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 na rede de pesquisa GEOMA para a modelagem e simulação de processos de mudança de uso e cobertura do solo (LUCC) para toda região amazônica. Na FIOCRUZ vem sendo utilizada para o desenvolvimento de modelos epidemiológicos, como são os casos do controle da Dengue nas cidades do Rio de Janeiro e Recife. Além do TerraME, as plataformas como *Swarm*, *Stella*, *Vensim*, *DINAMICA-EGO* são alternativas de ambientes de modelagem. Nenhuma dessas plataformas foi desenhada para ser executada em arquitetura de computadores com memória compartilhada ou distribuída. Diante deste cenário, propomos o TerraME HPA (*TerraME 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 ao executar modelos de fraca dependabilidade utilizando o TerraME HPA é possível obter runtime linear e speedup de 7.62 ao utilizar 8 núcleo de processamento, o que indica 95% de paralelismo segundo *Amdhal*.

Abstract

TerraME is a modeling and simulation platform for the development of spatially explicit dynamic models that has been used on search for GEOMA for modeling and simulation of land use change and land cover (LUCC) for the entire Amazon region. In FIOCRUZ has been used for the development of epidemiological models, as are the cases of dengue control in the cities of Rio de Janeiro and Recife. Besides TerraME, platforms such as *Swarm*, *Stella*, *Vensim*, *DINAMICA-EGO* are alternative modeling environments. None of these platforms is designed to run on computer architecture with shared memory or distributed. In this scenario, we propose the HPA TerraME (*Archtecture TerraME High Performance*). 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 when you run low dependability models using TerraME HPA is possible to obtain linear runtime and speedup of 7.62 when using 8 core processing, which shows 95% of parallelism as *Amdhal* demonstrated.

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 8
2.5 TerraView 9
2.6 Computação Paralela 9

3 Calibração Paralela de Modelos no TerraME **11**
3.1 Método Genético 12
3.2 Método de calibração Monte Carlo 14
3.3 Estratégia de paralelização dos algoritmos de calibração no TerraME 16

4 Simulação paralela no TerraME **20**

5 Experimentos **25**
5.1 Experimentos para calibração paralela 25
5.1.1 Testes para o Método de Monte Carlo 25
5.1.2 Testes para o Método Genético 26
5.2 Experimentos para simulação paralela 27

6 Conclusões **30**
6.1 Trabalhos futuros 30

Referências Bibliográficas **32**

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 13
3.2 Operador mutação 13
3.3 Passos de execução Monte Carlo 15
3.4 Exemplo de uma calibração paralela no TerraME 17
3.5 Instrumentação necessária para utilizar a calibração paralela dos métodos de Monte Carlo e Genético 19
3.6 Interpretador para o módulo de calibração paralela 19

4.1 Interpretador do módulo TerraME HPA 21
4.2 Arquitetura do módulo TerraME HPA 22
4.3 Estrutura de um Worker 23
4.4 Exemplo da instrumentação de um código, para um contador de palavras paralelo . 24

5.1 Speedup para o teste de Monte Carlo 26
5.2 Speedup para o teste Genético 27
5.3 Speedup para os testes utilizando o módulo TerraME HPA 28

Lista de Tabelas

3.1 Plataformas vs. métodos de calibração 12
3.2 Parâmetros do algoritmo Genético 14
3.3 Tempo de execução para a calibração de modelos utilizando o algoritmo de Monte Carlo 17
3.4 Tempo de execução para a calibração de modelos utilizando o algoritmo de Monte Carlo, com melhoria de I/O 18

5.1 Tempo de execução para os testes realizados utilizando o módulo TerraME HPA . 27

Lista de Algoritmos

Capítulo 1

Introdução

Os sistemas Terrestres são submetidos à 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 [L. F. Salazar e Oyama, 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 humanidade.

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 (chamadas políticas saudáveis).

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). O TerraME permite a simulação em espaços celulares bidimensionais que podem ser regulares e irregulares, possuindo suporte a modelagem ambiental espacial dinâmica, que auxilia na construção de modelos computacionais que utilizam conceito de agentes, autômatos celulares [Wolfram, 2003], conceitos de autômatos celulares aninhados (Nested-CA) [Tiago, 2006], multi-escala, multi-paradigma e ainda trabalha com duas importantes inovações que são modelos de autômatos híbridos [Henzinger, 1996] e espaços anisotrópicos [Anselin, 1999].

É importante ressaltar que modelos espaciais dinâmicos são uma representação abstrata de um fenômeno que evolui no tempo e no espaço, baseado em descrições de entidades, processos e relações entre eles. Os resultados destes modelos são mapas que mostram a distribuição

espacial de um padrão ou de uma variável contínua.

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. Na maioria das vezes para que o modelo seja executado em tempo hábil, o modelador muitas das vezes tem que omitir alguns pontos do modelo que consequentemente, distorcem a dinâmica do modelo e influenciam nos resultados finais.

Atualmente as plataformas mais utilizadas baseadas em Teoria Geral dos Sistemas,Teoria de Agentes e autômatos celulares são: *Swarm* [N. Minar, 1996], *Stella* [Costanza e Voinov, 2001], TerraME [Tiago, 2006], *Dinamica-EGO* [B. S. Soares, 2003] e *Repast*. No entanto, tais plataformas não são desenhadas para tirar proveito de arquiteturas computacionais de alto desempenho.

Programar para tais arquiteturas de computadores não é trivial. Na literatura temos a programação baseada no endereçamento compartilhado e troca de mensagens. Mas explicitar tal complexidade ao modelador não é razoável, pois a grande maioria dos modeladores são especialistas no seu domínio de aplicação e não tem conhecimento dos problemas como Dead Lock, acesso a variáveis, seção crítica, dentre outros que surgem ao se utilizar o conceito de computação paralela, 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, foi concebido um conjunto de diretivas a plataforma TerraME 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. Para isso utilizamos o conceito de anotações para instrumentar o código 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 para modelos com baixa dependabilidade, o que sugere *Amdhal* [Amdahl, 1967] como sendo cerca de 95% de paralelismo. Para efetuar 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 no sub-bosque da floresta. Também é considerado no modelo o crescimento e mortalidade das árvores assim como a dispersão das sementes.

O restante deste trabalho esta organizado da seguinte forma: Capítulo 1 é uma introdução

que, apresenta as motivações e objetivos dessa monografia. O Capítulo 2 conceitua modelagem ambiental e a plataforma TerraME. Apresentamos também quais os principais problemas que os modelos espaço temporais enfrentam e as soluções que adotamos para contorná-los na plataforma TerraME. O capítulo 3, descreve a calibração de modelos espaço temporais e os eventuais problemas dessa etapa, em seguida é apresentada uma solução para calibração de modelos de forma paralela. O capítulo 4 descreve o módulo TerraME HPA, uma solução desenvolvida para contornar os problemas referentes a execução dos modelos na plataforma TerraME. No Capítulo 5 descreve-se os testes efetuados e apresenta os resultados do módulo de calibração paralela e para o módulo TerraME HPA. Finalmente, no Capítulo 6 são apresentadas as conclusões finais e discussões sobre os resultados obtidos.

Capítulo 2

Conceitos Básicos

Neste capítulo são apresentados os principais objetivos e características da modelagem ambiental. É detalhado também a estrutura e o funcionamento da plataforma TerraME. São abordados dois softwares livres, o *TerraLib* e o *TerraView* que podem trabalhar em conjunto com o TerraME e por fim é apresentado o paradigma de computação paralela.

2.1 Modelagem

Computadores vêm sendo utilizados como ferramentas de apoio a procedimentos, 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].

Utilizando ferramentas de SIG, é possível prever o impacto que um determinado fenômeno causará. Tendo como objetivo representar propriedades e facilitar o estudo e compreensão de fenômenos que representam riscos e que consequentemente necessitam de monitoramento, modelos matemáticos, aritméticos e lógicos vem sendo implementados [Felgueiras, 1999]. Por exemplo, o aumento da temperatura global poderá implicar em explosões populacionais de vetores de doença tropicais e resultar em epidemias globais [S. Hales, 2002] [A. K. Githeko, 2000] [J. A. Patz, 1998] [Martens, 1995]. Essas simulações podem ser feitas para vários tipos de sistemas, como os sistemas ecológicos, sistemas climáticos, sistema hidrológicos,sistemas de uso do solo.

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 sim-

ulação das interações humano-ambiente são instrumentos potenciais como meio de detecção, identificação, mapeamento e monitoramento das 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.

2.2 TerraME

O 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 [Tiago, 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 pode-se 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, ilustrado 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 a 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 temporais. 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

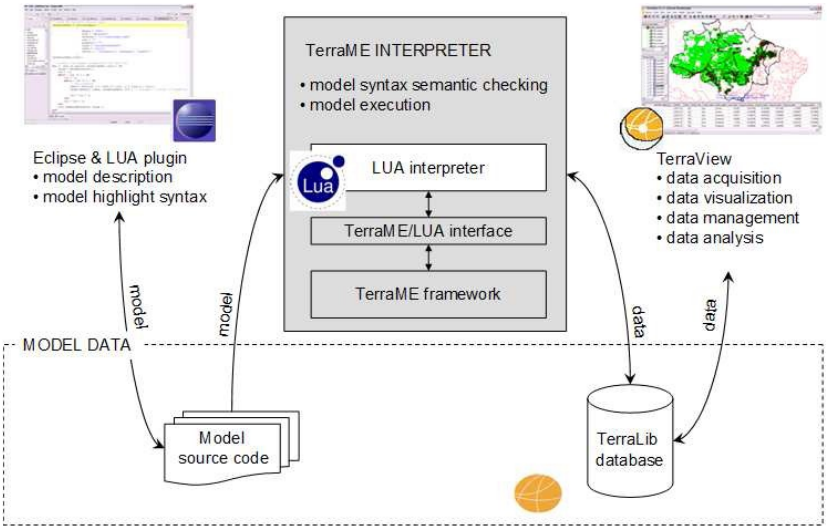

Figura 2.1: Ambiente de desenvolvimento e suporte a modelagem ambiental TerraME

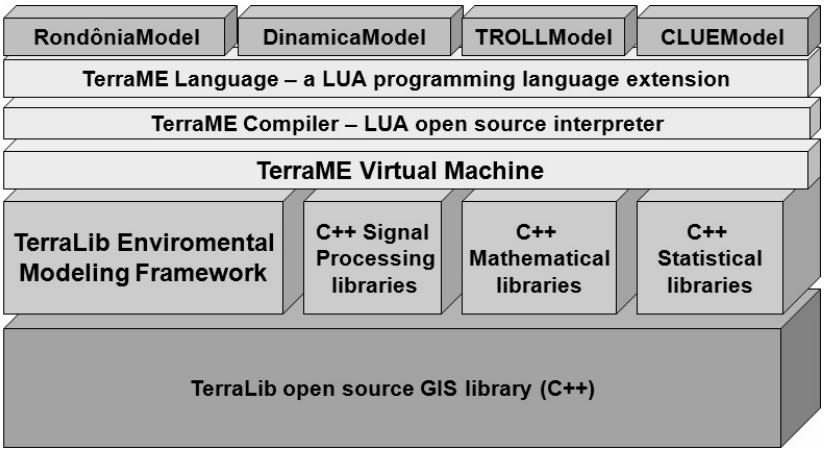

Figura 2.2: Arquitetura do Ambiente TerraME

uma linguagem de modelagem de alto nível.

A terceira camada da arquitetura implementa a linguagem de modelagem TerraME é 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 corresponde a camada de aplicação e inclui os modelos do usuário final.

2.3 A Linguagem LUA

LUA [Ierusalimschy, 2003] é uma linguagem de programação poderosa, rápida e leve, projetada para estender aplicações. Ela 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, é exemplificado a codificação da linguagem LUA, em um exemplo de contador de palavras, as palavras neste exemplo são obtidas através de um arquivo que o usuário deve passar como parâmetro.

```
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, é possível ver um exemplo de comentário na linguagem LUA. Todos os comentários em LUA são precedidos com $--$. Das linhas 2 - 4, são realizadas instanciações de tabelas globais, estas tabelas podem ser acessadas em qualquer parte do código, são consideradas variáveis globais, qualquer variável instanciada sem a palavra reservada *local*. Nas

linhas 7 - 22, é 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 é feita 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 utilizando a linguagem LUA.

Das linhas 11 - 21 e 28 - 34, são apresentados exemplos de laços de repetição. O primeiro laço de repetição é responsável por *"varrer"* (efetuar a leitura de linha por linha)todo o arquivo, e o segundo efetua um *"merge"* (junção dos resultados) entre as duas tabelas utilizadas no exemplo. Nas linhas 13 - 16, é exemplificado 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 para usuário quanto a operações referentes a tabelas. Na linha 15, acontece uma operação de inserção de um dado elemento na tabela T(*linha* é inserida na tabela *T*).

A linguagem 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.

Os principais fatores que levam a escolha da linguagem LUA é por ela ser, bem estabelecida, robusta, rápida, portátil, embutível, simples e livre. A plataforma TerraME por exemplo 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 licenciado como código aberto de acordo com a *LGPL* e desenvolvido com a parceria entre a Divisão de Processamento de Imagem do *INPE*, e a *PUC-RIO*.

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 instruções. 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.[S.V. Adve, 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, 2006a]

Computadores paralelos podem ser classificados de acordo com o nível em que o hardware suporta paralelismo. Computadores com multinúcleos ou multiprocessadores 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 computadores paralelos são mais difíceis de programar que sequenciais,[A. Patterson e Hennessy, 1998] pois a concorrência introduz diversas novas classes de problemas potenciais, como a *dead lock e acessos a variáveis compartilhadas*. 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 caros computacionalmente. 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, 2006b]

1. Problemas de grades estruturadas
2. Problemas de grades não estruturadas
3. Método de Monte Carlo
4. Simulação de máquina de estado finito
5. Busca em grafos

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 modelo, 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. Eles 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 [P. R. B. Lustosa, 2004].

Algoritmos genéticos (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 de 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. Até o fechamento deste trabalho, não foi encontrada nenhuma plataforma 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.

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.

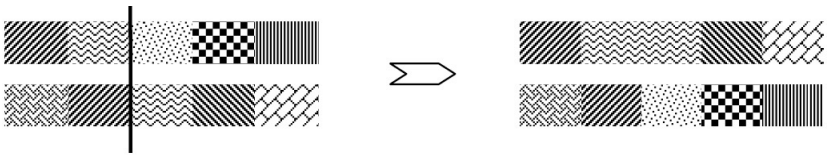

Figura 3.1: Operador cruzamento

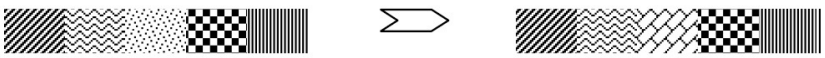

Figura 3.2: Operador mutação

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 a seguir:

CalibracaoAG(boolEstocastico,mModelo, fAvaliacao, boolFlag,precisao, nTamPop, matrizLimites, pc, pm, nMaxGeracoes). O usuário deve passar como parâmetros 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

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.

Quando um modelo é criado existem certo número de parâmetros de entrada e algumas equações que utilizam esses valores para produzir a dinâmica do modelo, logo para que a dinâmica do modelo se assemelhe o mais próximo do real, é preciso realizar a calibração dos parâmetros em questão.

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 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. Ele 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.

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 3.3:

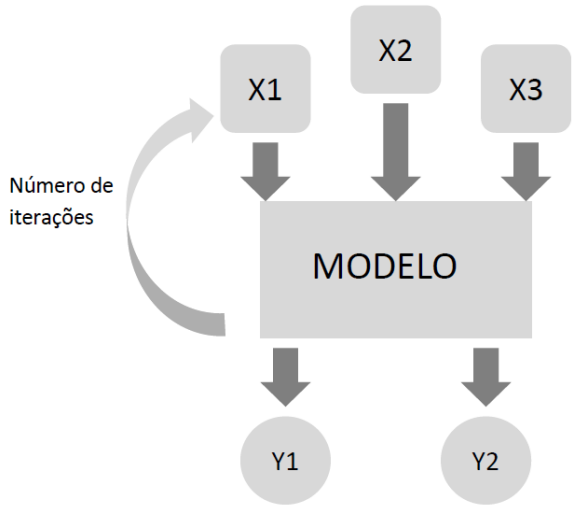

Figura 3.3: Passos de execução Monte Carlo

- **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.

O principal motivo deste trabalho de paralelização do Algoritmo de calibração Monte Carlo, é o fato de geralmente a calibração de modelos envolver 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, 2003].

3.3 Estratégia de paralelização dos algoritmos de calibração no TerraME

Nesta subseção é apresentada a estratégia de paralelização adotada.

Para conceber a solução foi utilizada a estratégia de decomposição dos dados. Os parâmetros de entrada do método de calibração(variáveis a serem calibradas), são divididos em X subconjuntos independentes. Onde a quantidade X de subconjuntos é determinada pela quantidade de *cores* que a máquina a ser utilizada possui. O tamanho de cada subconjunto é determinado pela seguinte regra: $\frac{nMax}{cores}$, onde $nMax$ é a quantidade de iterações do método de calibração.

Como técnica de mapeamento foi utilizada a técnica de *bag of tasks* [Ananth Grama, 2003]. Cada *Task*(tarefa) de calibração é responsável por um subconjunto de parâmetros e é executada em um core da máquina. Ou seja, 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.

Na figura 3.4 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. O número de subconjuntos na *Bag of Tasks* pode variar dependendo das características da simulação.

Ao comparar a tabela de tempo 3.3 com a tabela de tempo 3.4, é possível observar que, o *I/O* é um grande problema na calibração paralela, chegando a influenciar em até 5% no tempo de execução.

O problema de *I/O*, se resume no seguinte fato, os resultados da calibração precisam ser armazenados de forma não volátil a cada simulação. Existem duas formas de executar esta tarefa,*(i)* armazenar todos os dados no final da execução do modelo, *(ii)* ou ao final de toda a simulação.

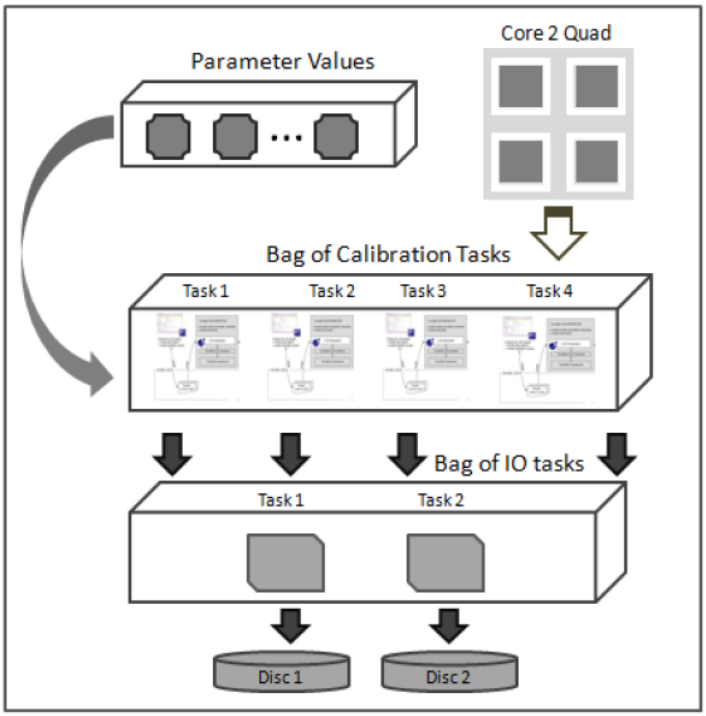

Figura 3.4: Exemplo de uma calibração paralela no TerraME

Exec/Thread's	1(Sequenc.)	2	4	8
10000	4:00:41	2:27:24	1:25:55	1:01:27
15000	6:02:36	3:40:54	2:04:14	1:30:14
20000	8:02:55	4:57:17	2:49:24	2:00:26
25000	10:03:52	6:09:01	3:27:02	2:36:42
30000	12:05:10	7:40:25	4:07:58	3:06:20
60000	24:09:31	14:36:47	8:15:09	5:57:45

Tabela 3.3: Tempo de execução para a calibração de modelos utilizando o algoritmo de Monte Carlo

Como simulações lidam com uma grande quantidade de dados, a segunda alternativa é geralmente inviável, pois implica em um alto consumo de memória. Logo para que fosse possível prover um *I/O* paralelo, foi escolhida a primeira alternativa, pois esta fornece a possibilidade de efetuar operações de *I/O* ao longo da calibração. A partir dessa escolha, foi implementado, um *Bag I/O*, assim toda gravação de dados de forma persistente deve ser enviada primeiro para o *Bag I/O*, antes de ser gravada em disco.

Para realizar a gravação persistente dos dados enviados para o *Bag of I/O*, foi implementado *Task's* que realizam a gravação destes dados de forma persistente, a quantidade de *task's* para *I/O* depende da quantidade de discos rígidos disponíveis na máquina. Cada *Task* de

I/O grava uma parte do *log* final de calibração em um disco, e ao término da calibração essas partes são agrupadas em um único arquivo de *log*. Com o *I/O* sendo realizado em paralelo, o impacto que o *I/O* causa no *speedup* é amenizado.

Exec/Thread's	1(Sequenc.)	2	4	8
10000	4:00:41	2:21:19	1:22:44	0:59:44
15000	6:02:36	3:32:35	1:59:46	1:27:46
20000	8:02:55	4:45:32	2:44:23	1:57:00
25000	10:03:52	5:54:54	3:19:12	2:32:28
30000	12:05:10	7:23:27	3:58:44	3:01:32
60000	24:09:31	14:02:54	7:57:08	5:47:18

Tabela 3.4: Tempo de execução para a calibração de modelos utilizando o algoritmo de Monte Carlo, com melhoria de I/O

Quando um modelo é calibrado utilizando o método genético, os valores dos parâmetros produzidos em cada simulação são utilizados para avaliar as simulações seguintes. Basicamente é possível 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 foi desenvolvido apenas a primeira abordagem, já que o objetivo é construir uma solução escalável e não uma que prove uma acurácia maior.

Para que o usuário use o módulo de calibração paralela, foi adotada 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 é possível executar um modelo com anotações tanto na versão TerraME com o módulo de calibração paralelo e na versão sem o módulo de calibração paralelo(versão sequencial). A Figura 3.5, 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.

Para identificar quais pontos do código o usuário instrumentou o código, foi construído 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 é responsável por converter 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 3.6 é ilustrado a arquitetura do módulo de calibração paralela.

```
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 3.5: Instrumentação necessária para utilizar a calibração paralela dos métodos de Monte Carlo e Genético

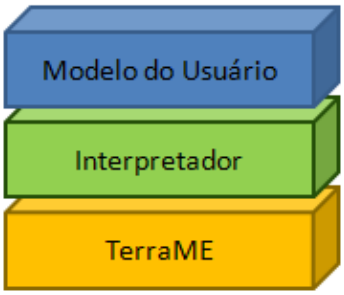

Figura 3.6: Interpretador para o módulo de calibração paralela

Capítulo 4

Simulação paralela no TerraME

No capítulo anterior foi apresentado o módulo de calibração paralela para modelos construídos na plataforma TerraME. 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.

Diante do cenário exposto acima, é apresentado neste capítulo a solução paralela TerraME para simulação de modelos espacialmente explícitos.

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 discutidos em seguida 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.
- HPA JOINALL, diretiva semelhante a especificada acima, no entanto *JOINALL*, aguarda até que todas as funções que foram chamadas até o momento utilizando o módulo TerraME HPA, tenham sido executadas por completo.
- HPA LOCK Gl, diretiva responsável pela sincronização de *thread's*, ela gerência o acesso a um recurso global(*Gl*). Com *HPA LOCK* restringe-se 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 construído. O interpretador recebe o modelo do usuário como entrada e efetua a interpretação das diretivas que compõem o TerraME HPA.

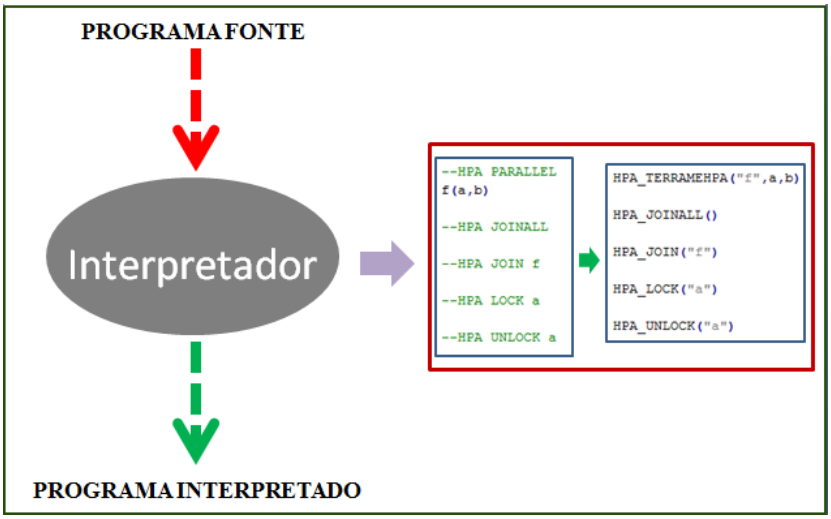

Figura 4.1: Interpretador do módulo TerraME HPA

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 determinada pelo número de núcleos de processamento da máquina.

A pilha de execução do modelo possui todas as variáveis globais e locais, 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 LUA.

Toda instrução do modelo passa pelo interpretador TerraME HPA, quando este detecta 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.

Como no modulo apresentado anteriormente, foi adotada uma *bag of task's*, esta é compartilhada entre todos os *worker's*. A cada função previamente instrumentada para ser executada de forma paralela, o interpretador adiciona o nome desta função e seus respectivos parâmetros a *bag of tasks*. Ao efetuar a inserção na *bag os task's*, uma notificação(*Wake Up*) é enviada a um dos *worker's* disponíveis que estiver em estado de *Sleeping* (ocioso), este *worker* deve acessar a *bag of task's* para executar a primeira função disponível na *bag of task's*. Caso não existam *worker's*, em estado ocioso, a notificação não é enviada a nenhum deles, e em algum momento um *worker*, vai acessar a *bag of task's* e assumir a execução dessa função, uma vez que os *worker's*, executam até que existam elementos na *bag of task's*.

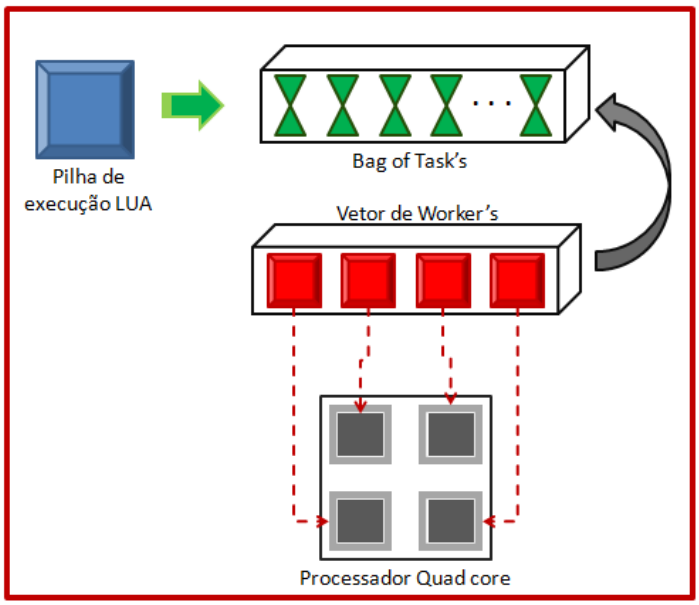

Figura 4.2: Arquitetura do módulo TerraME HPA

Um dos grandes desafios em conceber soluções paralelas utilizando a linguagem LUA, é a falta de suporte a programação paralela. LUA oferece como suporte ao conceito de multi tarefas, *coroutines* e a infra estrutura *ALua* [C.Ururahy, 1999].

Coroutines, 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 *coroutines* e *threads* é que conceitualmente, um programa com *threads* executa vários segmentos simultaneamente. *Coroutines*, por outro lado, são colaborativas, ou seja um programa que utilize *coroutines* 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).

O *ALua* é uma infra estrutura para a linguagem LUA, que auxilia na construção de aplicações distribuídas orientadas a eventos. Os eventos do *ALua* são mensagens assíncronas constituídas de trechos de códigos LUA,que são executados no receptor. Como é o caso das *coroutines* a infraestrutura *ALua*, não atende aos requisitos do modulo TerraME HPA, pois não provê de fato a execução concorrente das funções, não tem suporte a variáveis globais e exige modificações no código original.

Devido a falta de suporte da linguagem LUA a programação simultânea. Cada *Worker* foi implementado utilizando o recurso *QThread* da plataforma *QT*. Cada *Worker*, possui uma pilha de execução LUA local, tem acesso a pilha de execução global e a *Bag of Task's*.

Cada elemento da *Bag of Task's* como mencionado anteriormente, contêm o nome de cada função que devera ser executada e 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 of Task's* 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. Quando ocorre uma nova inserção no *Bag of Task's* de uma nova chamada de função a ser executada.

A execução do modelo termina quando a última instrução do modelo do usuário passa pelo interpretador e é executada pelo *kernel* TerraME. Ou seja quando a *bag of Task's* estiver vazia e a pilha LUA principal estiver sem instruções a serem executadas. Na Figura 4.2 ilustramos o cenário com quatro *workers* numa arquitetura de computador quad-core.

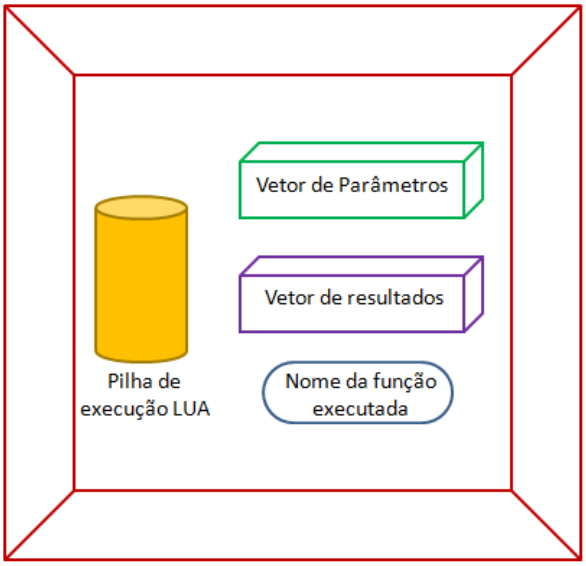

Figura 4.3: Estrutura de um Worker

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 entender melhor o exposto acima, vejamos o exemplo 4.4,que corresponde a um contador de palavras paralelo, é possível observar o uso das diretivas do módulo TerraME HPA. Na linha 19, podemos observar o uso da diretiva *HPA PARALLEL*, responsável por delegar a execução da função logo abaixo para algum dos *worker's* que estejam disponíveis. Neste exemplo a função *CONTADOR_ARQ* é chamada duas vezes, através de um laço de repetição. Nos casos em que a máquina ofereça suporte a multitarefa (multi-processada), as duas chamadas da função, vão executar em *Worker's* distintos (ou seja estarão executando simultaneamente).

Na linha 22, podemos ver a utilização da diretiva *HPA JOINALL*. Esta diretiva é responsável por obrigar a pilha LUA principal, aguardar o processamento de todas as chamadas de

funções feitas anteriormente. No nosso exemplo a pilha principal aguarda até que as duas chamadas feitas a função *CONTADOR_ARQ*, tenham terminado a execução.

```
1  T1 = {}
2  T2 = {}
3  T_vet = {T1,T2}
4  T_TOTAL = {}
5
6  function CONTADOR_ARQ(indx,T)
7      local fread = io.open("D:\\A"..indx.."\\Arq"..indx..".txt");
8      for line in fread:lines() do
9          if T[line] == nil then
10             table.insert(T,line)
11             T[lisne] = 1
12         else
13             T[line] = T[line] + 1
14         end
15     end
16 end
17
18 for i=1,2 do
19     --HPA PARALLEL
20     CONTADOR_ARQ(i,T_vet[i]);
21 end
22 --HPA JOINALL
23
24 T_TOTAL = T1
25 for ind=1,table.maxn(T2) do
26     if T_TOTAL[T2[ind]] == nil then
27         table.insert(T_TOTAL,T2[ind])
28         T_TOTAL[T2[ind]] = 1
29     else
30         T_TOTAL[T2[ind]] = T_TOTAL[T2[ind]] + T2[T2[ind]]
31     end
32 end
```

Figura 4.4: Exemplo da instrumentação de um código, para um contador de palavras paralelo

Uma solução onde não seja necessário utilizar as diretivas *HPA LOCK* e *HPA UNLOCK* explicitamente esta atualmente em construção. Essa solução vai reduzir ainda mais as instrumentações que os modeladores necessitam inserir em seu modelo, para utilizar dos recursos do módulo TerraME HPA. Tudo isso vai tornar o módulo TerraME HPA, mais intuitivo para o modelador.

Capítulo 5

Experimentos

Para o estudo de casos, foi mensurado o tempo gasto para a execução, e para conhecer o comportamento que cada solução traz em diferentes cenários, variou se 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, a solução foi avaliada executando testes com 10000, 15000, 2000, 25000, 30000 e 60000 simulações. Já para o método Genético foi testado com 500, 1000, 1500, 2000, 2500 e 5000 simulações.

5.1.1 Testes para o Método 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*, obteve 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* ocasionou em 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, o que pode se observar no teste com 60000 simulações. Inicialmente o *speedup* executando 60000 simulações e com duas *thread's* é semelhante aos demais cenários, no entanto, o *speedup* ao se utilizar quatro *thread's* e oito *thread's* apresenta um pequeno aumento, comprovando o ganho do *speedup* ao se utilizar um número maior de simulações.

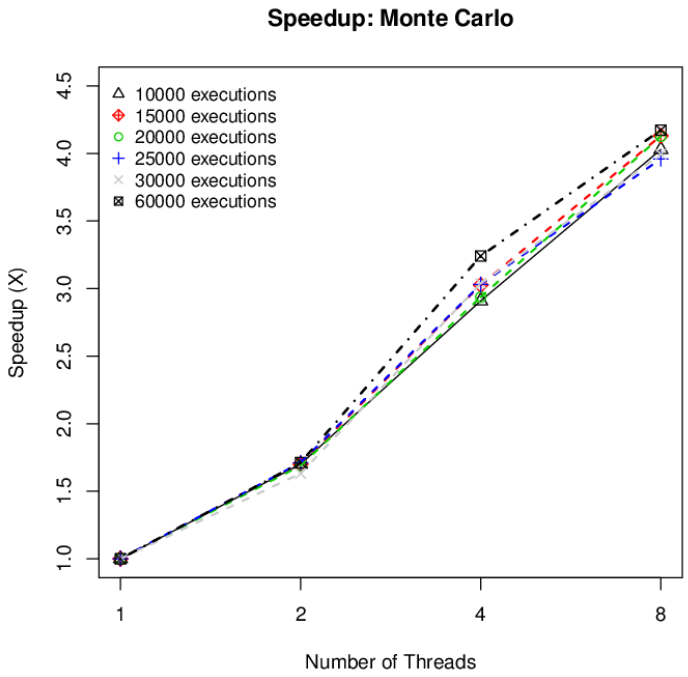

Figura 5.1: Speedup para o teste de Monte Carlo

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, algoritmo genético e método de Monte Carlo eram esperados, pois o número de discos é menor do que o número de núcleos de processamento. A perda 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 sendo a principal razão do *speedup* ser 50% do linear ao se utilizar oito *thread's* em uma máquina com 8 núcleos de processamento. A cada simulação do processo de calibração, os resultados são armazenados no disco, mas nos experimentos, são disponíveis apenas quatro discos. Com essa restrição de número de discos rígidos, ao executar a calibração utilizando 8 *thread's*, 2 *thread's* escrevem concorrentemente em cada disco.

Tanto os métodos de Monte Carlo e o método de calibração genética foram testados com dez *thread's*, mas os resultados, como esperado, tem *speedup* bem inferior comparando se com o cenário onde foi utilizado 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, os resultados da calibração paralela comparado com a versão sequencial(original) foram os mesmos.

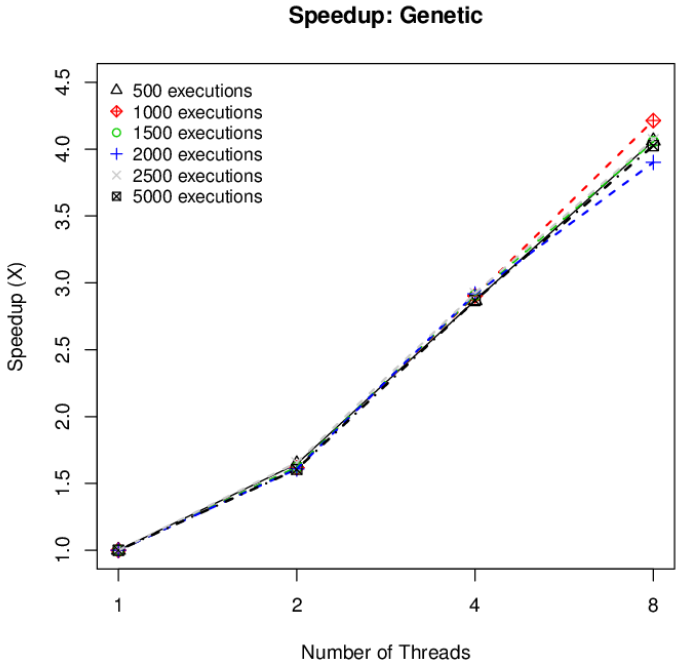

Figura 5.2: Speedup para o teste Genético

5.2 Experimentos para simulação paralela

Para efetuar os testes do módulo TerraME HPA, foram utilizadas duas abordagens, (i) o módulo TerraME HPA foi utilizado para executar um modelo altamente paralelizável (contador de palavras paralelo),este modelo auxilia a observar 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 foi utilizado o modelo TROLL [Chave, 1999], para tal fim. No gráfico 5.3 é apresentado o *Speedup* da solução e é feita uma análise a respeito dos resultados alcançados.

Teste/Thread's	1(Sequenc.)	2	4	8
Contador de palavras	00:10:30	00:05:19	00:02:57	00:01:22
TROLL	00:19:57	00:28:48	00:29:40	00:28:24

Tabela 5.1: Tempo de execução para os testes realizados utilizando o módulo TerraME HPA

É possível observar pelos resultados ilustrados na Figura 5.3 e pela tabela 5.1, que para o Modelo *Contador de Palavras* o *Speedup* da solução chegou a ser linear com a utilização de duas *thread's*, para quatro e oito *thread's* foi obtido um Speedup de aproximadamente 95% do linear. Neste modelo não há nenhuma dependência entre os dados, dessa forma não é complicado atingir tal resultado.

Para o modelo TROLL é possível observar um *Speedup* abaixo da solução sequencial, uma

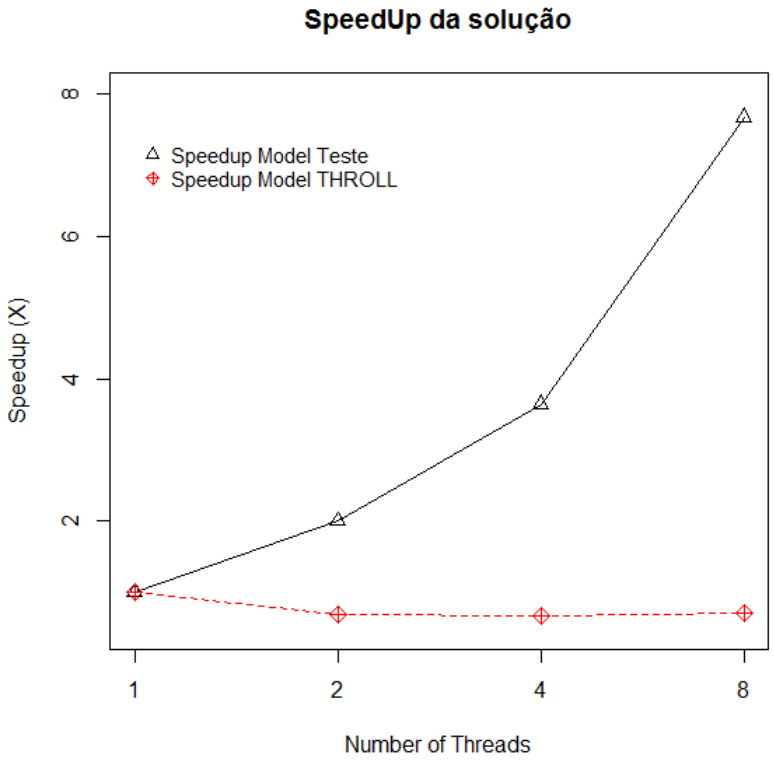

Figura 5.3: Speedup para os testes utilizando o módulo TerraME HPA

das razões deste resultado é a alta dependabilidade cronológica que cada função do TROLL possui. A paralelização do modelo *TROLL*, foi possível de ser realizada apenas, paralelizando pequenas partes independentes do modelo, ou seja são funções de baixa complexidade computacional que são executadas de forma exaustiva. Estes dois motivos implica em 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 em algumas limitações. Uma delas é que todas variáveis globais estão contidas na pilha de execução principal (pilha *MAIN*) do LUA, as pilhas de execução dos *worker's* referenciam a pilha de execução principal, pois estas necessitam acessar as variáveis globais. Devido a esta referencia é necessário sincronizar o início e o final da execução das pilhas de execução dos *worker's*(ou seja, existe uma seção crítica no inicio e final das pilhas de execução, elas devem ser iniciadas e finalizadas uma após a outra).

A limitação discutida acima, para o caso do modelo TROLL influencia drasticamente no *speedup*, pois como a paralelização para este modelo foi possível, apenas paralelizando funções de baixa complexidade e que devem ser executas de forma exaustiva, existe então uma alta taxa de sincronização de inicio e finalização entre os *worker's*. Outro fator que degrada o *speedup* é a sincronização do acesso a variáveis globais. A diretiva *HPA LOCK*, restringe o

acesso a uma variável global inteira. Por exemplo se a variável global X é composta por vários campos, ao se utilizar a diretiva *HPA LOCK*, todos os campos dessa variável estarão bloqueadas para acesso até ocorrer o desbloqueio com *HPA UNLOCK*.

Uma forma de amenizar o impacto do acesso a variáveis globais, seria adicionar a capacidade de bloquear apenas um campo de uma variável global. Ou seja, apenas o campo ou os campos que serão modificados por um *worker* serão bloqueados, permitindo assim acesso aos campos restantes da variável global. Por exemplo vamos considerar a variável global X sendo uma matriz de dimensões *3 X 3*, e vamos considerar que um *worker*, necessita modificar na posição *X[1][1]*, com a diretiva *HPA LOCK*, tendo a capacidade de bloquear porções de menor granularidade, apenas a posição da variável *X[1][1]*, seria bloqueada, o restante das posições estaria disponível para ser acessada apelo demais *worker's*.

Para validar os resultados obtidos ao se utilizar o módulo TerraME HPA, foi selecionado 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

A estratégia do modulo de calibração paralela é simples, eficiente e representou os estudos iniciais de um plano mais ambicioso que é o módulo TerraME HPA. Em geral, foi possível alcançar um *speedup* de 75-80% do 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. Nesta solução não foi verificada a qualidade da acurácia.

As diretivas oferecidas pelo módulo TerraME HPA, são intuitivas e não necessitam da alteração do modelo sequencial original do usuário. Os testes de desempenho 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 sendo o mais eficiente o possível. Já para modelos como o TROLL 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, foi desenvolvida apenas uma versão de memória compartilhada para a plataforma TerraME. Com uma versão distribuída pode ser possível alcançar um *speedup* mais próximo do linear do que a versão paralela. Atualmente, esta sendo investigada a possibilidade de utilização de *GPU's* para utilização nas simulações de modelos paralelos.

6.1 Trabalhos futuros

Como trabalhos futuros pretende-se, adicionar ao módulo TerraME HPA a capacidade de execução distribuída. Distribuindo a execução do TerraME, o poder computacional disponível será aumentado, pois será possível executar simulações de modelos em um *grid* de computadores,e com isso aumentar consideravelmente o espaço de memória disponível para simulação, uma vez que mais máquinas estarão a disposição. 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

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

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

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

[Amdahl, 1967] Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. *AFIPS spring joint computer conference.*

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

[Anselin, 1999] Anselin (1999). Interactive techhniques ando exploratory spatial data analysis. *Geographical Information Systems*, pp. 253 – 266.

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

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

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

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

[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.*

[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.

[C.Ururahy, 1999] C.Ururahy, N. (1999). Alua: An event-driven communication mechanism for parallel and distributed programming. *ISCA 12th International Conference on Parallel and Distributed Computing Systems.*

[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.*

[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.*

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

[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.*

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

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

[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*.

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

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

[P. Hadjidoukas, 2003] P. Hadjidoukas, C. Bousis, 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*.

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

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

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

[SHMITT, 2004] SHMITT, L. M. (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*.

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

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

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

[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.