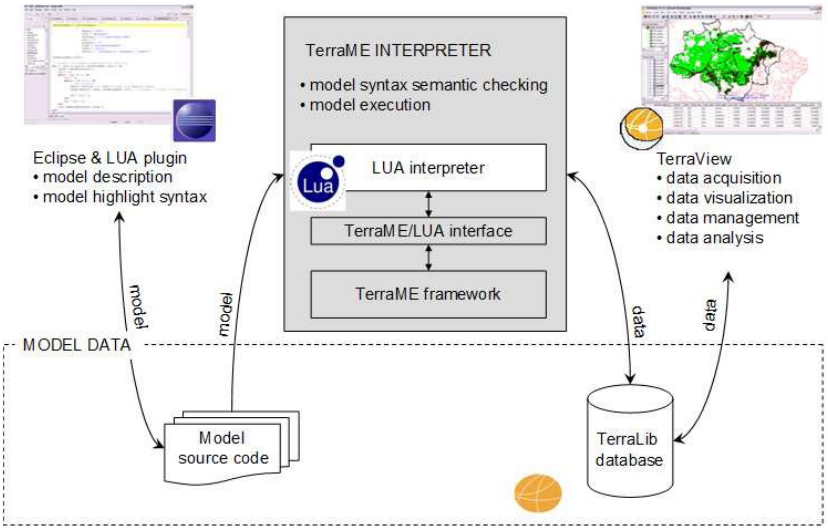

Figura 2.1: Ambiente de desenvolvimento e suporte a modelagem ambiental TerraME

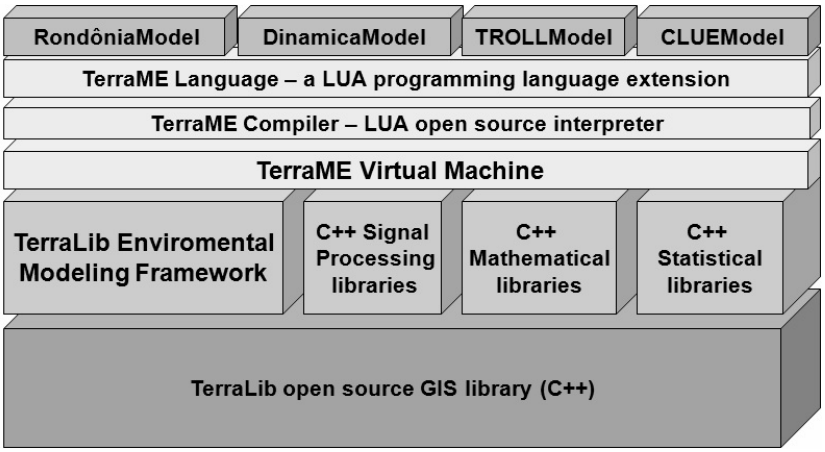

Figura 2.2: Arquitetura do Ambiente TerraME

rais. Na segunda, a plataforma TerraME provê a simulação e os serviços de calibração e validação. Foi desenvolvido para ser independente da plataforma. Esta plataforma pode ser utilizada diretamente para desenvolvimento de modelo, porém o desenvolvimento de modelos na linguagem C++ pode ser um desafio para modeladores, por isso, o TerraME proporciona uma linguagem de modelagem de alto nível.

A terceira camada da arquitetura implementa a linguagem de modelagem TerraME e o ambiente de execução. A interface TerraME/LUA estende a linguagem de programação LUA com os novos tipos de dados espaciais para modelagem dinâmica e serviços para simulação e avaliação do modelo. Para tornar possível utilizar a plataforma através do interpretador LUA, foi necessário exportar a API do TerraME para a API de LUA, assim o TerraME reconhece os tipos descritos no modelo. Através dessa exportação outras aplicações escritas em C ou

C++ podem ter suas APIs exportadas para a linguagem LUA. A última camada é chamada camada de aplicação e inclui os modelos de usuário final.

2.3 A Linguagem LUA

LUA [Ierusalimschy, 2003] é uma linguagem de programação poderosa, rápida e leve, projetada para estender aplicações.

LUA combina sintaxe simples para programação procedural com poderosas construções para descrição de dados baseadas em tabelas associativas e semântica extensível. LUA é tipada dinamicamente, é interpretada a partir de bytecodes para uma máquina virtual baseada em registradores, e tem gerenciamento automático de memória com coleta de lixo incremental. Essas características fazem de LUA uma linguagem ideal para configuração, automação (scripting) e prototipagem rápida.

Na figura 2.3, exemplificamos a codificação da linguagem LUA, em um contador de palavras que são provenientes de um arquivo.

```
1  --instanciando tabelas
2  T1 = {}
3  T2 = {}
4  T_TOTAL = {}
5
6  --declaração de função
7  function CONTADOR_ARQ(indx,T)
8      --leitura do arquivo armazenado no caminho especificado
9      local fread = io.open("D:\\A"..indx.."\\Arq"..indx..".txt");
10     --leitura de todo o arquivo .txt
11     for line in fread:lines() do
12         --verifica se existe entrada na tabela com a string(palavra) "lida" doo arquivo
13         if T[line] == nil then
14         --inserção de mais um elemento na tabela
15             table.insert(T,line)
16             T[lisne] = 1
17         else
18         --modificação de um elemento da tabela
19             T[line] = T[line] + 1
20         end
21     end
22 end
23 --chamada de função com seus respectivos parâmetros
24 CONTADOR_ARQ(1,T1) CONTADOR_ARQ(2,T2)
25 --cópia de tabela
26 T_TOTAL = T1
27 --percorrer uma tabela por completo
28 for ind=1,table.maxn(T2) do
29     if T_TOTAL[T2[ind]] == nil then
30         table.insert(T_TOTAL,T2[ind])
31         T_TOTAL[T2[ind]] = 1
32     else
33         T_TOTAL[T2[ind]] = T_TOTAL[T2[ind]] + T2[T2[ind]]
34     end
35 end
```

Figura 2.3: Exemplo de código LUA

Na linha 1,podemos ver um exemplo de comentário na linguagem LUA, todos os comen-

tários são realizados, iniciando-se com $--$. Das linhas 2 - 4, são realizadas instanciações de tabelas globais, estas tabelas podem ser acessadas em qualquer parte do código, na linguagem considera-se que qualquer variável instanciada sem a palavra reservada *local*, sendo uma variável global. Nas linhas 7 - 22, estamos representando o escopo de uma função cujo nome na aplicação é: *"CONTADOR_ARQ"*, e seus parâmetros são respectivamente *indx* que deve ser um tipo *numeral* e *T* que é do tipo *table*. Na linha 9 efetuamos a abertura de um fluxo de leitura para o arquivo localizado no diretório passado como parâmetro para a função *io.open*, note como é bem intuitivo criar fluxo de dados a partir da linguagem LUA.

Das linhas 11 - 21 e 28 - 34, apresentamos exemplos de laços de repetição.O primeiro laço de repetição é responsável por *"varrer"* todo o arquivo, e o segundo efetua um *"merge"* entre as duas tabelas utilizadas no exemplo. Nas linhas 13 - 16, temos a parte condicional, note que o comando *if* na linha 13 executa as instruções dentro do escopo da mesma, caso a condição passada seja verdadeira. LUA oferece suporte ao usuário quanto a operações referentes a tabelas, na linha 15 do exemplo, temos uma operação de inserção de um dado elemento (no nosso exemplo *linha*) na tabela, no exemplo denominada *T*.

Os principais fatores que levam a escolha da linguagem LUA é pelo fato de:

1. LUA é uma linguagem estabelecida e robusta
2. LUA é rápida
3. LUA é portátil
4. LUA é embutível
5. LUA é poderosa(e simples)
6. LUA é pequena
7. LUA é livre
8. LUA tem importância global

LUA é inteiramente projetada, implementada e desenvolvida no Brasil, por uma equipe na PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro). LUA nasceu e cresceu no Tecgraf, o Grupo de Tecnologia em Computação Gráfica da PUC-Rio. Atualmente, LUA é desenvolvida no laboratório Lablua. Tanto o Tecgraf quanto Lablua são laboratórios do Departamento de Informática da PUC-Rio.

A plataforma TerraME utiliza uma extensão de LUA, incluindo os tipos TerraME, para que o modelador evite a complexidade do C++.

2.4 TerraLib

TerraLib é uma biblioteca open source GIS que adota SGBD (Sistema Gerenciador de Banco de Dados) relacional para manipular tipos de dados espaço-temporais. A biblioteca suporta *SGDB* diferentes, incluindo *MySQL*, *PostgreSQL* e *Oracle*. O *TerraLib* suporta o desenvolvimento de aplicações geográficas usando bancos de dados espaciais.

O objetivo do projeto para a *TerraLib* é apoiar aplicações em larga escala utilizando dados socioeconômicos e ambientais. Ele lida com tipos de dados espaço-temporal (eventos, objetos em movimento, espaços celulares, objetos modificáveis) e permite consultas espaciais, temporais e atributo no banco de dados. O *TerraLib* suporta modelagem dinâmica em espaços celulares generalizado e tem um vínculo dinâmico com a linguagem de programação R para análise estatística.

TerraLib é desenvolvido com a parceria entre a Divisão de Processamento de Imagem do *INPE*, e a *PUC-RIO*. O *TerraLib* é licenciado como código aberto de acordo com a *LGPL*.

Na plataforma TerraME o *TerraLib* é usado pelo *kernel* TerraME para manipulação de dados espaço-temporais.

2.5 TerraView

TerraView é um aplicativo GIS livre distribuído sob a licença *GPL*, foi construído sobre a biblioteca *TerraLib* GIS. O *TerraView* manipula dados vetoriais (polígonos, linhas, pontos) e dados raster (grades e imagens), ambos armazenados em um banco de dados relacionais ou geo-relacionais, incluindo *ACCESS*, *PostgreSQL*, *MySQL* e *Oracle Spatial*. *TerraView* possui uma interface que permite a visualização de atributos e consultas espaciais sobre o objeto no banco de dados geográficos. A interface permite visões diferentes sobre o banco de dados, produzindo mapas temáticos com diferentes tipos de legendas.

TerraView é capaz de gerenciar os dados *raster's* no banco de dados geográficos e permite a visualização e manipulação de dados raster em conjunto com dados vetoriais. Dados raster podem ser compartilhados em diferentes formatos, como *GeoTIFF*, *TIFF*, *JPEG*, *RAW*, *ASCII-Grid* ou *ASCIISpring*.

TerraView suporta operações com vetores, incluindo cruzamento e mapas de buffer. Ela também tem funções de análise estatística: índices de autocorrelação *local* e *global*, semivariogramas e regionalização.

O *TerraView* é o atual visualizador de simulações na plataforma TerraME.

2.6 Computação Paralela

Tradicionalmente, o software tem sido escrito para ser executado sequencialmente. Para resolver um problema, o algoritmo é construído e implementado como um fluxo serial de in-

struções. Tais instruções são então executadas por uma unidade central de processamento de um computador. Somente uma instrução pode ser executada por vez; após sua execução, a próxima então é executada.[Barney, 2007]

Por outro lado, a computação paralela faz uso de múltiplos elementos de processamento simultaneamente para resolver um problema. Isso é possível ao quebrar um problema em partes independentes de forma que cada elemento de processamento possa executar sua parte do algoritmo simultaneamente com outros. Os elementos de processamento podem ser diversos e incluir recursos como um único computador com múltiplos processadores, diversos computadores em rede, hardware especializado ou qualquer combinação dos anteriores.[Barney, 2007]

A Computação paralela é então uma forma de computação em que vários cálculos são realizados simultaneamente [Almasi e Gottlieb, 1989] operando sob o princípio de que grandes problemas geralmente podem ser divididos em problemas menores, que então são resolvidos concorrentemente (em paralelo). Existem diferentes formas de computação paralela: em *bit*, instrução, de dado ou de tarefa. A técnica de paralelismo já é empregada há vários anos, principalmente na computação de alto desempenho, mas recentemente o interesse no tema cresceu devido às limitações físicas que previnem o aumento de frequência de processamento.[Adve S.V., 2008] Com o aumento da preocupação do consumo de energia dos computadores, a computação paralela se tornou o paradigma dominante nas arquiteturas de computadores sob forma de processadores multinúcleo.[Asanovic, 2006b]

Computadores paralelos podem ser classificados de acordo com o nível em que o hardware suporta paralelismo. Computadores com multinúcleos ou multiprocesadores possuem múltiplos elementos de processamento em somente uma máquina, enquanto *clusters*, *MPP* e grades usam múltiplos computadores para trabalhar em uma única tarefa. Arquiteturas paralelas especializadas às vezes são usadas junto com processadores tradicionais, para acelerar tarefas específicas.

Programas de computador paralelos são mais difíceis de programar que sequenciais,[Patterson e Hennessy, 1998] pois a concorrência introduz diversas novas classes de defeitos potenciais, como a condição de corrida. A comunicação e a sincronização entre diferentes subtarefas é tipicamente uma das maiores barreiras para atingir eficiência em programas paralelos.

Com o desenvolvimento de computadores paralelos, torna-se mais viável resolver problemas anteriormente muito demorados para se executar. A computação paralela é usada em diversos campos, da bioinformática (para o enovelamento de proteínas) a economia (para simulações de matemática financeira). Tipos comuns de problemas encontrados em aplicações de computação paralela são:[Asanovic, 2006a]

1. Problemas de grades estruturadas

2. Problemas de grades não estruturadas

3. Método de Monte Carlo
4. Simulação de máquina de estado finito
5. Visita de grafo

Capítulo 3

Calibração Paralela de Modelos no TerraME

O uso de dados empíricos para se calibrar um modelo é uma forma importante para dar realismo na dinâmica do sistema, portanto não se podem prescindir as etapas de calibração e validação do modelo. Dependendo da complexidade do modelo e do método escolhido para realizar a calibração, o tempo da execução da calibração pode chegar a levar dias, semanas ou meses. Ao se desenvolver modelos espacialmente explícitos baseados em autômatos celulares, o número de parâmetros a serem ajustados pode ser muito elevado,isso aumenta ainda mais o custo para a calibração do respectivo modelo.

Os autômatos celulares (AC) foram introduzidos nos anos 50 por John Von Neumann. Os Autômatos Celulares podem ser definidos como uma grade ou rede, cujos nós que as compõem são chamados células. A propriedade de destaque para aplicação de Autômatos Celulares em modelagem é o seu caráter evolutivo proporcionado por interações entre suas células. Estas células interagem através de algoritmos computacionais ou funções de transição, partindo de configurações de estados presentes e evoluindo para novas configurações.

Atualmente, existem vários métodos para se calibrar modelos baseados em Autômatos Celulares entre eles o Método de Monte Carlo [Larissa M. Fraga, 2009], Algoritmo Genético [Larissa M. Fraga, 2009], Método dos quadrados Mínimos [Tellinghuisen, 2010], Mínimos quadrados não-linear ponderado [Katiani da Conceição Loureiro, 2010].

O Método de Monte Carlo é um modelo de simulação que utiliza a geração de números aleatórios para atribuir valores às variáveis que se deseja investigar. Os números podem ser obtidos através de algum processo aleatório (tabelas, roletas, etc.) ou diretamente do computador, através de funções específicas [Lustosa P. R. B., 2004].

Algoritmo genético (AG) é uma técnica de procura utilizada para encontrar soluções aproximadas em problemas de otimização e busca. Eles consistem numa classe particular de algoritmos evolutivos que usam técnicas inspiradas na biologia evolutiva como hereditariedade, mutação, seleção natural e recombinação [Larissa M. Fraga, 2009].

O Método dos Mínimos Quadrados, ou Mínimos Quadrados Ordinários (MQO) ou OLS (do inglês Ordinary Least Squares) é uma técnica de otimização matemática que procura encontrar o melhor ajustamento para um conjunto de dados tentando minimizar a soma dos quadrados das diferenças entre o valor estimado e os dados observados (tais diferenças são chamadas resíduos). Um requisito para o método dos mínimos quadrados é que o fator imprevisível (erro) seja distribuído aleatoriamente, essa distribuição seja normal e independente. O Teorema de Gauss-Markov garante (embora indiretamente) que o estimador de mínimos quadrados é o estimador não-enviesado de mínima variância linear na variável resposta.

O TerraME possui os métodos de calibração Monte Carlo e Genético, ambos sequenciais. A tabela abaixo 3.1 apresenta uma comparação entre os métodos de calibração existentes em cada plataforma, a tabela mostra também se as plataformas existentes oferecem o serviço de calibração paralela ou não. Até o fechamento deste trabalho, não encontramos nenhuma de modelagem e simulação que possua calibração paralela de modelos.

Plat./Metod.	**Genético**	**M. Carlo**	**Min. Quadrado**
Stella	-	+	+
Dinamica-EGO	+	+	-
Swarm	+	+	+
Repast	+	+	-
TerraME	+	+	-

Tabela 3.1: Plataformas vs. métodos de calibração

− Não possui o método de calibração.

+ Possui o método de calibração.

Pela tabela 3.1 é possível observar que não existe plataforma espaço temporal que forneça calibração de modelos de forma paralela. Uma das alternativas para solucionar o problema do tempo gasto na execução do modelo e da calibração, é utilizar o paradigma de computação paralela.

3.1 Método Genético

As simulações computacionais de sistemas genéticos começaram a ser desenvolvidas por John Holland, que iniciou gradualmente as pesquisas no tema e teve seu ápice em 1975 com a publicação do livro *Adaptation in Natural and Artificial Systems* [Holland, 1992].

O processo de seleção natural utilizado com método de solução para um problema matemático deve obrigatoriamente gravar as características do indivíduo nos seus genes. Deste modo, a próxima geração guarda as características da geração anterior, sendo que os indivíduos mais aptos terão maiores chances de se perpetuar, e quaisquer indivíduos menos aptos terão

a mesma probabilidade de sofrer mutação. Sendo assim, cada indivíduo será uma solução da função objetivo caracterizando sua aptidão [SHMITT, 2001].

No algoritmo genético o indivíduo pode ser codificado de várias formas, mais tradicionalmente são utilizados vetores binários onde cada elemento do vetor pode ser preenchido de 2 maneiras: 1 ou 0. O indivíduo também pode ser codificado de maneira real, onde cada gene é uma incógnita ou uma parte da resposta da função objetivo. Nestes casos a implementação ocorre de forma específica restringindo a aplicação ao problema em discussão [SHMITT, 2001].

Avaliar um indivíduo significa determinar o seu nível de aptidão de sobrevivência, ou seja, sobrevivem prioritariamente os indivíduos mais aptos. Em problemas de otimização, o critério de sobrevivência do indivíduo é determinado pelo valor da função objetivo [SHMITT, 2004].

São definidos originalmente dois operadores, o operador responsável pelo cruzamento e o operador responsável pela mutação. O operador cruzamento consiste em gerar um ou dois cromossomos filhos a partir das informações dos dois cromossomos pais, como ilustra a Figura 3.1. O operador mutação, ilustrado na Figura 3.2, altera um indivíduo aleatoriamente em um ou mais genes. Quanto maior o número de mutações em uma mesma operação menos informações do gerador são mantidas.

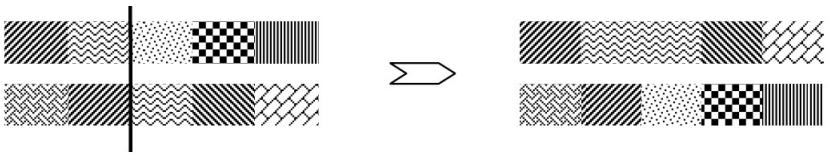

Figura 3.1: Operador cruzamento

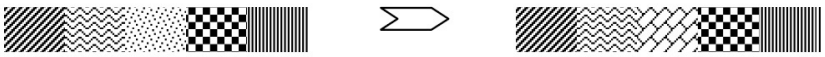

Figura 3.2: Operador mutação

O algoritmo Genético pode trabalhar com duas abordagens de seleção de indivíduo. O primeiro é a roleta, que confere a cada indivíduo um peso relativo à qualidade da solução que ele representa. A segunda abordagem é o torneio onde o primeiro indivíduo é selecionado comparando-se a aptidão de um par de indivíduos e o segundo indivíduo é selecionado de outro par de indivíduos.

O algoritmo Genético, implementado para Calibrar modelos que utilizam o TerraME para simulação utilizam a abordagem do torneio [Larissa M. Fraga, 2009]. Esse algoritmo funciona da seguinte maneira: o modelador deve fazer uma chamada a função como apresentado abaixo:

CalibracaoAG(boolEstocastico,mModelo, fAvaliacao, boolFlag,precisao, nTamPop, matrizLimites, pc, pm, nMaxGeracoes). Passando como parâmetro as configurações desejadas, que atendam na resolução do seu problema, os quais a tabela 3.2 informa.

O algoritmo começa com a inicialização aleatória de uma população com *nTamPop* indivíduos, que codificam configurações candidatas para o problema. O número de *bits* para cada variável depende da precisão desejada para aquela variável, um parâmetro que é fornecido pelo usuário. Caso o usuário não forneça uma precisão, o valor default é igual a 10^{-3} vezes a faixa de variação da variável (limite superior menos o limite inferior).

Cada geração t consiste na execução iterativa dos operadores genéticos que caracterizam o algoritmo. Na avaliação da população, linha 5, o modelo desenvolvido na linguagem LUA e indicado por *mModelo* é simulado usando os parâmetros de calibração codificados pelo genótipo do indivíduo. O resultado da simulação do modelo é então analisado para se estimar a qualidade do modelo, isto é, seu ajuste aos dados previamente fornecidos. A função de aptidão dos indivíduos é, portanto uma função da qualidade do modelo em modelar os dados do fenômeno sob análise.

Parâmetro	**Descrição**
boolEstocastico	Variável booleana que indica se o modelo a ser calibrado é estocástico ou não
mModelo	Contém o modelo a ser calibrado, descrito em linguagem LUA. O modelo recebe os parâmetros de calibração como argumento.
fAvaliacao	É a função de avaliação da qualidade do modelo, em geral o erro de ajuste a dados previamente fornecidos. Representa a função objetivo a ser minimizada ou maximizada pelo AG.
boolFlag	Variável que indica se a função objetivo deve ser minimizada ou maximizada.
precisao	Indica a precisão desejada para cada variável, internamente, reflete o número a ser usado na representação Binária do individuo do algoritmo genético (AG).
nTamPop	Tamanho da população.
matrizLimites	Matriz (nX2) contendo os limites máximos e mínimos para cada um dos parâmetros de calibração do modelo.
pc	Probabilidade de cruzamento.
pm	Probabilidade de mutação. A probabilidade de mutação usada é por indivíduo, isto é,indica a probabilidade de um indivíduo sofrer mutação ou não. Caso o indivíduo sofra mutação, um de seus bits é escolhido aleatoriamente e invertido.
nMaxGeracoes	Número máximo de gerações sem melhora, usado como critério de parada do algoritmo.

Tabela 3.2: Parâmetros do algoritmo Genético

Nas linhas 7 a 13, uma nova população de soluções candidatas é criada em função da população atual e de seus valores de aptidão. A etapa de seleção dos indivíduos *"pais"* para a reprodução é realizada de forma estocástica usando torneio binário. Nesta forma de seleção, dois indivíduos são selecionados aleatoriamente entre a população e competem entre si de forma

determinística, isto é, o melhor indivíduo entre os dois vence e é selecionado para a reprodução. Os melhores indivíduos possuem, portanto maior probabilidade de serem selecionados para a reprodução. Dois pais $Pt^{(i)}$,$Pt^{(j)}$ são escolhidos dessa forma,ver linha 8.

Entrada: boolEstocastico, mModelo, fAvaliacao, boolFlag,precisao, nTamPop, matrizLimites, pc, pm, nMaxGeracoes.

Saída: Valores estimados dos parâmetros ótimos de calibração.

```
1  início
2  t ← 0 {contador de gerações};
3  P_t = {P_t^(1), ..., P_t^(μ)} ← Inicializa População (nTamPop);
4  enquanto t < nMaxGeracoes faça
5      F_t ← Avaliar População (P_t, mModelo, fAvaliacao);
6      t*, p* ← Armazenar Melhor Indivíduo(P_t, F_t);
7      repita
8          P_t^(i), P_t^(j) ← Selecionar pais por torneio(P_t, F_t);
9          q_t^(1), q_t^(2) ← Aplicar cruzamneto (pc, p_t^(i), p_t^(j));
10         q_t^(1), q_t^(2) ← Aplicar mutação (pm, q_t^(1), q_t^(2));
11         Efetuar Competição (p_t^(i), p_t^(j), q_t^(1), q_t^(2));
12         Atuzalizar (P_t+1);
13     até Até que P_t+1 esteja completa;
14     t ← t + 1;
15 fim
16 fim
```

Figura 3.3: Pseudocódigo do algoritmo Genético

Nas linhas 9 e 10, duas novas soluções são produzidas a partir dos operadores de cruzamento e mutação. Esses operadores são constituídos por heurísticas específicas e em geral com algum componente de aleatoriedade. O módulo de calibração conta atualmente com operador de cruzamento com dois pontos de corte por variável e mutação por inversão de bit.

Finalmente, na linha 11, as duas novas soluções geradas (*indivíduos "filhos"*) competem de forma determinística com seus pais, isto é, os descendentes substituem os pais caso sejam melhores do que eles. A substituição pode ser vista como um operador de sobrevivência. Os indivíduos escolhidos na competição são armazenados em Pt+1 até que a nova população esteja completa com $nTamPop$ indivíduos.

Quando o modelo a ser calibrado é estocástico, o resultado de sua simulação varia para um mesmo conjunto de valores para os parâmetros de Calibração. Dessa forma, a função objetivo do problema de calibração torna-se uma função com ruído.

Abordagem utilizada na calibração de modelos estocásticos leva em conta a variância estimada do modelo para definir um limiar (*threshold*) na competição entre os pais e seus descendentes, conforme linha 11. Essa escolha torna o AG implementado bastante versátil

caso o modelo a ser calibrado seja estocástico ou não. Se o modelo for determinístico, o limiar de comparação para a substituição é zero, caso contrário, o limiar é definido em função da variância do modelo.

A variância do modelo é estimada de forma automática pelo módulo de calibração, executando-se o modelo inúmeras vezes para uma combinação fixa dos parâmetros e calculando o desvio padrão dos resultados dessas execuções. Esse procedimento é repetido para algumas combinações fixas dos parâmetros, fazendo uma média dos valores estimados para o desvio padrão. O resultado encontrado é utilizado como um limiar na competição dos filhos em relação aos pais.

Após o cruzamento, um filho substitui o pai se seu valor de aptidão for melhor que o dele em um valor superior a esse limiar, caso contrário, o pai permanece na população. Essa abordagem é interessante porque assegura de certa forma que indivíduos melhores permaneçam na população, pois, apesar da execução de um modelo retornar valores diferentes para um mesmo valor de parâmetro, flutuações próximas ao desvio padrão da função não são considerados.

3.2 Método de calibração Monte Carlo

O Método de Monte Carlo utiliza métodos de amostragem para resolver problemas de natureza estocástica ou determinística. Monte Carlo é um método apropriado para resolver problemas de alta dimensão e/ou parâmetros estocástico e, muitas vezes, é usado para calcular o valor esperado de uma variável que é função de várias variáveis estocásticas e que não pode ser tratada analiticamente.

O uso do Método Monte Carlo na calibração de modelos é "tentar imitar o comportamento da realidade". Quando um modelo é criado existem certo número de parâmetros de entrada e algumas equações que utilizam esses valores para dar resultados (respostas). Este é denominado determinista, ou seja, se você entrar com determinado valor, o resultado será sempre o mesmo, não importando quantas vezes executá-los.

Monte Carlo avalia iterativamente um modelo determinístico, utilizando conjuntos de números aleatórios como entradas. Dessa forma, ele é muito utilizado em modelos complexos, não lineares, ou que envolvem mais do que um casal de parâmetros incertos.

O método de Monte Carlo é apenas um dos muitos métodos para análise de propagação de incerteza, onde o objetivo é determinar como a variação aleatória, a falta de conhecimento, ou de erro afeta a sensibilidade, o desempenho ou a confiabilidade do sistema que está sendo modelado.

Monte Carlo é categorizado como um método de amostragem, pois as entradas são geradas aleatoriamente a partir de distribuições de probabilidade para simular o processo de amostragem de uma população real. Escolhemos então, uma distribuição para os insumos que mais se aproximam de dados que já temos, ou melhor, representa o nosso estado atual

de conhecimentos. Os dados gerados a partir da simulação podem ser representados como distribuições de probabilidade (ou histogramas) ou convertidos em barras de erro, previsões de confiabilidade, zonas de tolerância, e intervalo de confiança. A execução do Método de Calibração de Monte Carlo pode ser explicada nos seguintes passos, Figura 3.4:

- **Passo 1:** Gerar um conjunto de entradas aleatórias, $y = f(x_1, x_2, ..., x_q)$
- **Passo 2:** Execute o modelo e avalie as respostas, $x_{i1}, x_{i2}, ..., x_{iq}$
- **Passo 3:** Repita os passos 1 e 2 para i = 1 até n
- **Passo 4:** Analise os resultados usando histogramas, estatísticas de resumo, intervalos de confiança e etc.

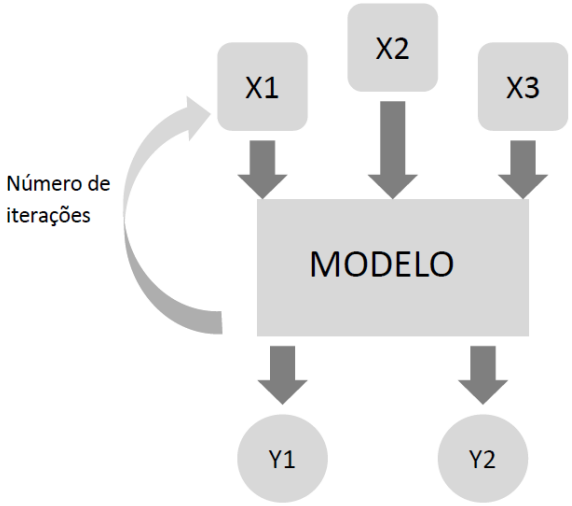

Figura 3.4: Passos de execução Monte Carlo

Um dos principais motivos para realizarmos este trabalho de paralelização do Algoritmo de calibração Monte Carlo, é o fato de geralmente a calibração de modelos envolve mais que 10000 avaliações do modelo, uma tarefa que ha tempos só era possível por super computadores, e que ainda hoje é muito custosa computacionalmente [Larsson, 2000] [Benjamin Block, 2001] [P. Hadjidoukas a, 2003].

3.3 Estratégia de paralelização dos algoritmos de calibração no TerraME

Vamos apresentar nesta subseção a estratégia de paralelização adotada. Para construir a solução, utilizamos a técnica de decomposição dos dados, dividimos os dados de entrada para

os algoritmos de calibração em pequenos conjuntos para que fosse possível executar o método de calibração simultâneamente em cada um desses subconjuntos de dados.

Como técnica de mapeamento implementamos a técnica de *bag of tasks* [Ananth Grama, 2003]. Atribuímos uma *Task*(Tarefa) de calibração para cada core da máquina onde a calibração é executada. O método de calibração sequencial executa com $nMax$ parâmetros. Na versão paralela cada core da máquina é responsável por $\frac{nMax}{cores}$ parâmetros da calibração, onde *cores* é a quantidade de núcleos de processamento da máquina de execução. A calibração de um modelo com um determinado subconjunto de parâmetros de entrada, é encapsulado em uma tarefa (*Task*) de calibração única que não interage com as demais. De forma geral essa característica não gera nenhuma sobrecarga de comunicação.

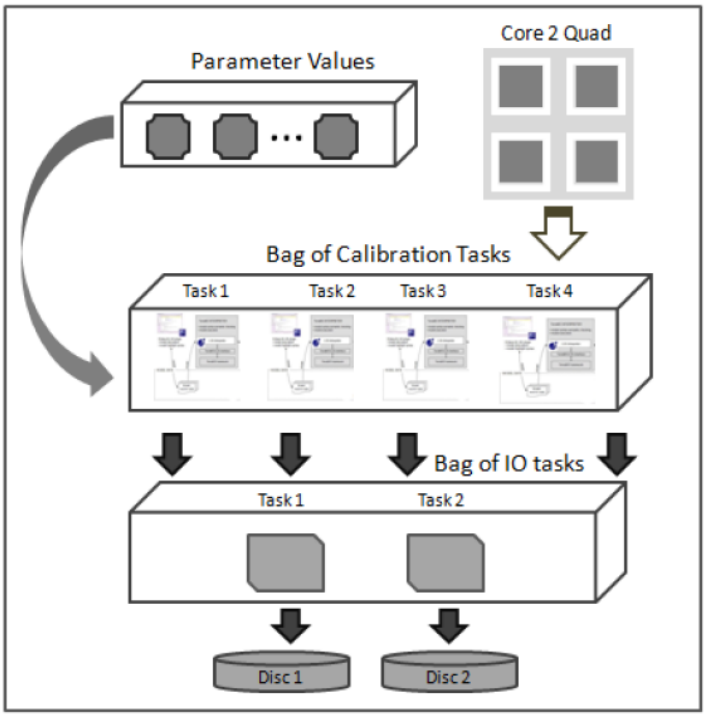

Figura 3.5: Exemplo de uma calibração paralela no TerraME

Na figura 3.5 a máquina tem uma CPU quad-core. Cada *task* de calibração recebe um subconjunto de parâmetros para avaliar e executar a calibração. Ao término da calibração a *Task* consulta o Bag para pegar mais um subconjunto de parâmetros para calibrar.A calibração paralela termina quando não existem mais subconjuntos a serem avaliados. Em vez de um número fixo de subconjuntos para calibração no *Bag*, esse número pode variar dependendo das características de simulação.

O *I/O* é um grande problema em calibração paralela. Resultados da calibração precisam ser armazenados de forma não volátil a cada simulação, ou de uma única vez ao final de

toda a simulação. Como simulações lidam com uma grande quantidade de dados, a segunda alternativa pode ser inviável. A primeira alternativa permite que várias operações de *I/O* possam ser executas ao longo da simulação. Seguindo esta alternativa, para que fosse possível prover um *I/O* paralelo, implementamos um *Bag I/O*, assim no ato da escrita do *log file* de calibração, o *I/O* não influencia com tanta intensidade no speedup da aplicação.

Quando calibramos um modelo utilizando o método de calibração genético, os valores dos parâmetros produzidos em cada simulação são utilizados para avaliar as simulações seguintes. Basicamente podemos implementar duas alternativas: (i) cada *Task* (tarefa) de calibração pode evoluir independente, ou seja, sem o uso dos resultados das *Task's* que estão executando em paralelo, (ii) a *task* (tarefa) de calibração pode usar o resultado das *task's* que já foram executadas e das que estão sendo executadas no momento. Neste trabalho nos contemos em implementar apenas a primeira abordagem, já que nosso objetivo é construir uma solução escalável e não uma que prove uma acurácia maior.

Para que o usuário use o módulo de calibração paralela, adotamos a técnica de instrumentação. Com esta técnica o usuário não recodifica seu código original. O usuário apenas insere anotações na forma de comentários ao código existente. Esta técnica fornece portabilidade ao TerraME pois podemos executar um modelo com anotações tanto na versão TerraME onde o módulo de calibração é paralelo e na versão com módulo de calibração sequencial. A Figura 3.6, ilustra tanto as anotações necessárias para utilizar o módulo de calibração genético e de Monte Carlo em paralelo. A instrumentação é feita pelo comentário *@ParallelCalibration*, a ser inserido antes do método de calibração.

```
function Model(x, x1, ..., xn)
        MODEL IMPLEMENTATION
end

paramRange  = {100, 1000};  -- range of parameter values
precision –
maxInteractions --


--@ParallelCalibration
GeneticCalibration (model, paramRange, precision, maxInteraction , ...);

--@ ParallelCalibration
MonteCarloCalibration(Model, lowerBound, upperBound, nMax);
```

Figura 3.6: Instrumentação necessária para utilizar a calibração paralela dos métodos de Monte Carlo e Genético

Para identificar quais pontos do código o usuário instrumentou o código a fim de utilizar do módulo de calibração paralela, implementamos um interpretador, que recebe como entrada o código do usuário e efetua uma varredura procurando pelas instrumentações que o usuário inseriu. Após identificar o local da instrumentação ele converte a instrumentação para a devida

chamada do método de calibração paralelo. Ao completar a varredura o interpretador repassa o novo modelo("traduzido") para a plataforma TerraME que é encarregada de executar todas as chamadas de funções. Na Figura 3.7 ilustramos a arquitetura do módulo de calibração paralela.

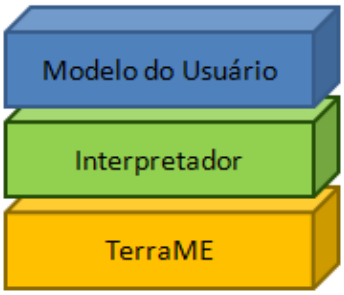

Figura 3.7: Interpretador para o módulo de calibração paralela

Capítulo 4

Simulação paralela no TerraME

No capítulo anterior apresentamos um módulo de calibração paralela para modelos construídos na plataforma TerraME. Infelizmente a solução descrita no capítulo anterior possui severas limitações, pois assume o paralelismo apenas no processo de calibração de modelos e não na simulação de um modelo. Note que ao permitirmos simulações paralelas permitimos também calibrações paralelas, porém o inverso não é verdadeiro.

Diante do cenário exposto acima, apresentamos neste capítulo a solução paralela TerraME para simulação de modelos espacialmente explícitos, mantendo a portabilidade dos modelos já escritos de forma sequencial.

No intuito de tornar a modelagem o mais simples possível, o módulo TerraME HPA fornece um conjunto de diretivas simples e intuitivas que, são encarregadas de efetuar a transparência de toda a arquitetura TerraME HPA. As diretivas de programação paralela oferecidas atualmente pelo módulo TerraME HPA são:

- HPA PARALLEL, indica que a função logo abaixo, será enviada para a *Bag of Task's* e consequentemente executada por algum dos *Worker's* de forma paralela. Os conceitos *worker* e *bag of tasks* são explicados mais a frente neste capítulo.

- HPA JOIN Fx, criando uma analogia a função *JOIN* da linguagem JAVA, esta diretiva aguarda até que todas as chamadas da função *Fx* que foram feitas até o momento sejam executadas. Ou seja, até que todas chamadas desta função sejam retiradas da *Bag of Task's* e até que todos os *Worker's* que estejam executando a função *Fx* terminem sua execução.

- HPA JOINALL, diretiva semelhante a especificada acima, no entanto *JOINALL*, aguarda até que a *Bag of Task's* esteja totalmente vazia e todos os *Worker's* estejam em estado de *Sleeping*.

- HPA LOCK Gl, diretiva responsável pela sincronização de *thread's*, ela gerência o acesso a um recurso global(Gl). Com *HPA LOCK* restringimos o acesso a variável global Gl para todas as *thread's* que tentarem acessar este recurso (Gl).
- HPA UNLOCK Gl, indica que a variável global Gl, esta disponível para ser acessada por outras *thread's*. Pois a thread que detinha o direito da variável não necessita dela mais.

Para oferecer as diretivas de programação paralela ao usuário TerraME, um interpretador foi implementado. O interpretador recebe o modelo do usuário como entrada e efetua a interpretação das diretivas que compõem o TerraME HPA para controle de concorrência.

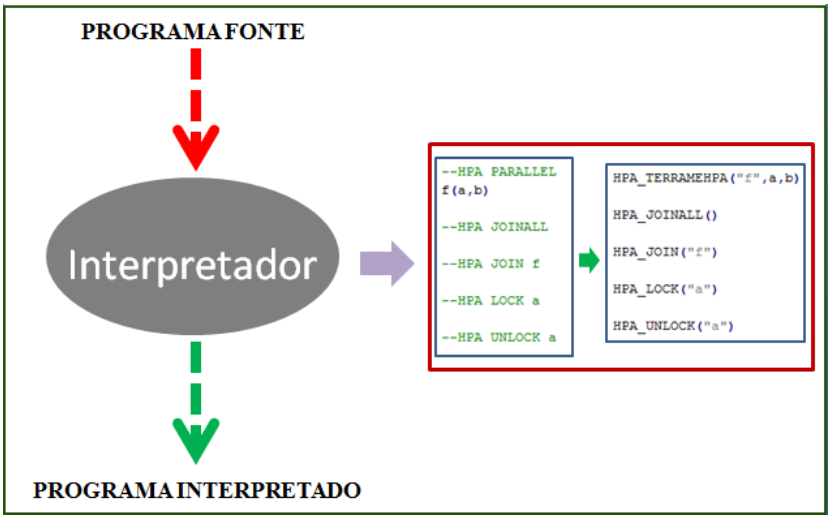

Figura 4.1: Interpretador do módulo TerraME HPA

Inicialmente, o interpretador comunica com o kernel TerraME para que a pilha de execução do modelo seja criada, assim como um conjunto de threads chamadas workers que são responsáveis por executar funções em paralelo de um determinado modelo espacial. A quantidade de *workers* é normalmente igual ao número de núcleos de processamento da máquina.

A pilha de execução do modelo possui todas as variáveis globais, assim como a primeira chamada que inicializa o modelo. Tal pilha é passada como referência para cada worker, permitindo o acesso concorrente ao recurso, no caso a pilha de execução.

Cada instrução do modelo passa pelo interpretador TerraME HPA, portanto ao detectar uma primitiva de paralelismo (–HPA Parallel) o *kernel* é notificado para que a função requerida seja executada em paralelo. Neste momento um worker assume a execução da função especificada pelo usuário.

Por questões de eficiência, adotamos um *bag of tasks* compartilhado entre os *workers*. A cada função instrumentada para ser executada em paralelo, o interpretador adiciona o nome da função e seus parâmetros ao *bag of tasks*, o que gera a notificação de algum *worker*. O

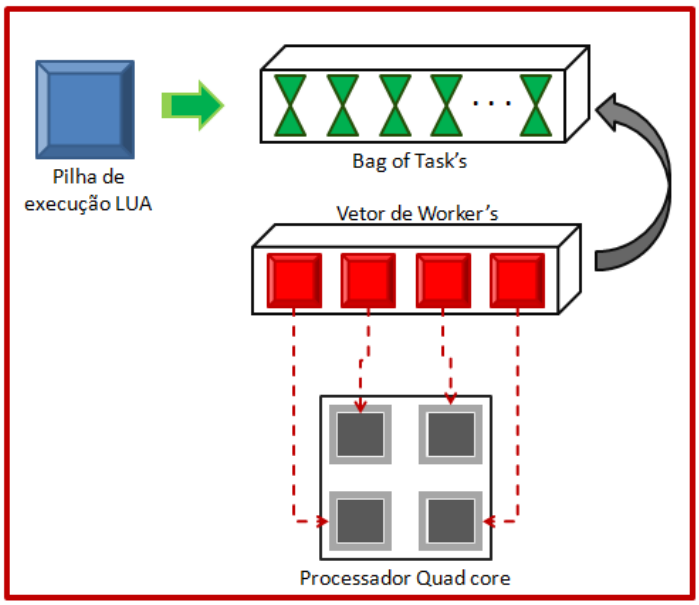

Figura 4.2: Arquitetura do módulo TerraME HPA

worker notificado assume a execução paralela da função definida pelo usuário e registra o retorno desta função na mesma pilha de execução.

A execução do modelo termina quando a última instrução do modelo do usuário passa pelo interpretador e é executada pelo *kernel* TerraME. Na Figura 4.2 ilustramos o cenário com quatro *workers* numa arquitetura de computador quad-core.

Cada *Worker* foi implementado como uma *QThread*, uma vez que o suporte que a linguagem LUA oferece a multi tarefas é o conceito de co-routine. Co-routines são chamadas também de *multithreading* colaborativa, pois possuem sua própria pilha de execução, suas variáveis locais, e seu ponteiro de instrução própria, mas compartilham as variáveis globais. A principal diferença entre co-rotinas e threads é que conceitualmente, um programa com threads executa vários segmentos simultaneamente. Co-rotinas, por outro lado, são colaborativas, ou seja um programa que utilize co-rotinas executa apenas uma a cada momento, elas estão dentro de um mesmo processo, não atendendo os requisitos do módulo TerraME HPA, pois este precisa executar segmentos de forma paralela (simultânea).

Os *Worker's*, possuem acesso a pilha de execução global, possuem uma pilha de execução local da função sendo processada e tem acesso a uma *Bag of Task's*, comum a todos *worker's*. A *Bag of Task's* contêm o nome de cada função que deve ser executada de forma paralela assim como seus respectivos parâmetros para execução,como ilustra a Figura 4.3. Cada *Worker* executa enquanto existirem elementos na *Bag of Task's*. Ao final da execução de uma função, o *Worker* consulta a *Bag* para verificar se ainda existem funções para serem executadas, (i) caso exista ele retira da *Bag of Task's* a próxima função a ser executa, (ii)caso não exista ele entra em um estado denominado *sleeping*, e pode ser reiniciado a qualquer momento.

Eventualmente eles são re-iniciados na ocorrência de uma nova inserção no *Bag of Task's* de uma nova chamada de função a ser executada.

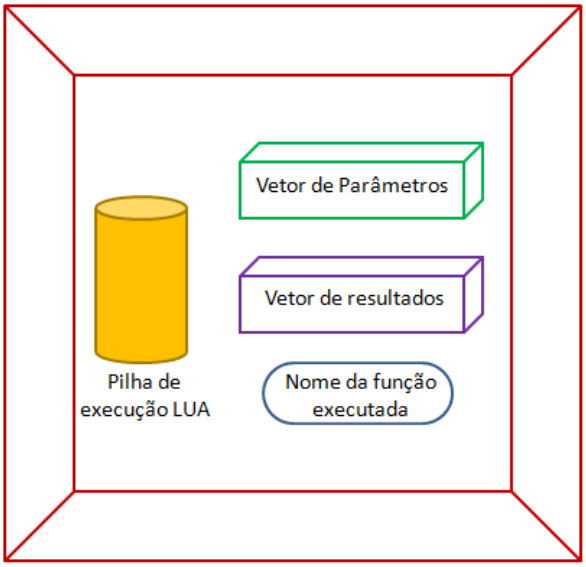

Figura 4.3: Estrutura de um Worker

Uma vez explicado o funcionamento geral do interpretador e *kernel* TerraME HPA, passamos a explicação do funcionamento das diretivas de paralelização do TerraME em modelos reais.

O princípio básico por trás do TerraME HPA é permitir que funções de modelos TerraME possam ser executadas de forma assíncrona, garantindo ao usuário final um ganho de desempenho quando tais funções possuem baixa dependabilidade.

Para que possamos entender melhor o exposto acima, vejamos um exemplo.

A primeira diretiva encontrada pelo interpretador TerraME é *–HPA PARALLEL*, porém esta diretiva é chamada cem mil vezes. Nesta porção de código o usuário almeja nível máximo de concorrência na execução da função *ModelUser*.

Internamente o TerraME não instancia cem mil *workers* para executar as funções. Conforme explicado anteriormente, as cem mil funções são inseridas num *bag of tasks* e um número n de *workers* inicia seu consumo, onde n normalmente é proporcional ao número de núcleos de processamento da maquina.

Ao final do programa principal o usuário aguarda o término das cem mil execuções da função *ModelUser* usando a diretiva *–HPA JOINALL*. Notem que no modelo temo a variável global *VetofPar1* e a implementação da função *ModelUser*. A função *ModelUser* realiza escrita na variável global, portanto para que este procedimento seja feito de forma correta o usuário deve bloquear a variável para seu uso exclusivo e desbloquear ao término da utilização (*HPA LOCK* e *UNLOCK*, respectivamente).

```
VetOfPar1 = {}

function ModelUser(par1,par2)

    --HPA LOCK VetOfPar1
     table.insert(VetOfPar1,par1);
    --HPA UNLOCK VetOfPar1

    --implementação do modelo
end

for i = 1, 100000 do
     --HPA PARALLEL
     ModelUser(i,100000-1)
end
--HPA JOINALL
```

Figura 4.4: Exemplo da instrumentação de um código, para este execute em paralelo

Já estamos trabalhando para reduzir ainda mais o uso das diretivas de paralelismo do TerraME. Acreditamos não ser necessário usar explicitamente *LOCK* e *UNLOCK* no exemplo anterior, pois a operação de escrita numa variável global é sempre *thread safe*. Por outro lado, ainda acreditamos ser útil continuarmos com as primitivas *LOCK* e *UNLOCK*, pois o usuário pode almejar bloquear uma variável e fazer inúmeras operações de leitura, lógica, aritmética e escrita antes de desbloquear tal variável.

Capítulo 5

Experimentos

Para o estudo de casos, realizamos testes medindo o tempo gasto para a execução, e para conhecer o comportamento que cada solução traz em diferentes cenários, variamos o número de núcleos de processamento disponíveis da máquina utilizada para os testes.

As execuções dos testes foram realizados em uma máquina com *Sistema Operacional Windows Server 2003, 16 GB RAM, Intel(R) Xeon(R), clock 2.00 GHz, com 4 discos 7200rps e 32MB buffer.*

5.1 Experimentos para calibração paralela

Para o teste da calibração paralela de Monte Carlo, testamos com 10000, 15000, 2000, 25000, 30000 e 60000 simulações. O método Genético foi testado com 500, 1000, 1500, 2000, 2500 e 5000 simulações.

5.1.1 Testes para o Método de Monte Carlo

Abaixo o gráfico referente aos tempos finais da execução dos testes, utilizando a abordagem paralela de Monte Carlo.

A figura 5.1, ilustra o speedup do método de calibração de Monte Carlo. Em geral, o algoritmo de Monte Carlo utilizando duas *thread's*, obtemos uma redução no tempo de execução em torno de 40%. O *speedup* neste caso foi de 80% do linear. Ao aumentar o número de *thread's* observamos uma perda no *speedup*. Com quatro *thread's* e quatro discos o *speedup* foi de 77% do linear e com oito *thread's* e quatro discos o *speedup* foi de 50% do linear. Conforme o número de iterações do algoritmo aumenta o *speedup* apresenta um pequeno aumento, é o que podemos observar no teste com 60000 simulações. Inicialmente o *speedup* executando 60000 simulações e com duas *thread's* é semelhante aos demais cenários, mas podemos constatar que o *speedup* ao se utilizar quatro *thread's* e com oito *thread's* apresenta um pequeno aumento.

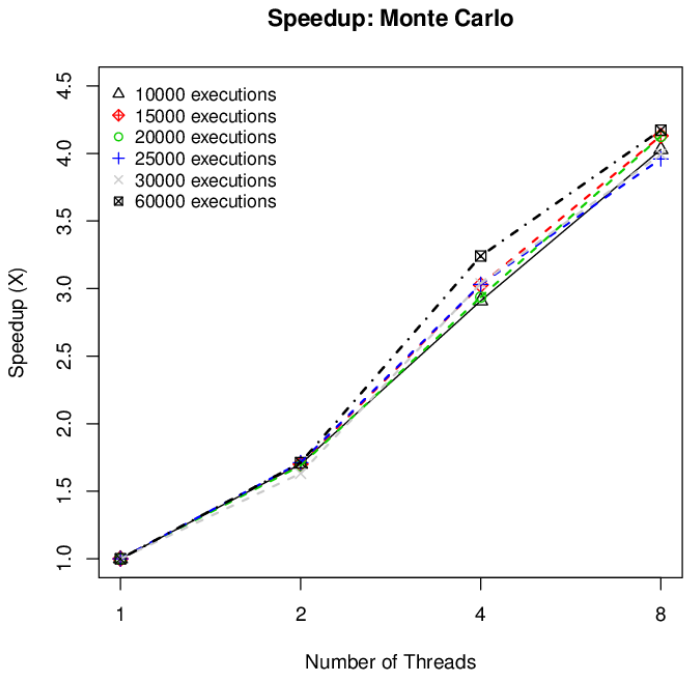

Figura 5.1: Speedup para o teste de Monte Carlo

5.1.2 Testes para o Método Genético

Os resultados para a calibração Genética 5.2 são semelhantes aos do método de Monte Carlo, ou seja, com duas *thread's* o *speedup* foi de 80% do *speedup* linear, com quatro *thread's* em torno de 75% do *speedup* linear e com oito *thread's* cerca de 50% do *speedup* linear. Em geral, os resultados de ambos, genética e método de Monte Carlo são esperadas, pois o número de discos é menor do que o número de núcleos de processamento. A deterioração do *speedup* a partir de duas *thread's* a quatro *thread's* é maior do que a perda do *speedup* a partir de quatro *thread's* a oito *threads*.

O *I/O* tornou-se o gargalo e assumimos que é a principal razão de apenas 50% do *speedup* linear com oito *thread's* em uma máquina com oito núcleos de processamento. Cada calibração simulação dos algoritmos de calibração, armazena os resultados no disco, mas em nossos experimentos, temos apenas quatro discos, para que possamos ter duas tarefas *I/O* concorrentes a escrever em um único disco. Testamos o método de Monte Carlo e o método de calibração genética com dez *thread's*, mas os resultados, como esperávamos, tem *speedup* bem inferior comparando se com o cenário onde utilizamos oito *threads*. A razão para isso deve-se ao fato que ao utilizar dez *thread's* ocorre uma grande quantidade de troca de contexto entre as *thread's* uma vez que a máquina utilizada para testes possui apenas oito cores de processamento.

É importante ressaltar que ao final de todos os testes, comparamos o resultado da calibração paralela com a sequencial e os resultados foram os mesmos.

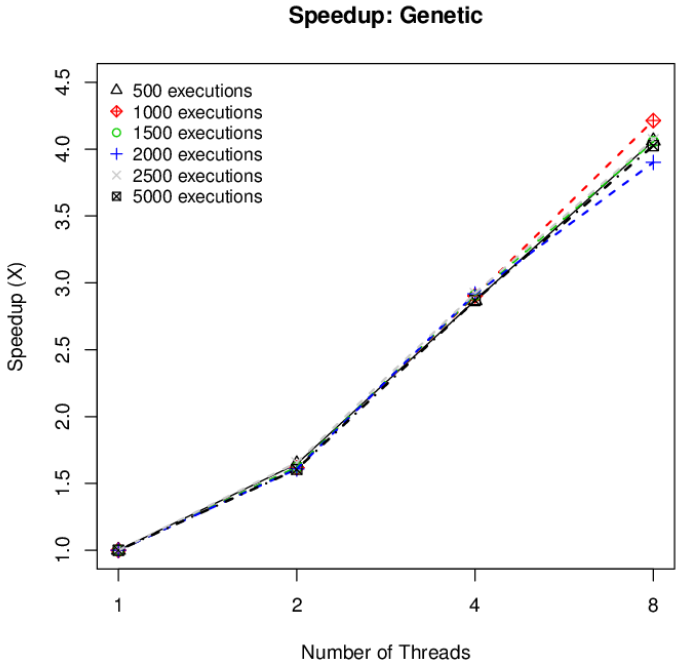

Figura 5.2: Speedup para o teste Genético

5.2 Experimentos para simulação paralela

Para efetuar os testes do módulo TerraME HPA, utilizamos duas abordagens, (i) aplicamos o módulo TerraME HPA a um modelo altamente paralelizável (contador de palavras paralelo),ele nos auxilia a enxergar como a carga de trabalho esta sendo dividida entre as thread's, (ii) A fim de verificar a eficiência do módulo TerraME HPA em modelos reais utilizamos o modelo THOLL [Chave, 1999], para tal fim. Logo abaixo apresentamos o gráfico do *Speedup* da solução e discutimos a respeito dos resultados alcançados.

É possível observar pelos resultados que ilustramos na Figura 5.3 que, para o Modelo *Teste* o *Speedup* da solução chegou a ser linear com a utilização de duas *thread's*, para quatro e oito atingimos um Speedup de aproximadamente 95% do linear. Neste modelo não temos nenhuma dependência entre os dados, dessa forma não é complicado atingir tal resultado. Já para o modelo THROLL podemos observar um *Speedup* abaixo da solução sequencial, uma das razões deste resultado é a alta dependabilidade cronológica que cada função do THROLL possui. Dessa forma a paralelização foi obtida, em uma escala bem fina, ou seja executando funções que não tem ordem cronológica entre si e de baixa complexidade computacional que são executas de forma exaustiva, estes dois últimos motivos nos levam a um segundo problema que é explicado a seguir.

O módulo TerraME HPA trabalha com o suporte de acesso as variáveis globais, e esse suporte ocasiona na seguinte limitação, as variáveis globais estão contidas na pilha de execução

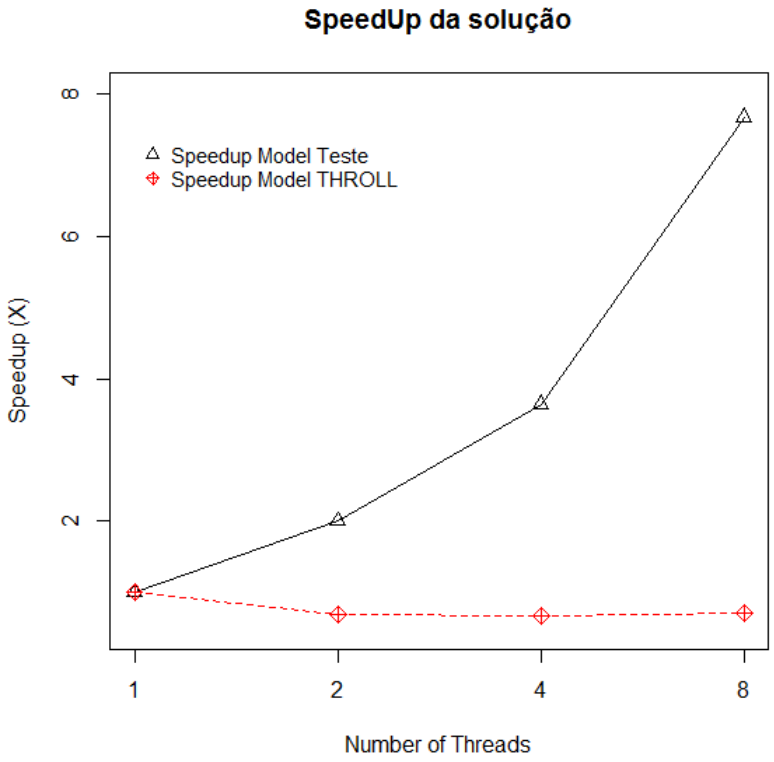

Figura 5.3: Speedup para os testes utilizando o módulo TerraME HPA

principal (pilha *MAIN*) do LUA , logo todas as pilhas de execução das *thread's* implementadas em *C++* devem referenciar esta pilha principal, com isso temos a limitação que só é permitido finalizar e iniciar uma pilha de execução filha de cada vez, é necessário então, sincronizar a finalização e o inicio de cada *thread*. Dessa maneira, para o modelo THROLL em que conseguimos paralelizar apenas funções de baixa complexidade computacional(são executadas de forma muita rápida) e que são executadas repetidas vezes, nos leva a um número muito grande de sincronizações ocorrendo a toda hora, ocasionando em uma enorme perda de *Speedup* da solução paralela, deixando a pior que a solução sequencial já implementada.

Estamos trabalhando no encapsulamento de funções pouco custosas do modelo Throll em funções com maior granularidade, com o intuito de melhorar o *speedup* do modelo. De uma forma geral, almejávamos comprovar que se modelar bem o problema atingimos níveis de concorrência aceitáveis para arquiteturas de computadores com memória compartilhada. Caso contrário, o cenário pode ficar inclusive pior do que a versão sequencial do modelo proposto, como observamos nos resultados apresentados na Figura 5.3.

Para validar os resultados obtidos ao se utilizar o módulo TerraME HPA, escolhemos um cenário onde é necessário um número alto de sincronização para atualização das variáveis compartilhadas (globais). Ao final de todos os testes os resultados esperados foram obtidos,

indicando que a utilização do módulo TerraME HPA para execução paralela de modelos, ao utilizar as diretivas de *lock* e *unlock* para sincronização das variáveis globais, não altera no resultado final da simulação.

Capítulo 6

Conclusões

Com a calibração de modelos de forma paralela, conseguimos realizar a primeira extensão da plataforma de modelagem TerraME. A estratégia de paralelização é simples, eficiente e representou os estudos iniciais de um plano mais ambicioso que é o módulo TerraME HPA. Em geral, conseguimos 75-80% do *speedup* linear no processo de calibração de estudo de caso real, quando o número de núcleos é idêntico ao número de discos. A aptidão do modelo produzido por métodos de calibração sequencial e paralela são semelhantes. Nós não investigamos os benefícios do paralelismo na acurácia da calibração.

Para o módulo TerraME HPA,observamos que as diretivas oferecidas pelo mesmo, são intuitivas e não necessitam da alteração do modelo sequencial original do usuário. Os testes de desempenho nos mostram que para modelos que apresentam funções de alta complexidade computacional e que podem ser paralelizadas o módulo TerraME provê um suporte para que os resultados finais sejam mantidos e a execução paralela seja totalmente transparente para o usuário e o mais eficiente o possível. Já para modelos como o THROLL que apresentam grande dependabilidade e necessita de uma ordem cronológica entre a execução de suas funções fica a cargo do usuário encontrar funções com uma granularidade mais alta para que estas possam ser executadas de forma paralela, minimizando assim a sincronização entre as *thread's*.

Neste trabalho, desenvolvemos apenas uma versão de memória compartilhada para a plataforma TerraME. Acreditamos que o speedup da versão distribuída pode estar mais perto linear do que a versão paralela. Este fenômeno é normalmente observado em sistemas distribuídos em relação paralela. Note que, sempre precisamos otimizar as comunicações em sistemas distribuídos para que nossa solução atinja este requisito. Estamos investigando também a utilização de GPU's para simulações de modelos em paralelo.

6.1 Trabalhos futuros

Como trabalhos futuros pretendemos, adicionar ao módulo TerraME HPA a capacidade de execução distribuída. Distribuindo a execução do TerraME, adicionaremos mais poder com-

putacional, pois poderemos executar simulações de modelos em um *grid* de computadores,e aumentaremos com isso o espaço de memória disponível para simulação. Para que isso seja possível, novas diretivas devem ser criadas, assim como vários problemas devem ser tratados como a divisão dos dados entre os *nós* do *grid* e questões como tempo de informações trafegando na rede, sincronização entre processos rodando de forma distribuída,começam a ter que ser levadas em consideração.

Referências Bibliográficas

[Adve S.V., 2008] Adve S.V., e. (2008). Parallel computing research at illinois: The upcrc agenda. *Illinois*.

[Almasi e Gottlieb, 1989] Almasi, G. e Gottlieb, A. (1989). Highly parallel computing.

[Ananth Grama, 2003] Ananth Grama, e. (2003). *Introduction to Parallel Computing (2nd Edition)*. Massachusetts,USA.

[Asanovic, 2006a] Asanovic, Krste, e. (2006a). The landscape of parallel computing research: A view from berkeley. *Technical Report UCB/EECS*.

[Asanovic, 2006b] Asanovic, K. e. (2006b). The landscape of parallel computing research: A view from berkeley. *Technical Report UCB/EECS*.

[Barney, 2007] Barney, B. (2007). Introduction to parallel computing. *Lawrence Livermore National Laboratory*.

[Batistella, 2005] Batistella, M. e Moran, E. F. (2005). Dimensões humanas do uso e cobertura das terras na amazônia: uma contribuição do lba. *Acta Amazonica*.

[Benjamin Block, 2001] Benjamin Block, Peter Virnau, T. P. (2001). Multi-gpu accelerated multi-spin monte carlo simulations of the 2d ising model. *Department of Physics, Mathematics and Computer Science, Johannes Gutenberg University Mainz*.

[Carneiro e Câmara, 2007] Carneiro, T. e Câmara, G. (2007). A gentle introduction to terrame. *Technical report,Instituto Nacional de Pesquisa Espacial (INPE), São José dos Campos, Brasil*.

[Carneiro, 2006] Carneiro, T. G. (2006). Nested-ca: a foundation for multiscale modeling of land use and land change. phd thesis in computer science.

[Chave, 1999] Chave, J. (1999). Study of structural, sucessional and spatial patterns in tropical rain forest using troll, a spatially explicit forest model. *Ecological Modeling*, (124):233–254.

[Coppin, 2004] Coppin, Jonckheere, e. (2004). Digital change detection methods in ecosystem monitoring: a review. *International Journal of Remote Sensing.*

[Costanza e Voinov, 2001] Costanza, R. e Voinov, A. (2001). Modeling ecological and economic systems with stella. *Ecological Modelling.*, (143):1 – 7.

[Felgueiras, 1999] Felgueiras, C. A. (1999). Modelagem ambiental com tratamento de incertezas em sistemas de informação geográfica: O paradigma geoestatístico por indicação. *PhD thesis, INPE - Instituto Nacional de Pesquisas Espaciais.*

[Githeko, 2000] Githeko, A. K.; Lindsay, e. (2000). limate change and vector- borne diseases: a regional analysis. *Bulletin- World Health Organization.*

[Hales S., 2002] Hales S., e. (2002). Potential efect of population and climate changes on global distribution of dengue fever: an empirical model. *The Lancet.*

[Henzinger, 1996] Henzinger, T. A. (1996). The theory of hybrid automata. *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996).*

[Holland, 1992] Holland, J. H. (1992). An introductory analysis with applications to biology, control, and artificial intelligence. *University of Michigan, Ann Arbor.*

[Ierusalimschy, 2003] Ierusalimschy, R. (2003). Programming in lua. *ISBN 85-903798-1-7.*

[Kaimowitz e Angelsen, 1998] Kaimowitz, D. e Angelsen, A. (1998). Economic models of tropical deforestation: A review. technical report.

[Katiani da Conceição Loureiro, 2010] Katiani da Conceição Loureiro, Mirian Buss Gonçalves, e. (2010). Uso de contagens de tráfego e modelos econométricos na estimativa de uma matriz o-d de transporte interurbano de veículos leves. *XVI PANAM.*

[Lambin, 1994] Lambin, E. F. (1994). Modeling deforestation processes, a review. trees, tropical ecosystem environment observation by satellites.

[Lana, 2009] Lana, R. M. (2009). Modelos dinâmicos acoplados para simulação da ecologia do vetor aedes aegypti. *Master's thesis, Universidade Federal de Ouro Preto, Programa de Pós-Graduação em Ecologia de Biomas Tropicais, Ouro Preto, Brasil.*

[Larissa M. Fraga, 2009] Larissa M. Fraga, Raquel M. Lana, e. (2009). Calibração em modelagem ambiental na plataforma terrame usando algoritmos genéticos. *XLIISBPO.*

[Larsson, 2000] Larsson, J. (2000). Monte carlo implementation of financial simulation on cell/b.e. multi-core processor. *University of Reading, Whiteknights.*

[Lustosa P. R. B., 2004] Lustosa P. R. B., Ponte V. M. R., e. (2004). Pesquisa operacional para decisão em contabilidade e administração. *Contabilometria.*

[Martens, 1995] Martens, Niessen, e. (1995). Potential impact of global climate change on malaria risk. *Environ Health Perspective.*

[Minar, 1996] Minar, N., e. (1996). The swarm simulation system: A toolkit for building multi-agent simulation. *SFI Working Paper.*

[P. Hadjidoukas a, 2003] P. Hadjidoukas a, C. Bousis b, D. E.-g. (2003). Parallelization of a monte carlo particle transport simulation code. *Department of Computer Science and Department of Medical Physics , University of Ioannina.*

[Patterson e Hennessy, 1998] Patterson, D. A. e Hennessy, J. L. (1998). Computer organization and design, second edition. *Morgan Kaufmann Publishers.*

[Patz, 1998] Patz, J. A., e. (1998). Dengue fever epidemic potential as projected by general circulation models of global climate change. *Environ Health Perspective.*

[Salazar L. F. e D., 2007] Salazar L. F., N. C. A. e D., O. M. (2007). Climate change consequences on the biome distribution in tropical south america. *Geophysical Research Letters.*

[Saude, 2006] Saude, M. (2006). Abordagens espaciais na saude publica.

[SHMITT, 2001] SHMITT, L. (2001). Theory of genetic algorithms. handbock of estatistics. *The University of Aizu.*

[SHMITT, 2004] SHMITT, L. (2004). Theory of genetic algorithms ii: Models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling. *The University of Aizu.*

[Soares, 2003] Soares, B. S., e. (2003). Dinamica - a stochastic cellular automata model designed to simulate the landscape dynamics in an amazonian colonization frontier. *Ecological Modelling*, (154):217–235.

[Systems., 2004] Systems., V. V. (2004). Vensim.

[Tellinghuisen, 2010] Tellinghuisen, J. (2010). Least-squares analysis of data with uncertainty in x and y: A monte carlo methods comparison.

[Turner II, 1995] Turner II, Skole, e. (1995). Land-use and land-cover change science, research plan. *The International Geosphere-Biosphere Programme (IGBP) and The Human Dimensions of Global Environmental Change Programme (HDP).*

[Wolfram, 2003] Wolfram, S. (2003). New kind of science.

[Xiangrong Li, 1998] Xiangrong Li, e. (1998). Anisotropic mesh gradation control.