

Adriano Serckumecka

***Avaliação do Uso de Computação Paralela
Utilizando uma Rede P2P na Simulação de Dados
Climáticos: Velocidade e Direção do Vento***

Ponta Grossa

2012

Adriano Serckumecka

***Avaliação do Uso de Computação Paralela
Utilizando uma Rede P2P na Simulação de Dados
Climáticos: Velocidade e Direção do Vento***

Dissertação apresentada a Universidade Estadual de Ponta Grossa, para a obtenção do Título de Mestre em Computação Aplicada - Área de Concentração: Computação Aplicada à Agricultura.

Orientador: Prof. Dr. José Luciano Senger

Ponta Grossa

2012

Ficha Catalográfica Elaborada pelo Setor Tratamento da Informação Bicen/UEPG

- S482a Serckumecka, Adriano
 Avaliação do uso de computação paralela utilizando uma rede P2P na
 simulação de dados climáticos : velocidade e direção do vento / Adriano
 Serckumecka. Ponta Grossa, 2012.
 57 f.
- Dissertação (Mestrado em Computação Aplicada - área de concentração
 Computação Aplicada a Agricultura), Universidade Estadual de Ponta Grossa.
 Orientador: Prof. Dr. José Luciano Senger.
1. Computação paralela. 2. Simulação climática. 3. Redes P2P. 4. Ventos.
 I. Senger, José Luciano. II. Universidade Estadual de Ponta Grossa. Mestrado
 em Computação Aplicada. III. T.

CDD: 004.22

Adriano Serckumecka

**Avaliação do Uso de Computação Paralela Utilizando uma Rede P2P na
Simulação de Dados Climáticos: Velocidade e Direção do Vento**

Dissertação apresentada para obtenção do título de mestre na Universidade Estadual de Ponta Grossa, Área de Computação Aplicada à Agricultura.

Ponta Grossa, 17 de julho de 2012.

Prof. Dr. Luciano José Senger - Orientador
Doutor em Ciências de Computação
Universidade Estadual de Ponta Grossa

Profa. Dra. Maria Salete Marcon Gomes Vaz
Doutora em Ciências de Computação
Universidade Estadual de Ponta Grossa

Prof. Dr. Jorim Sousa das Virgens Filho
Doutor em Engenharia Agrônômica
Universidade Estadual de Ponta Grossa

Dedico: Aos meus pais Valdomiro e Lidia e minha amada Scheila.

Agradecimentos

Agradeço a Deus pela força, coragem e saúde para a concretização de mais uma etapa em minha vida.

À Universidade Estadual de Ponta Grossa, pela oportunidade de qualificação e por ceder os recursos para o desenvolvimento deste trabalho.

Ao meu orientador prof. Dr. Luciano José Senger, pelo apoio prestado no desenvolvimento e por acreditar nesta proposta e em mim.

Ao prof. Dr. Jorim Sousa das Virgens Filho pela contribuição prestada, por ceder gentilmente seu tempo e os materiais para realização dos experimentos.

Ao prof. Dr. Marcio Augusto de Souza, pelo apoio e pelos conselhos que só contribuíram para a melhoria deste trabalho.

Aos meus amados pais, Valdomiro e Lidia, pelo apoio incondicional que sempre me dedicaram e pelo exemplo de vida.

À minha amada Scheila, pela dedicação, compreensão e companheirismo durante esta jornada.

Aos meus irmãos Fabio e Laila e toda minha família que sempre me incentivaram e apoiaram neste objetivo.

Aos colegas e amigos que me acompanharam durante esses dois anos.

A todos que, direta ou indiretamente, contribuíram na elaboração deste trabalho.

Resumo

O objetivo principal deste trabalho é a avaliação de sistemas distribuídos baseados em redes par-a-par (P2P), juntamente com técnicas de computação paralela, para reduzir os tempos de resposta de simulações climáticas. Um modelo probabilístico específico para simulação de dados de vento foi adotado e duas aplicações computacionais foram implementadas e avaliadas. A primeira aplicação foi desenvolvida com o uso do Framework P2PComp, que permite a criação de programas paralelos e sua execução em redes P2P. A segunda aplicação adotou diretamente uma rede P2P, sem o uso desse framework, como infraestrutura de comunicação. Dados climáticos da região do Município de Lapa-PR, compreendendo o período entre 1998 e 2007 foram empregados nos experimentos. Os resultados obtidos demonstram a viabilidade de utilização de computação paralela em redes P2P, nas simulações climáticas. Os melhores resultados medidos em relação ao fator de aceleração foram observados em situações onde foi utilizado um número maior de pares (igual a 26) e simulações climáticas para um período de tempo igual a 100 anos. Para este caso, o fator de aceleração obtido foi aproximadamente igual a 7.

Palavras-chave: Computação Paralela, Simulação Climática, P2P, Vento

Abstract

The main objective of this work is the evaluation of distributed systems based on peer-to-peer network and parallel computing techniques to reduce response times of climate simulations. A probabilistic model for simulating wind data was used and two computing applications was implemented and evaluated. The first application was developed by using the framework P2PComp, which is directed to building parallel software and its deploy in P2P networks. The second application was developed without the support of this framework and using straightly the communication infrastructure software. Climatic data of the municipality of Lapa, Paraná, comprising the period between 1998 and 2007 were used in the experiments. The results demonstrate the feasibility of using parallel computing and P2P networks for executing climate simulations. Best results, measured by the speedup obtained, were observed when using multiple peers (equal to 26) and when executing simulations for a long period of time (near to 100 years). In such case, the speedup obtained is near to 7.

Keywords: Parallel Computing, Climate Simulation, P2P, Wind

Lista de Figuras

3.1	Comunicação entre <i>Peers</i> que não estejam numa mesma rede local	29
4.1	Estrutura de uma base de dados de vento	36
4.2	Base de dados simulada.	38
4.3	Dados de validação para uma base simulada	39
4.4	Funcionamento do P2PComp na execução de uma tarefa	40
4.5	Funcionamento do JXTA Venthor na execução de uma tarefa	41
4.6	Gráfico com os tempos de execução incluindo o Venthor Original	45
4.7	Gráfico com os tempos de execução sem o Venthor Original	46

Lista de Tabelas

4.1	Tempos de execução para o Venthor original	42
4.2	Tempos de execução para o Venthor Java (sem paralelização)	43
4.3	Tempos de execução para o Venthor P2PComp (paralelizado) utilizando 6 Peers	43
4.4	Tempos de execução para o Venthor P2PComp (paralelizado) utilizando 14 Peers	43
4.5	Tempos de execução para o Venthor P2PComp (paralelizado) utilizando 26 Peers	44
4.6	Tempos de execução para o JXTA Venthor (paralelizado) utilizando 6 Peers . .	44
4.7	Tempos de execução para o JXTA Venthor (paralelizado) utilizando 14 Peers .	44
4.8	Tempos de execução para o JXTA Venthor (paralelizado) utilizando 26 Peers .	44
4.9	Tempos de execução para todas as versões do Venthor	45
4.10	Comparativo entre o <i>speedup</i> do Venthor original e as demais versões	47
4.11	Comparativo entre o <i>speedup</i> do Venthor Java e das versões paralelizadas . . .	48

Sumário

1	Introdução	11
2	Revisão da Literatura	14
2.1	Variável Climática Vento	16
2.2	Computação Paralela e Distribuída	17
3	Materiais e Métodos	20
3.1	Considerações Iniciais	20
3.2	JXTA	20
3.3	O Framework P2PComp	29
3.4	O Software Venthor	31
3.4.1	Detalhes do Simulador	32
3.5	Recursos Utilizados nos Experimentos	33
4	Resultados	35
4.1	Migração para a Linguagem Java	35
4.2	Adaptação e Integração ao P2PComp	37
4.2.1	Esquema do Funcionamento no P2PComp	39
4.3	Ferramenta JXTA Venthor	40
4.3.1	Esquema do Funcionamento no JXTA Venthor	41
4.4	Tempos de Execução	42
4.5	Resultados de Speedup	46
4.5.1	Venthor Original versus Java e P2P	47

4.5.2	Venthor Java versus P2P	47
5	Conclusão	49
	Referências Bibliográficas	50
	Anexo A – Equações do Modelo	53
A.1	Distribuições de Probabilidade	53
A.1.1	Distribuição de Weibull	53
A.1.2	Distribuição de Rayleigh	54
A.1.3	Distribuição de Beta	54
A.1.4	Distribuição de Gama	55
A.2	Teste de Kolmogorov-Smirnov	56
A.3	Cálculo da Direção do Vento	57

1 *Introdução*

As variáveis climáticas influenciam consideravelmente a produtividade de diversas culturas agrícolas em todo o mundo, podendo agir como facilitadoras na maximização da produção, ou como vilãs, podendo prejudicar toda uma safra, trazendo prejuízos aos produtores e afetando a economia de um país.

As pesquisas agronômicas alcançaram avanços significativos no que diz respeito à produtividade e qualidade de sementes e plantas, através do desenvolvimento de insumos e defensivos e, também, através da manipulação genética. Infelizmente, quando se trata de fenômenos naturais e do clima, pouco pode ser feito para evitar perdas nas culturas, sejam elas por estiagem, períodos chuvosos, granizos, vendavais ou geadas.

Um dos recursos que pode auxiliar os agricultores e pesquisadores na tomada de decisões é a utilização de simulação climática. Através dela é possível prever as ocorrências, para um período de tempo desejado, de variáveis climáticas como: precipitação, temperatura, umidade do ar, velocidade e direção do vento.

Para a realização dessas estimativas são utilizados modelos computacionais matemáticos e estatísticos, aplicados a uma série de dados históricos, coletados por estações meteorológicas de uma determinada região. Embora a simulação de sistemas possa ser empregada sem o uso do computador, frequentemente requer o desenvolvimento de programas complexos e um grande investimento de tempo de programação e experimentação [1].

Devido aos avanços tecnológicos na área da computação, sistemas de simulação começam a ser aplicados em maior escala, mas ainda com restrições devido aos altos preços dos equipamentos, ou seja, melhorou-se consideravelmente a capacidade computacional, mas muitas vezes ela ainda é inacessível.

Uma alternativa ao uso de supercomputadores é a utilização de técnicas de processamento paralelo e distribuído, como os *clusters*¹ e grades computacionais, que aproveitam o poder

¹ Máquinas interligadas numa rede fortemente acoplada para compartilhamento de recursos.

computacional ocioso de computadores *desktop* interligados através de redes físicas ou lógicas para resolverem cálculos que exigem maior poder de processamento.

Além do alto custo dos supercomputadores, um dos fatores que contribuíram para a ascensão da computação paralela e distribuída foi a limitação de *clock* (ciclos de execução) que as tecnologias atuais podem alcançar. Sendo assim, tem-se um custo/benefício maior na utilização de multiprocessadores ou multinúcleos do que investir no desenvolvimento de um núcleo mais rápido e potente. Desde o lançamento desses processadores, empresas e pesquisadores buscam aprimorar e criar técnicas de desenvolvimento de software, algoritmos e hardware que se encaixem nos novos padrões, sendo que atualmente, estão em uso diversas técnicas e aplicações que utilizam os princípios da computação paralela e distribuída como: banco de dados distribuídos, grades computacionais, *clusters*, redes P2P e *web crawlers* (programa que coleta e armazena páginas da Internet) [2] [3].

Através da associação de técnicas de computação paralela e distribuída em simulações climáticas, torna-se possível a redução no tempo de execução das simulações de dados de velocidade e direção do vento realizadas, onde são utilizados algoritmos sequenciais, executados numa única máquina e sem paralelismo, como no Software Venthor [4] [5].

A contribuição que pretende-se trazer à agricultura através dessas simulações, é auxiliar nas tomadas de decisões como: quais os dias e/ou horários mais propícios para aplicação de insumos ou defensivos; quais culturas melhor se adaptam às condições do local estudado, devido às condições de vento e identificar regiões de maior polinização.

Como objetivo geral, pretende-se abrir caminho para a aplicação dessa técnica (paralelização de simulações) para as demais variáveis climáticas, já que faz uso de rede P2P através do *Framework* P2PComp, podendo ser desenvolvidos novos aplicativos de forma semelhante ao apresentado aqui, ou apenas adaptando o mesmo conforme a necessidade.

O objetivo específico deste trabalho é reduzir o tempo gasto nas simulações realizadas pelo Software Venthor, através da utilização de técnicas de computação paralela e distribuída e de redes ponto-a-ponto.

Os próximos capítulos estão organizados conforme descrito a seguir:

Capítulo 2: Revisão da Literatura (Onde são descritos os principais simuladores e geradores estocásticos climáticos; a escolha da variável Vento; tecnologias e ferramentas de computação paralela e distribuída).

Capítulo 3: Materiais e Métodos (A tecnologia JXTA e suas características; detalhes do Framework P2PComp; funcionamento do Software Venthor e detalhes do simulador; recursos

utilizados nos experimentos).

Capítulo 4: Resultados (Migração do simulador para a linguagem de programação JAVA; Integração ao P2PComp; criação da nova rede P2P, JXTA Venthor; tempos de execução das simulações e o cálculo do fator de aceleração).

Capítulo 5: Conclusão.

Anexo A: Descrição das equações matemáticas empregadas no modelo de simulação.

2 *Revisão da Literatura*

Ao longo do tempo, a pesquisa conduzida de forma estruturada desenvolveu técnicas e métodos que bastante contribuíram para impulsionar a humanidade no processo de conhecimento e entendimento do universo. Entre as técnicas de pesquisa, a de simulação, tornou-se amplamente difundida após o grande avanço da informática por oferecer a possibilidade de simular um sistema real com alto grau de semelhança e permitir o estudo de sistemas complexos [6].

A utilização de simulações na área agrícola iniciou-se nos anos 70 nos Estados Unidos, e no Brasil a partir de 1974 na Embrapa, no antigo Departamento de Métodos Quantitativos - DMQ [7]. Devido ao crescimento populacional e escassez de recursos naturais, pesquisadores e profissionais da área viram nas técnicas de simulação uma possibilidade de melhorar resultados, como: previsão de safra, crescimento de plantas, desenvolvimento de pragas, previsão de perdas, potencial produtivo, relação entre a variação climática e desenvolvimento das culturas, entre outras.

Em Virgens Filho[6] são enumerados três fatores, que juntos favorecem a aplicação de técnicas de simulação à agricultura: a automatização das estações meteorológicas, a capacidade de disseminação destas informações por meio de sistemas eletrônicos e, a simulação computacional de processos biológicos associados com a produção agrícola.

Das variáveis que envolvem a agricultura, o clima ainda é a mais importante no cultivo das mais diversas culturas, principalmente pelo fato de ser praticamente impossível para o homem controlá-lo. Isto ocorre através das influências que o clima exerce sobre as fases da produção agrícola, como a colheita, estocagem, transporte e comercialização.

Existem bons simuladores climáticos, fruto de pesquisas envolvendo computação, matemática, estatística e agricultura. Alguns dos mais utilizados serão descritos a seguir.

LARS-WG. É um gerador climático estocástico desenvolvido pelo Dr. Mikhail Semenov do Rothamsted Research, do Reino Unido. Um gerador climático estocástico é um modelo estatístico capaz de simular dados meteorológicos diários, com base nas características estatísticas

observadas do clima em um único local. Este tipo de modelo estatístico pode:

- Gerar longa série temporal de dados meteorológicos (tipicamente precipitação, temperatura e radiação solar) adequados para a avaliação do risco agrícola e hidrológico;
- Fornecer um meio para a simulação de dados meteorológicos em locais onde não há observações, ou onde o registro de observação é muito curto; e
- Servir como uma ferramenta de computação de baixo custo que pode produzir cenários de mudança climática de alta resolução.

Este modelo em particular é capaz de incorporar as mudanças na variabilidade climática, assim como mudanças na média climática [8] [9].

CLIGEN. Cligen é um simulador climático estocástico que produz estimativas diárias de precipitação, temperatura, ponto de orvalho, vento e radiação solar para um único ponto geográfico, usando parâmetros mensais (médias, desvio padrão, assimetria, etc) derivados a partir das medições históricas [10].

WGEN. Um gerador climático estocástico, que produz séries temporais de dados meteorológicos artificiais de comprimento ilimitado para uma localização, com base nas estatísticas de clima observadas. Estes tipos de modelos estatísticos são geralmente desenvolvidos em duas etapas, com o primeiro passo focando a modelagem da precipitação diária, enquanto o segundo passo concentra-se nas demais variáveis de interesse, tais como temperatura máxima e mínima, radiação solar, umidade e velocidade do vento, que são modeladas e condicionadas à ocorrência de precipitação [11].

WEATHERMAN. O Weatherman é um pacote de software concebido para auxiliar na preparação de dados meteorológicos diários para uso com modelos de simulação. O software pode importar ou exportar arquivos climáticos diários, com qualquer formato de coluna, e converter os dados em formato desejável. Os dados são verificados e sinalizados para identificar possíveis erros na importação. O gerador estocástico é baseado no WGEN e SIMMETEO [12].

No Brasil, poucos trabalhos relacionados à simulação climática ainda são vistos. Um dos precursores na utilização desses modelos foi Virgens Filho[13]. Abaixo são descritas as características do GEPAC - Gerador Estocástico de Parâmetros Climáticos e do SEDAC-R - Simulador Estocástico de Dados Climáticos com Retroalimentação.

GEPAC. Ferramenta Computacional para simulação das variáveis climáticas: precipitação pluviométrica, radiação solar global, umidade do ar e temperatura (máxima e mínima). Os mo-

delos probabilísticos para cada componente climático, foram baseados em distribuições de probabilidade determinados mensalmente, através do teste de aderência de Kolmogorov-Smirnov¹. A escolha dos modelos foi baseada na ocorrência de precipitação, determinada pela primeira ordem da cadeia de Markov, com dois estados [13].

SEDAC-R. Ferramenta computacional para simulação das variáveis climáticas: precipitação pluviométrica, radiação solar global e temperatura (mínima e máxima), estabelecendo um algoritmo baseado na parametrização dinâmica das distribuições de probabilidade, decorrente da retroalimentação de dados [6].

2.1 Variável Climática Vento

Dentre as diversas variáveis climáticas, o vento foi escolhido como base para o desenvolvimento desse trabalho, porque assim como as demais variáveis climáticas, tem seu grau de importância para o clima e conseqüentemente para a agricultura, atuando em todas as fases do ciclo de cultivo de uma cultura agrícola. Também foi escolhido devido ao fato de ser uma das que menos aparece na literatura e em estudos relacionados. Outra variável poderia ser escolhida da mesma forma, já que seguem padrões de simulação parecidos, podendo variar os parâmetros e a escolha da distribuição de probabilidade, sendo perfeitamente aplicáveis a esse estudo.

Encontram-se na literatura alguns estudos com modelos probabilísticos propostos para se ajustar dados de distribuições empíricas de velocidade do vento [14] [15] [16]. Entretanto, um erro que pode ocorrer na análise de dados climatológicos desta natureza é o fato de desprezar-se as características da distribuição de probabilidade mais adequada para os dados em estudo. Tal equívoco pode resultar na utilização desnecessária de um modelo mais complexo e trabalhoso, assim como no emprego de um modelo simplificado, porém que resulte em conclusões erradas, se os dados não aderirem a essa distribuição. É importante ressaltar que o ajuste de uma distribuição teórica a uma distribuição empírica de probabilidade envolve, na sua essência, um procedimento para verificar se a primeira é capaz de representar, o mais idêntico possível, a segunda. No caso deste e de outros estudos, a qualidade do ajuste pode ser verificada por meio do teste KS [17] [18], encontrado em A.2.

O modelo de simulação que será utilizado é o implementado no Software Venthor, descrito em 3.4. O Venthor foi desenvolvido na UEPG-Universidade Estadual de Ponta Grossa e é utilizado para pesquisa na Universidade e também auxilia instituições parceiras.

O Venthor, através de análise estatística com o uso do teste KS, aponta o melhor modelo

¹No decorrer do trabalho, este será chamado também de 'teste KS'.

entre as distribuições Weibull, Rayleigh, Beta e Gama, para cada conjunto de dados horários empíricos, e assim poder simular uma série de dados pelo período de tempo solicitado, possibilitando um estudo mais detalhado do local, fornecendo dados para o uso desde o plantio até a colheita.

2.2 Computação Paralela e Distribuída

Segundo Coulouris et al.[19], um sistema distribuído é uma coleção de computadores autônomos interligados através de uma rede de computadores e equipados com software que permita o compartilhamento dos recursos do sistema: hardware, software e dados. E, segundo Lorin[20], computação paralela é uma forma de computação em que vários cálculos são realizados simultaneamente, operando sob o princípio de que grandes problemas geralmente podem ser divididos em problemas menores, que então são resolvidos paralelamente.

O paralelismo pode ser projetado pelo programador, onde o mesmo é responsável por fornecer todas as características necessárias para o correto funcionamento, como concorrência, troca de mensagens entre processos e tratamento de erros. Essa é a forma mais aplicada, mas o paralelismo pode também ser definido pelo compilador. Segundo Shen e Mikko[21], a habilidade de um compilador realizar paralelismo automático de um programa sequencial é um dos assuntos mais estudados no campo de computação paralela. Entretanto, apesar de décadas de trabalho por pesquisadores de compiladores, pouco sucesso se teve na área. Linguagens de programação paralelas de amplo uso continuam explícitas, isto é, requerendo a programação paralela propriamente dita; ou no máximo parcialmente implícitas, em que o programador fornece ao compilador diretivas para incorporar o paralelismo.

Apesar das dificuldades na programação paralela e distribuída, existem arquiteturas e ferramentas bem difundidas que permitem tirar maior proveito dos recursos disponíveis nos computadores atuais (multiprocessadores e multinúcleos), mas não só deles, também de computadores mononúcleos através do acoplamento em redes físicas ou lógicas. Como exemplo dessas arquiteturas pode-se citar os *clusters* (podendo ser usados com ferramentas como MPI [22] [23] e PVM [24] [25]) e as grades computacionais que podem utilizar ferramentas que implementam redes P2P, RMI (*Remote Method Invocation*) ou o protocolo HTTP (*request e response*).

Um *cluster* é um conjunto de computadores, chamados nós, interligados por uma rede física local de alto desempenho e tem memória distribuída [26]. É basicamente de três tipos: [1] Alta Disponibilidade, onde são construídos para fornecer disponibilidade de recursos e serviços de forma ininterrupta. Bastante utilizados para bases de dados de missão crítica e servidores de

arquivos e aplicações. [2] Balanceamento de Carga, onde o tráfego é dividido entre os participantes do *cluster*. Bastante utilizados em *web farms* (fazendas de servidores web). [3] Combinação de Alta Disponibilidade com Balanceamento de Carga, combinando disponibilidade de recursos e escalabilidade de serviços. Bastante utilizados em servidores de web, mail, FTP (Protocolo de Transferência de Arquivos) entre outros [27]. Pontos positivos dos *clusters* são: segurança, velocidade, desempenho e o fato de parecer uma única máquina. Pode-se destacar como ponto negativo dos *clusters* a limitação física, impedindo uma expansão em larga escala, caso seja necessário, e o alto grau de homogeneidade entre as máquinas locais.

O surgimento de grades computacionais nasceu da comunidade de Processamento de Alto Desempenho (PAD). O conceito foi apresentado pelos pesquisadores Ian Foster e Carl Kesselman, sendo composto por uma infra-estrutura de hardware e software que permite-nos acesso a grandes capacidades computacionais geograficamente distribuídas, de forma confiável, consistente, econômica e persistente [27]. Grades Computacionais podem ser descritas resumidamente como: computadores em diferentes localidades, interligados logicamente por diferentes redes, disponibilizando computação de alto desempenho e ambiente colaborativo, permitindo compartilhar diferentes recursos como: ciclos ociosos de processador, memória, discos e sensores.

Desde a década de 1990 os sistemas Desktop Grid tais como: SETI@Home [28], SZ-TAKI [29], XtremWeb [30], Globus [31], Legion [32], têm se destacado como poderosos sistemas de computação distribuída. Eles oferecem poder computacional a uma fração do custo de um supercomputador. Como exemplo podemos citar o SETI@Home, que tem por objetivo descobrir padrões de comunicação que evidenciem a existência de vida extraterrestre a partir da análise de ondas eletromagnéticas capturadas por um conjunto de rádio-telescópios. Colaboradores de todo o mundo instalam um aplicativo gerenciador do projeto, e fornecem voluntariamente os ciclos de processador ociosos de suas máquinas [28].

O poder de processamento oferecido pelos sistemas *Desktop Grid*, tem sido utilizado por aplicativos de diversos domínios científicos. Sistemas em grade permitem uma execução mais rápida de um processo, pelo uso de paralelismo. Essa tecnologia possibilita descobertas científicas que não seriam possíveis num sistema convencional. Alguns projetos *desktop grids* de relevância, aplicados à agricultura e agropecuária, descritos por Calin et al.[33], são apontados na sequência.

Na Universidade do Estado do Colorado, nos Estados Unidos, existe uma grade computacional para um sistema nacional de identificação e rastreamento animal. Essa grade permite aos pesquisadores rastrear o deslocamento de um animal em segundos ou minutos, permitindo

também o controle de doenças como, por exemplo, a doença da vaca louca (*encefalopatia espongiforme bovina*), fornecendo dados desses animais desde seu nascimento até sua morte.

Outro exemplo de sistema em grade com propósito agrícola, é a parceria entre o *US Cornell Theory Center* e o *Cornell's College of Agriculture and Life Sciences*, que utilizam sua infra-estrutura para desenvolver tecnologias de suporte aos agricultores, tendo como principais domínios envolvidos na agricultura de precisão: o conhecimento sobre o solo e as culturas, economia, modelagem climática e mineração de dados.

No Japão, o Centro Nacional de Pesquisa Agrícola de Tsukuba desenvolve um projeto chamado "Grade para Suporte à Decisão Agrícola", que combina dados de diferentes bases espalhadas pelo país como: bases de clima, solo, culturas, lavouras e mercado. A intenção é criar uma ferramenta de suporte à decisão, combinando esses dados que poderão ser acessados por outros servidores remotos distribuídos por diversas regiões.

O Instituto Europeu de Bioinformática, através de uma grade Européia, disponibiliza recursos para pesquisas nas áreas da biologia molecular, genética, agricultura, biotecnologia, química e indústria farmacêutica. Os pesquisadores envolvidos estão distribuídos geograficamente e trabalham sobre plataformas diferentes. A grade compartilha diversos recursos, bases de dados diferentes e serviços de informação.

Em 2006, quatro faculdades e o Instituto de pesquisa em Iasi na Romênia iniciaram um projeto chamado GRAI - Grade Acadêmica para Aplicações Complexas. Essa grade compartilha os recursos entre os participantes com fins de pesquisa e acadêmico. Cada um dos cinco participantes será um nó da grade, com um servidor de alto desempenho e estações interligadas a ele. Os participantes poderão desenvolver serviços e aplicações para a grade.

3 *Materiais e Métodos*

3.1 Considerações Iniciais

Após a análise de viabilidade das tecnologias anteriormente citadas, a opção escolhida foi a de grades computacionais utilizando redes P2P¹. A justificativa para esta escolha se deve ao fato de que grades associadas à protocolos P2P são expansíveis, independentes de plataforma, de homogeneidade entre o hardware dos participantes e independência de localização, já que trabalham com o conceito de rede lógica e não física. Trabalhar com outras tecnologias em grade ou com *clusters* associados com MPI [23] ou PVM [25] também seriam alternativas interessantes e viáveis, mas que limitariam o alcance deste projeto, principalmente se tratando do crescimento no volume de dados no futuro.

Diversos softwares simuladores ainda utilizam dados diários médios e não dados horários, que geram um volume 24 vezes maior de dados. As coletas realizadas nas estações meteorológicas já são coletadas a cada 5 minutos, o que geraria um volume 12 vezes maior que os dados horários e 288 vezes maior que os valores médios diários. A seguir são descritas as características principais da Tecnologia JXTA e do *Framework* P2PComp, escolhidos como ferramentas para este estudo.

3.2 JXTA

A seguir são descritas as características principais do JXTA, sendo que maiores detalhes podem ser encontrados em Gradecki[34] e Wilson[35].

JXTA é um conjunto de protocolos abertos e generalizados para aplicações baseadas no Modelo P2P, de modo a permitir que quaisquer dispositivos, tais como: sensores, celulares, PDAs, notebooks, desktops, servidores e supercomputadores possam se comunicar e comparti-

¹Peer-to-peer (tradução literal do inglês de "par-a-par" ou "entre pares"; tradução livre: ponto a ponto) é uma arquitetura de sistemas distribuídos caracterizada pela descentralização das funções na rede, onde cada elemento realiza tanto funções de servidor quanto de cliente.

lhar recursos mutuamente.

O nome JXTA não é uma sigla, mas um resumo da palavra *juxtapose*. O propósito desta plataforma é fornecer um conjunto de protocolos que permitam que qualquer pessoa possa desenvolver sua aplicação P2P. A maioria das aplicações P2P existentes é construída sobre protocolos proprietários e isso limita o alcance delas.

Com a plataforma JXTA é possível:

- Permitir o compartilhamento de recursos entre dois *Peers* sem um servidor central (*Peers* são unidades numa rede P2P, serão mais bem definidos adiante);
- Usar milhares de ciclos do processador de computadores remotos para resolver problemas complexos;
- Expor serviços Web para serem utilizados por outros *Peers*.

A implementação de referência estabelece um conjunto de seis protocolos que têm por objetivo cumprir com todas as tarefas básicas que uma aplicação P2P comum deve ser capaz de tratar, eles são assim nomeados: *Peer Discovery Protocol* (PDP), *Peer Resolver Protocol* (PRP), *Rendezvous Protocol* (RVP), *Peer Information Protocol* (PIP), *Pipe Binding Protocol* (PBP), *Endpoint Routing Protocol* (ERP).

Uma característica da plataforma é a possibilidade de localizar recursos em máquinas remotas, sem que se tenha nenhuma informação adicional sobre a máquina além de seu PeerID², que é publicado e replicado entre os Super-Peers.

Peers

Um *Peer* é qualquer elemento de uma rede que implemente um ou mais dos protocolos JXTA. Esses elementos de rede podem ser sensores, telefones celulares, computadores pessoais, servidores ou supercomputadores. Cada um destes elementos pode responder por mais de uma conexão com a rede, podendo representar mais de um *Peer* dentro da rede JXTA. Isso ocorre porque têm associado a si mais de um valor de *Endpoint*³, que é utilizado para estabelecer a conexão direta ponto a ponto. Os *Peers* são identificados de forma única por seu PeerID.

As conexões diretas nem sempre são possíveis. Quando isso ocorre, um *Peer* deve localizar um Super-Peer que implemente protocolos específicos para fazer o roteamento das mensagens.

²Identificador do *Peer* conforme descrito na Seção 3.2.

³Cada elemento de rede pode ter associado a si, um ou mais *Peers*, devendo assim, ter sua localização, seu endereço na rede, identificado unicamente para evitar conflitos.

Os *Peers* podem ser de três tipos básicos, a saber:

- *Minimal-Edge*: são *Peers* que não implementam todos os serviços básicos da plataforma. Eles precisam da ajuda de outros *Peers* para participar completamente da rede P2P;
- *Full-Edge*: implementam todos os serviços padrão da plataforma e podem participar de todos os protocolos;
- *Super-Peer*: fornecem protocolos para auxiliar na comunicação e localização de recursos e podem ser:
 - *Relay*: usado para armazenar e procurar mensagens entre *Peers* que não conseguem se conectar diretamente com outros *Peers* devido à presença de *firewalls* ou servidores de NAT (*Network Address Translation*) entre eles;
 - *Rendezvous*: indexa anúncios e auxilia na localização física destes anúncios, ou seja, um diretório de cache em algum local da rede;
 - *Proxy*: auxilia os *Peers* do tipo *Minimal-Edge* a obterem todas as funcionalidades do JXTA.

Peer Groups

Um *Peer Group* é o agrupamento de *Peers* com um conjunto comum de serviços e interesses. Cada *Peer* pode pertencer a mais de um *Peer Group* simultaneamente. Por exemplo, assim que um *Peer* é criado, por padrão ele já pertence tanto ao *World Peer Group* quanto ao *Net Peer Group*. O ingresso de um *Peer* num *Peer Group* específico depende da política de membros que este impõe, podendo ser desde livre acesso até o acesso com exigência de autenticação, utilizando infra-estrutura de chaves públicas.

Network Services

Network Services são a forma pela qual os *Peers* cooperam para publicar, descobrir e invocar serviços da rede JXTA. Estes serviços são descobertos utilizando o *Peer Discovery Protocol* e são de dois tipos básicos: *Peer Services*: são serviços disponíveis somente no *Peer* que está publicando o serviço. Se ele falhar o serviço também falha. *Peer Group Services*: são serviços disponibilizados e rodando em múltiplos membros de um *Peer Group*. Quando um *Peer* falha o serviço permanece funcionando.

Como exemplos de *Peer Group Services* são elencados aqueles que são padrão da plataforma, a saber:

- *Endpoint Service*: que trata do envio e do recebimento de mensagens entre *Peers*;
- *Resolver Service*: para realizar envio de consultas genéricas a outros *Peers*;
- *Discovery Service*: usado para fazer pesquisa por recursos num *Peer Group*;
- *Membership Service*: útil para estabelecer um mecanismo de identificação seguro entre os *Peers*;
- *Access Service*: usado para validar requisições feitas por um *Peer* a outro;
- *Pipe Service*: necessário para criar um gerenciador de conexões de *Pipe* entre dois membros de um *Peer Group*; e
- *Monitoring Service*: permitir que um *Peer* monitore outro num mesmo *Peer Group*;

Modules

Um *Module* é qualquer trecho de código fonte que possa ser publicado, carregado e executado pela plataforma JXTA. Podem ser usados para implementar serviços e/ou o transporte de mensagens entre *Peers*. Por meio dos modules um *Peer* pode instanciar uma função ou serviço que roda em uma máquina remota.

Os *Modules* são publicados a partir de três anúncios específicos:

- *Module Class Advertisement*: apenas anuncia a existência de uma classe de *Module*;
- *Module Specification Advertisement*: tem por objetivo identificar, unicamente, um conjunto de protocolos compatíveis. Pode conter o anúncio para o Pipe⁴ utilizado na troca de informações com o *Module*;
- *Module Implementation Advertisement*: é o responsável final por informar as características de uma determinada implementação de um *Module*, incluindo sua localização.

Cabe aqui fazer duas observações, a primeira é que para todo *Module Class Advertisement* pode haver um ou mais *Module Specification Advertisement*, cada um especificando uma versão diferente do *Module*. A segunda, é que para cada *Module Specification Advertisement* pode existir um ou mais *Module Implementation Advertisements* contendo localizações diferentes do *Module*, por exemplo.

⁴Canal virtual de comunicação para troca de mensagens e transferência de dados, descrito na Seção 3.2

Messages

As mensagens são unidades básicas de dados trocadas entre *Peers* pelo uso dos serviços de *Endpoint*, *Pipe* ou *JXTASockets*. Os canais de comunicação para troca de dados mais comuns entre dois *Peers* são o *JXTASocket* e o *JXTABiDiPipe*.

O uso de mensagens XML torna a aplicação dos protocolos mais geral e, devido às tags⁵, cada *Peer* é livre para realizar a troca de dados da melhor maneira possível.

Existem dois formatos básicos para as mensagens: arquivo XML e formato binário. O JXTA-JXSE trabalha com o formato binário. Não há grandes problemas com relação a isso, pois a estrutura das mensagens utiliza a forma de criação normal de qualquer arquivo XML.

As mensagens são estruturadas a partir de Elementos de Mensagem, que contém a mensagem propriamente dita. Para cada elemento de mensagem pode ser adicionado um namespace de modo que múltiplas camadas do protocolo possam utilizar a mesma mensagem sem que ocorram colisões. Os Elementos de Mensagem podem transferir *Strings*, *byte arrays*, *documents*, *input streams*, etc.

Antes de serem enviados, os dados devem ser transformados em linhas de bytes. Essa abordagem é interessante porque permite facilmente que objetos sejam serializados e adicionados como elementos de mensagem, podendo ser novamente convertidos no objeto original quando chegarem ao seu destino.

Os elementos de mensagem possuem um nome, um *MIME Media Type*, que quando possui o valor (*Application/Octet-Stream*) significa transferência de dados binários, e o conteúdo que é a sequência de bits. Uma assinatura criptográfica opcional também pode ser adicionada, ou ainda é possível utilizar um mecanismo de hash.

Os dados contidos num Message Element podem ser acessados de quatro modos distintos:

- Como `java.io.InputStream`, na entrada de dados;
- No envio, como `java.io.OutputStream`;
- Como `java.lang.String` no envio e recebimento; ou ainda
- Como `bytearray`.

Cabe destacar que há uma taxa de aproximadamente 50 bytes de *overhead*⁶ na transmissão

⁵Etiquetas de identificação dos elementos de um arquivo XML.

⁶Sobrecarga no tráfego da rede

de cada Elemento de Mensagem adicionado a um *Message*, de modo que, quando se planeja uma mensagem é interessante projetá-la com poucos elementos.

Pipes

Os *pipes* são mecanismos para troca de mensagens e transferência de dados assíncronos, unidirecionais e não confiáveis (sem garantia de entrega). Pode-se definir também como canais virtuais de comunicação que podem interconectar *Peers*. Estes podem estar impossibilitados de comunicarem-se diretamente, nesse caso cria-se uma ligação intermediada. Por estes canais é possível transferir dados que incluem XML, arquivos com *Hypertext Markup Language* (HTML), imagens, músicas, códigos binários, strings e objetos Java.

Existem basicamente cinco tipos de Pipes:

Unicast Pipes - São canais onde a comunicação ocorre numa única direção. Se for necessário transferir dados em duas direções, ambos os *Peers* deverão publicar anúncios de Pipes diferentes. São construídos sobre o protocolo TCP/IP (que possui garantia da entrega), contudo, a implementação de referência não leva este fator em consideração e, assim os Unicast Pipes são não-confiáveis.

Unicast Secure Pipes - É um tipo de *Pipe* no qual os dados a serem trocados pelos *Peers* são criptografados, utilizando o protocolo Transport Layer Security (TLS) ou o Secure Socket Layer (SSL).

Propagate Pipes - São utilizados quando ocorre o caso de transferir dados a uma grande quantidade de *Peers* ao mesmo tempo, ou seja, realizar uma transmissão similar a um *broadcast*⁷.

Bidirecional Pipes (BiDiPipe) - Implementam mecanismos de comunicação onde os *Peers* podem trocar dados pelo *Pipe* em qualquer direção. Este tipo de *Pipe* não está presente na definição do *Pipe Binding Protocol*. Ele é fruto da implementação de referência para a linguagem Java.

Reliable Pipes - São criados sobre o mecanismo de *BiDiPipe* e a idéia básica é tornar confiável a transmissão sobre o *Unicast Pipe*, que não oferece garantia de entrega. Os *Reliable Pipes* não permitem o uso de *listeners*⁸, o sistema aguarda num laço a aceitação de conexão e envio de dados.

JXTASocket - É um *Pipe* bidirecional que implementa a interface `java.net.Socket` e possui

⁷ Espalhamento, transmissão para todos os participantes.

⁸ Ouvintes, que ficam aguardando uma mensagem

um comportamento similar aos java sockets comuns.

Advertisements

Recursos como *Peers*, *Peer Groups* e *Services* são representados com advertisements. Estes são metadados estruturados em documentos XML. Os advertisements são utilizados para descobrir e publicar a existência dos recursos de um *Peer* e possuem um *timelife*(tempo de vida) que habilita a sua exclusão automática quando do seu término, eliminando um gerenciamento centralizado dos anúncios.

Na sequência são apresentados os anúncios padrão da plataforma:

- PeerAdvertisement (Anuncia a existência de um Peer);
- PeerGroupAdvertisement (Anuncia a existência de um grupo de Peer);
- PipeAdvertisement (Anuncia a existência de um canal de comunicação);
- ModuleClassAdvertisement (Anuncia a existência de uma classe de módulo);
- ModuleSpecAdvertisement (Anuncia a existência de uma especificação de módulo);
- ModuleImplAdvertisement (Anuncia a existência de uma implementação de módulo);
- RendezvousAdvertisement (Anuncia a existência de um Rendezvous); e
- PeerInfoAdvertisement (Anuncia informações sobre um Peer).

IDs (Identificadores)

Os recursos da rede JXTA são identificados com IDs que são únicos. As entidades que possuem IDs são: *Peers*, *Peer Groups*, *Pipes*, *Contents*, *Module Classes* e *Module Specifications*. Os IDs são expressos a partir de Uniform Resource Names (URNs), sendo que todo ID possui um formato bem definido que permite saber como ele foi gerado e como os programas devem manipulá-lo. Nos exemplos abaixo foi usado o formato (uuid), mas poderia ter sido usado o formato (cbid).

São exemplos de IDs:

- PeerID

urn:jxta:uuid-BBECFF86ECDE4F808C25850DD6FC323B54DC76781CCF4C63B05FB2C7778A572903

- PeerGroupID

urn:jxta:uuid-4118A274A41F4560A0FD9CAA84F8EF3A02

- PipeID

urn:jxta:uuid-CB4700ACF6904756916DA86F909CC1D65875A24860234EA588E4936524C450FB04

- ModuleClassID

urn:jxta:uuid-F86F62A695304098BEF896638496E2D805

- ModuleSpecID

urn:jxta:uuid-70B8FC594C494B89886B817C95EB257993CF4582758F4E329EF0BF6ECAB70ABE06

Protocolos

São seis os protocolos que definem a Plataforma JXTA, a saber:

- *Peer Discovery Protocol* (PDP): Usado pelos *Peers* para publicar seus próprios recursos, e descobrir os recursos de outros *Peers*. Cada recurso de um *Peer* é descrito e publicado com um anúncio;
- *Peer Information Protocol* (PIP): utilizado pelos *Peers* para obter informações sobre o status de outros *Peers*;
- *Peer Resolver Protocol* (PRP): habilita os *Peers* a enviarem consultas genéricas a um ou mais *Peers* e receber respostas (ou múltiplas respostas) para a consulta. Este protocolo permite que os *Peer Services* definam e troquem qualquer informação arbitrária que necessitem;
- *Pipe Binding Protocol* (PBP): utilizado para estabelecer um canal de comunicação virtual entre um ou mais *Peers*. É utilizado para fazer a ligação entre dois *Endpoints*;
- *Endpoint Routing Protocol* (ERP): utilizado para encontrar rotas para as portas de destino em outros *Peers*. A informação sobre o roteamento inclui uma sequência ordenada de PeerIDs para retransmissão de modo a fazer com que a mensagem chegue ao seu destino;
- *Rendezvous Protocol* (RVP): usado por *Edge-Peers* para obter recursos, propagar mensagens e anunciar recursos locais. Os Rendezvous se organizam por conta própria utilizando este protocolo e compartilham DHTs (*Distributed Hash Tables*).

Arquitetura de Rede

Um *Peer* do tipo *Minimal Edge* pode enviar e receber mensagens, mas não armazena anúncios ou roteia mensagens para outros *Peers*. Por outro lado, um *Peer* do tipo *Full-Edge* pode enviar e receber mensagens e ainda armazenar anúncios. Este *Peer* replica a requisição por descoberta com a informação encontrada nos seus anúncios armazenados, mas não retransmite qualquer requisição por descoberta. Ou seja, ele apenas responde à requisição.

O *Peer Rendezvous* ajuda outros *Peers* com propagação de mensagens de descoberta de anúncios e rotas, e o mais importante, mantém um mapa da existência de outros *Peers* de infraestrutura.

Cada *Peer Group* mantém seu próprio conjunto de *Peers Rendezvous* e pode ter quantos forem necessários. *Peers Edge* podem fazer consultas aos *Rendezvous* e estes podem repassar as requisições que não conseguirem responder a outros *Peers Rendezvous* usando a topologia DHT (*Distributed Hash Table*).

Um *Peer* de fora de um *firewall* deve estar apto a ver ou usar um *Peer* detrás do *firewall* e precisa enviar um *HTTP Request* a este. Muitos *firewalls* só permitem o retorno de *HTTP Responses*. Para resolver isso, o *Peer Relay* ajuda *Peers* não endereçáveis, ou seja, aqueles que estão ocultos sob *firewalls* e NAT (*Network Address Translation*), a enviar e receber mensagens.

Para que *Peers* que estejam nesta condição possam se comunicar, é necessário que:

- Ao menos um *Peer* de dentro de um *firewall* deve estar ciente da existência de pelo menos um *Peer* fora do *firewall*;
- Os *Peers* de dentro e de fora devem oferecer suporte a protocolos de transporte comuns (HTTP, TCP);
- O *firewall* deve ao menos permitir saída HTTP ou conexões TCP;

Usando o Rendezvous Protocol

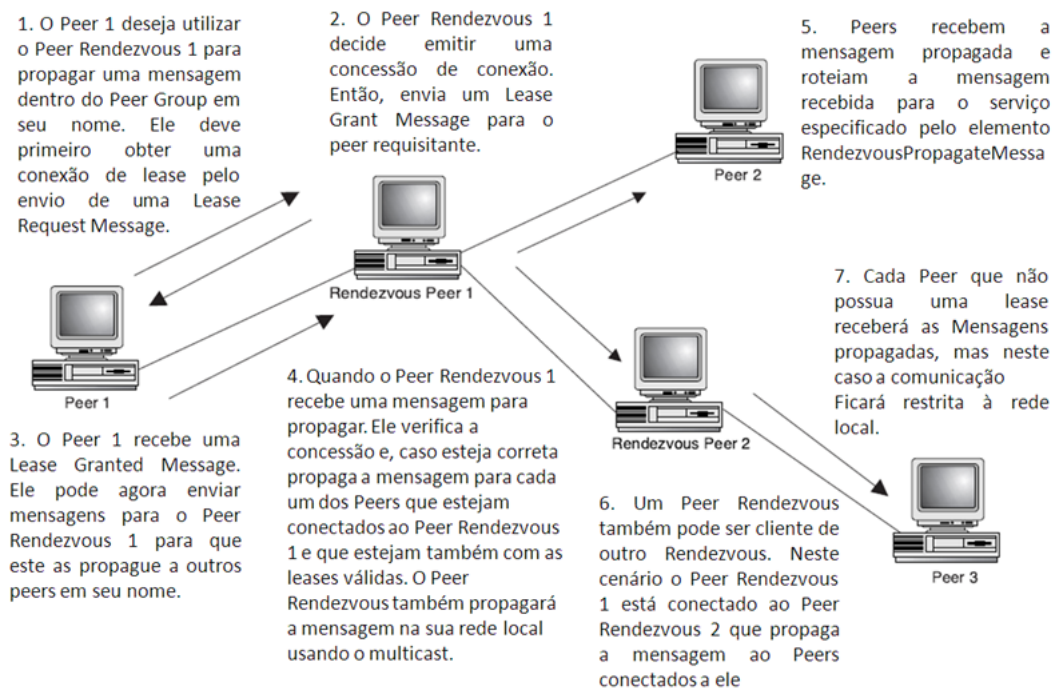
Um *Peer Rendezvous* é utilizado para permitir que um *Peer* simples envie mensagens a outros *Peers* localizados fora da sua rede. Esta funcionalidade independe do protocolo de rede utilizado.

Antes que um *Peer* possa propagar mensagens, ele deve se conectar a um *PeerRendezvous* e obter uma *lease* (concessão). Esta concessão especifica uma quantidade de tempo que o *Peer* requerente de conexão no *Rendezvous* estará conectado antes que tenha que requerer uma nova

lease. Para manipular as requisições e prover esta funcionalidade, o *Rendezvous Protocol* define três formatos de mensagens:

- *Lease Request Message* - Uma mensagem usada para um *Peer* requerer uma concessão no *Peer Rendezvous*.
- *Lease Granted Message* - Uma mensagem usada pelo *Rendezvous* para aprovar um *Lease Request Message* e permitir o tamanho da concessão.
- *Lease Cancel Message* - Uma mensagem usada por um *Peer* para desconectar de um *Rendezvous Peer*.

Figura 3.1: Comunicação entre *Peers* que não estejam numa mesma rede local.



Fonte: Wilson[35] adaptado.

Estas mensagens não são, especificamente, definidas em termos de XML. Em vez disso, elas são definidas em termos de *Message Elements*. Como elementos de mensagem consistem de um nome e do conteúdo do elemento, eles podem ser aninhados. Para se conectar a um *Peer Rendezvous* um *Peer* utiliza a sequência de mensagens mostrada na Figura 3.1.

3.3 O Framework P2PComp

O *Framework P2PComp* foi desenvolvido por Senger, Souza e Foltran Junior[36]. Ele foi construído utilizando a implementação JXSE, na linguagem JAVA, para o padrão JXTA.

O *Framework*⁹ P2PComp aplica os princípios do Modelo P2P puro, ou seja, implementa uma rede de compartilhamento de recursos totalmente descentralizada, onde cada *Peer* pode atuar como cliente ou servidor. Permite também a construção de aplicações paralelas de acordo com o SPMD (*Single Process, Multiple Data*) para a linguagem de programação Java, e oferece primitivas para comunicação entre tarefas em execução nos *Peers*. O P2PComp foi desenvolvido utilizando a biblioteca JXSE, a qual implementa o JXTA P2P para a linguagem Java.

O formato utilizado pelo P2PComp para criação de novas tarefas, segue um padrão bem próximo do utilizado no MPI [23] e no PVM [25], como exibido no código a seguir, que mostra a estrutura básica de uma tarefa.

```
1 public class HelloWorld extends Task{
2     public void run(){
3         if(getRank() == 0){ // Tarefa mestre
4             size = getSize();
5             i = 1;
6             while(i < size){
7                 Message msg = recv(-1, -1);
8                 i++;
9             }
10            Vector results = new Vector();
11            results.add(i);
12            sendResults(results);
13        }
14        else{ // Tarefa trabalhadora
15            Vector message = new Vector();
16            message.add(new String("Hello world from "+ getRank()));
17            send(0, 1, message);
18        }
19    }
20 }
```

Detalhes do Código

Como pode ser visto na linha 1, a tarefa de nome 'HelloWorld' estende a classe 'Task' (classe do P2PComp usada para criação de novas tarefas) que por sua vez implementa a interface *Runnable*¹⁰. A classe Task tem como principal funcionalidade a comunicação entre processos, enviando e recebendo mensagens. Também contém o método para iniciar a execução de uma

⁹Framework é um conjunto de classes e algoritmos extensíveis para fornecer funcionalidades genéricas ao programador.

¹⁰Utilizada na implementação de Threads em Java

*Thread*¹¹ para a tarefa.

Na linha 2 é sobrescrito o método *run* (obrigatório por se tratar de uma *Thread* com a interface *Runnable*) o qual contém o algoritmo da tarefa e que será executado na *Thread*.

Na linha 3 é realizado um teste (*if* - teste condicional) para verificação do identificador do processo, que sendo 0 (zero) executará o código Mestre e se for diferente de 0 (zero) executará o código Trabalhador. O P2PComp distribui aleatoriamente os identificadores de processo, sendo que qualquer processo executando nos *Peers* pode receber o identificador 0 (zero) ou outro, ficando o de número 0 (zero) responsável pelo gerenciamento, distribuindo a execução para os participantes e recebendo os resultados após o término.

Na linha 4, a variável 'size' recebe através do método 'getSize' o total de processos em execução.

Na linha 6 é criado um laço 'while' que ficar aguardando o retorno de mensagens dos processos trabalhadores. O recebimento é feito através do método 'recv'. Após o recebimento de todas as mensagens, o processo 0 (zero) envia o resultado para quem solicitou a execução através do método 'sendResults', que nesse caso envia o total de mensagens recebidas.

A partir da linha 14 está o código referente aos demais processos (trabalhadores), que adicionam uma frase contendo 'Hello world' e o identificador de seu processo, enviando a frase em seguida através do método 'send'.

Existem no P2PComp dois tipos de *Peers*: um Super-Peer (Rendezvous) e *Peers* de infraestrutura e controle (Edge). O Rendezvous tem a função de armazenar anúncios e informações sobre a localização dos demais *Peers*, conforme descrito na Seção 3.2. Os *Peers Edge* serão os responsáveis pelo gerenciamento e processamento das tarefas contidas no P2PComp, podendo ser distribuídos em: um *Peer* executor (de onde será iniciada a tarefa), um *Peer* controlador (que será responsável pelo gerenciamento da tarefa) e outros *Peers* trabalhadores (que executam o processamento). Esses *Peers* podem estar todos numa única máquina, ou podem estar distribuídos entre duas ou mais máquinas diferentes (podendo chegar ao nível 1/1, máquina/Peer).

3.4 O Software Venthor

O Venthor, em sua versão original, foi desenvolvido utilizando a Linguagem de Programação C# e o Sistema Gerenciador de Banco de Dados SQL Server 2005 da Microsoft, para a plataforma Windows. Seu método foi criado pelo prof. Dr. Jorim Sousa das Virgens Filho e

¹¹É a divisão de um processo em duas ou mais partes para que sejam executados concorrentemente.

desenvolvido pelo grupo de pesquisa comandado por ele na Universidade Estadual de Ponta Grossa-PR.

O modelo utilizado é composto por quatro partes principais que são comentadas a seguir, sendo que o Software Venthor apresenta ainda outras características e funcionalidades, tais como: interface gráfica, exibição e análise de dados, gráficos, relatórios, entre outras, que não serão tratadas aqui, devido a esse trabalho focar apenas na sua parte crítica e fundamental: a simulação dos dados horários de velocidade e direção do vento.

(1) Cálculo da Tabela de Frequência - Onde são tratados os dados obtidos a partir da base de dados empírica, sendo ordenados e distribuídos conforme as classes da tabela e calculados sua média, desvio padrão, valor máximo, valor mínimo, amplitude total, amplitude da classe e limites inferior e superior. Também são definidos os valores dos parâmetros para o cálculo das distribuições de probabilidade.

(2) Cálculo das Distribuições de Probabilidade - Onde são executados os cálculos para cada uma das distribuições com os parâmetros obtidos através da Tabela de Frequência e aplicado o teste KS para verificação de qual delas representa melhor os dados empíricos.

(3) Cálculo da Frequência da Direção - Onde é criada uma Tabela de Frequência para a direção do vento, e distribuídos os valores entre os oito pontos cardeais e colaterais (norte, nordeste, leste, sudeste, sul, sudoeste, oeste, noroeste).

(4) Execução da Simulação - Aqui são simulados os dados para o período de tempo solicitado, tanto para a velocidade como para a direção do vento, aplicando para a velocidade a função inversa da distribuição que melhor aderiu aos dados empíricos e para a direção, utilizando frequência relativa simples.

No Anexo A, são descritas as características principais e as fórmulas que compõem as distribuições utilizadas pelo modelo na simulação da velocidade, da direção e do teste KS, conforme descrito em Virgens Filho e Leite[5].

3.4.1 Detalhes do Simulador

O Venthor executa o método de simulação de forma sequencial, criando algumas *threads* durante a execução, mas utilizando apenas uma máquina. O modelo realiza a simulação de um conjunto de dados de cada vez, mas de forma ordenada sendo executado várias vezes, do primeiro até o último conjunto. A quantidade de conjuntos que serão passados ao método é definida pelo tipo de coleta realizada na estação meteorológica que forneceu os dados. Nessa

versão do Venthor são utilizadas bases de dados coletadas de hora em hora, contendo 24 (vinte e quatro) valores horários para cada dia do ano, sendo que cada grupo representará uma hora/mês.

Ao iniciar a execução de uma simulação, o software faz a leitura da base de dados empírica e o agrupamento dos dados, separando-os por mês (12) e por hora (24), ou seja, agrupa todos os dados da hora 00:00 (zero) do mês de janeiro (dia 01, 02 ... 31), da hora 01:00 (um) do mês de janeiro, até a hora 23:00 (vinte e três), repetindo o agrupamento para cada um dos 12 (doze) meses do ano. Também é realizado o agrupamento de todos os anos existentes na base como, por exemplo, uma base de 10 anos, a qual terá 310 (trezentos e dez) coletas para a hora 00:00 do mês de janeiro (31 dias x 10 anos), 300 coletas para a hora 00:00 do mês de abril, seguindo o mesmo padrão para todos os meses.

A cada chamada do método, é passado como parâmetro um grupo de dados hora/mês, como citado acima. O método recebe os dados e os distribui pela tabela de frequência para obter valores como: média, desvio padrão, valor máximo, valor mínimo, amplitude total, amplitude da classe e limites inferior e superior. Na sequência é executada a função para cada uma das quatro distribuições de probabilidade utilizando os parâmetros obtidos na Tabela de Frequência e aplicado o teste KS sobre o resultado para verificar qual das distribuições melhor representa os dados empíricos.

Para o teste KS, deve ser definido um valor de tolerância que limitará a aceitação ou rejeição da distribuição. O Venthor utiliza o valor de 5%, sendo escolhida a que tiver o menor valor dentre as que passarem no teste.

Após ter definida a distribuição, é executada a função inversa da distribuição que retornará os valores simulados de velocidade do vento pelo período de tempo definido pelo solicitante. Também é executada a simulação da direção do vento através de frequência relativa simples para cada mês, utilizando os pontos cardeais: norte, sul, leste, oeste e também os colaterais: nordeste, sudoeste, sudeste e noroeste, conforme descrito em A.3.

Ao término da execução das partes, o software faz a junção dos dados simulados e a gravação numa nova base, que poderá ser exportada para uso posterior no Venthor ou utilizada em outros aplicativos.

3.5 Recursos Utilizados nos Experimentos

Para elaboração das atividades descritas nesse trabalho, foi utilizado o LCC-Laboratório de Computação Científica, localizado na UEPG, bloco L do campus de Uvaranas.

Foram utilizados computadores desktop com a seguinte configuração:

- Processador Intel core I7 com quatro núcleos de 3,4 Ghz cada;
- Memória RAM de 4,0 GB DDR3;
- Disco rígido de 500 GB 7200 rpm;
- Monitor de 15,6 polegadas;
- Sistema Operacional Windows 7 na execução do Venthor original;
- Sistema Operacional Linux Slackware 13.37 na execução das demais versões;

Simultaneamente foram utilizadas no máximo, 07 máquinas como as descritas acima.

4 *Resultados*

4.1 Migração para a Linguagem Java

Antes de maiores detalhes, vale ressaltar que essa versão também executa o método de simulação de forma sequencial, como na versão original, passando ordenadamente as partes a serem simuladas. Esta etapa tem como objetivo principal a correta execução do modelo com a linguagem java, já que essa versão será a base para o simulador paralelizado.

Após o estudo sobre o funcionamento do método, iniciou-se a extração e transcrição do código original em C# para o Java. Esse procedimento foi necessário devido à necessidade de adaptação ao *Framework* P2PComp e aos protocolos do JXTA. Outro motivo importante para migração foi a portabilidade, permitindo a utilização do mesmo em sistemas operacionais Linux, por exemplo, o qual é utilizado em muitas instituições de ensino e laboratórios de pesquisa.

O projeto Java é agrupado em duas classes principais: A *VenthorManager*, responsável pelo gerenciamento da simulação e a classe *VenthorSimulation*, responsável pela simulação.

Na classe *VenthorManager* foram criados métodos para realizar a leitura da base de dados empírica, o agrupamento dos dados por hora/mês, a passagem desses grupos, por parâmetro, ao método de simulação (classe *VenthorSimulation*), o recebimento dos conjuntos já simulados, a junção das partes recebidas e a gravação da nova base simulada. A classe *VenthorManager* utiliza vetores multidimensionais para armazenar os dados lidos da base empírica e também no recebimento das partes simuladas, que ao final do processo serão gravadas na nova base.

Na classe *VenthorSimulation* foram agrupados os métodos e classes vindos do modelo original, através de subclasses (*innerclasses*). Essas classes são compostas pela tabela de frequência, pelos cálculos de cada uma das distribuições de probabilidade, pelo teste KS, pela simulação da velocidade e pela simulação da direção do vento. A classe *VenthorSimulation* executa a simulação para cada conjunto de dados, separadamente, recebendo uma a uma as 288 partes (24 horas X 12 meses) e retornando-as simuladas para a classe *VenthorManager*.

Quanto ao banco de dados (SQL Server, no Venthor), o mesmo foi suprimido desta versão e

das paralelizadas¹ porque seria necessário manter num único local de acesso, as bases de dados, tornando-se um ponto de falha crítico, que impossibilitaria a continuidade dos experimentos no caso de falha até que fosse reestabelecido seu funcionamento, ou então, seria necessário a instalação do SGBD² em todas as máquinas participantes da grade, a qual é expansível, não sendo viável tal medida. Desta forma, optou-se por arquivos do tipo texto (txt ou csv), os quais são de fácil leitura, manipulação e gravação, e também pela facilidade de importação vindas dos SGBDs fornecedores e exportação tanto para SGBDs, quanto para visualização e manipulação em softwares como BROffice Calc ou Microsoft Excel, por exemplo. Na Figura 4.1 é exibida a estrutura de uma base de dados exportada de um SGBD no formato *csv*.

Figura 4.1: Base de dados de vento empírica, com os campos data, hora, velocidade e direção (variando de 0° à 360°).

	A	B	C	D
1	01/01/1998	00:00	3,4	80
2	01/01/1998	01:00	1,1	136
3	01/01/1998	02:00	3,4	34
4	01/01/1998	03:00	2,6	85
5	01/01/1998	04:00	1,7	56
6	01/01/1998	05:00	3,7	76
7	01/01/1998	06:00	3,4	56
8	01/01/1998	07:00	5,6	20
9	01/01/1998	08:00	5,3	56
10	01/01/1998	09:00	3,0	359
11	01/01/1998	10:00	1,2	328
12	01/01/1998	11:00	4,6	26
13	01/01/1998	12:00	3,7	330
14	01/01/1998	13:00	3,9	348
15	01/01/1998	14:00	7,5	219
16	01/01/1998	15:00	2,8	74
17	01/01/1998	16:00	0,6	261
18	01/01/1998	17:00	2,4	297
19	01/01/1998	18:00	2,2	8
20	01/01/1998	19:00	0,2	224
21	01/01/1998	20:00	0,3	326
22	01/01/1998	21:00	0,6	337
23	01/01/1998	22:00	1,2	26
24	01/01/1998	23:00	3,5	21
25	02/01/1998	00:00	2,3	20
26	02/01/1998	01:00	1,4	36
27	02/01/1998	02:00	2,3	209
28	02/01/1998	03:00	0,4	10
29	02/01/1998	04:00	2,1	348
30	02/01/1998	05:00	0,7	317
31	02/01/1998	06:00	0,5	25

Fonte: O autor.

Para ambas as versões, foi utilizado uma base de dados reais, composta por dez anos de

¹Maiores detalhes do funcionamento da versão paralela serão descritos na próxima seção.

²Sistema Gerenciador de Banco de Dados

dados coletados de hora em hora. Os dados analisados foram cedidos pelo Instituto Tecnológico SIMEPAR do Paraná, à UEPG - Universidade Estadual de Ponta Grossa-PR e obtidos junto a uma Estação Meteorológica localizada no Município de Lapa-PR.

Não existe, nas versões criadas neste trabalho e nem no Venthor original, procedimentos automatizados para limpeza e verificação da base de dados em relação a inconsistências, dados truncados ou vazios, ficando dessa forma a cargo do executor da simulação a verificação e eliminação desses possíveis inconvenientes.

4.2 Adaptação e Integração ao P2PComp

Após a realização de testes de execução ³ que comprovaram o correto funcionamento do modelo na versão Java, gerando bases simuladas, iniciou-se a integração ao P2PComp.

Para que o modelo adaptado e transcrito para o Java fosse reconhecido pelo *framework* P2PComp, foram necessárias algumas alterações na classe *VenthorManager* responsável pelo gerenciamento da simulação, já que o formato exigido segue o padrão MPI, conforme descrito em 3.3. Essas alterações lhe permitiram distribuir os conjuntos de dados de forma aleatória aos participantes ativos na grade e recebê-los depois de simulados da mesma forma, não se preocupando com a sequência correta das partes. A função de identificar esses conjuntos foi transferida para a classe *VenthorSimulation*, que inclui um rótulo contendo a hora/mês a que se refere cada parte simulada.

No Venthor (original) é possível visualizar através de interface gráfica, vários dados gerados a partir da simulação como: a média dos dados empíricos e dos dados simulados, o desvio padrão de ambos, os resultados do teste KS para cada grupo de dados hora/mês, entre outros. Como esses dados permitem confrontar valores e comprovar a acurácia da simulação, são de suma importância para quem vai utilizar essa base.

Devido à essa necessidade foi alterada a classe *VenthorSimulation* para que juntamente com o conjunto de dados simulados envie as informações correspondentes à validação da simulação desse conjunto. Dessa forma, também foi alterada a classe *VenthorManager* para gerar, juntamente com a nova base, um outro arquivo de mesmo formato contendo esses valores para cada hora/mês, e gravados simultaneamente com nomes seguindo o padrão: nome da base, total de anos simulados, se é a simulação ou a validação e o formato do arquivo, como mostrado nos exemplos: *VDOMLAPA_10anos_simul.csv* e *VDOMLAPA_10anos_valid.csv*.

³Os tempos de execução serão descritos no Capítulo 4.

Na Figura 4.2 é exibida a estrutura de uma base de dados simulada, contendo os campos: **data**, **hora**, **velocidade**, **direção** (em graus, de 0 - 360), **código da direção** (rosa dos ventos, numerados de 1 - 8) e **nome da direção** (rosa dos ventos: N, NE, E, SE, S, SO, O, NO). Foram mantidos esses seis campos diferente do original, com quatro, para facilitar eventuais necessidades do solicitante da simulação, podendo ser excluído qualquer um deles através do Microsoft Excel ou BROffice Calc.

Figura 4.2: Base de dados de vento simulada no formato texto (csv).

	A	B	C	D	E	F
1	01/01/2008	00:00	1,9	81	3	E
2	01/01/2008	01:00	1,8	81	3	E
3	01/01/2008	02:00	1,7	81	3	E
4	01/01/2008	03:00	1,5	81	3	E
5	01/01/2008	04:00	1,5	81	3	E
6	01/01/2008	05:00	1,5	81	3	E
7	01/01/2008	06:00	1,4	81	3	E
8	01/01/2008	07:00	1,5	81	3	E
9	01/01/2008	08:00	2	81	3	E
10	01/01/2008	09:00	2,3	81	3	E
11	01/01/2008	10:00	2,5	81	3	E
12	01/01/2008	11:00	2,5	81	3	E
13	01/01/2008	12:00	2,5	81	3	E
14	01/01/2008	13:00	2,6	126	4	SE
15	01/01/2008	14:00	2,5	126	4	SE
16	01/01/2008	15:00	2,6	126	4	SE
17	01/01/2008	16:00	2,6	126	4	SE
18	01/01/2008	17:00	2,4	126	4	SE
19	01/01/2008	18:00	2,4	126	4	SE
20	01/01/2008	19:00	2,3	81	3	E
21	01/01/2008	20:00	2,3	81	3	E
22	01/01/2008	21:00	2,1	81	3	E
23	01/01/2008	22:00	2,2	81	3	E
24	01/01/2008	23:00	2,1	81	3	E
25	02/01/2008	00:00	1,5	107	3	E
26	02/01/2008	01:00	1,5	107	3	E

Fonte: O autor.

Na Figura 4.3 é exibida a estrutura de um arquivo de validação, composta pelos campos: **Mês** (1 a 12), **Hora** (0 a 23), **N emp** (número de coletas realizadas para aquela hora/mês, excluindo-se os dados com falha), **N simul** (número de simulações realizadas para aquela hora/mês pelo período solicitado), **Media emp** (média das coletas empíricas), **Media simul** (média dos dados simulados), **DP emp** (desvio padrão dos dados empíricos), **DP simul** (desvio padrão dos dados simulados), **KS veloc** (teste KS sobre os dados simulados de velocidade), **KS veloc 5%** (limite máximo de 5% para o KS veloc), **KS veloc 1%** (para simples análise. Limite máximo de 1% para o KS veloc), **KS dir** (teste KS sobre os dados simulados de direção), **KS dir 5%** (limite máximo de 5% para o KS dir), **KS dir 1%** (para simples análise. Limite

máximo de 1% para o KS dir).

Figura 4.3: Dados de validação para uma base simulada, no formato csv.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Mes	Hora	N emp	N simul	Media emp	Media simul	DP emp	DP simul	KS veloc	KS veloc 5%	KS veloc 1%	KS dir	KS dir 5%	KS dir 1%
2	1	0	302	310	2,2053	2,3771	1,2691	1,3361	0,0515	0,1100	0,1318	0,0185	0,1100	0,1318
3	1	1	307	310	2,0535	2,1984	1,0370	1,0475	0,0620	0,1096	0,1313	0,0106	0,1096	0,1313
4	1	2	307	310	2,0245	2,1794	1,1324	1,1484	0,0523	0,1096	0,1313	0,0125	0,1096	0,1313
5	1	3	307	310	1,8906	2,0884	1,1152	1,4524	0,0741	0,1096	0,1313	0,0106	0,1096	0,1313
6	1	4	307	310	1,7655	1,9117	0,9961	1,0820	0,0521	0,1096	0,1313	0,0106	0,1096	0,1313
7	1	5	307	310	1,7600	1,9020	0,9975	1,0013	0,0739	0,1096	0,1313	0,0141	0,1096	0,1313
8	1	6	307	310	1,5929	1,7155	0,8739	0,9211	0,0606	0,1096	0,1313	0,0271	0,1096	0,1313
9	1	7	307	310	1,8111	1,9549	1,0737	1,1532	0,0820	0,1096	0,1313	0,0239	0,1096	0,1313
10	1	8	308	310	2,3173	2,4855	1,2108	1,3170	0,0672	0,1095	0,1312	0,0174	0,1095	0,1312
11	1	9	308	310	2,6569	2,8194	1,3136	1,3867	0,0541	0,1095	0,1312	0,0210	0,1095	0,1312
12	1	10	307	310	2,8623	3,0446	1,4574	1,5108	0,0399	0,1096	0,1313	0,0155	0,1096	0,1313
13	1	11	308	310	2,8942	3,0894	1,5048	1,6177	0,0674	0,1095	0,1312	0,0226	0,1095	0,1312
14	1	12	308	310	2,9156	3,1207	1,4737	1,7095	0,0512	0,1095	0,1312	0,0174	0,1095	0,1312
15	1	13	307	310	3,0075	3,2120	1,7154	1,7125	0,0539	0,1096	0,1313	0,0272	0,1096	0,1313
16	1	14	306	310	3,0458	3,2613	1,6874	1,8431	0,0412	0,1096	0,1314	0,0311	0,1096	0,1314
17	1	15	308	310	3,1838	3,4130	1,9137	2,0826	0,0546	0,1095	0,1312	0,0202	0,1095	0,1312
18	1	16	307	310	3,0339	3,2397	1,6512	1,7238	0,0426	0,1096	0,1313	0,0212	0,1096	0,1313
19	1	17	307	310	2,8219	3,0142	1,5224	1,5423	0,0479	0,1096	0,1313	0,0148	0,1096	0,1313
20	1	18	307	310	2,8108	3,0088	1,5446	1,6420	0,0543	0,1096	0,1313	0,0338	0,1096	0,1313
21	1	19	281	310	2,7435	3,0391	1,6005	1,8774	0,0660	0,1121	0,1343	0,0236	0,1121	0,1343
22	1	20	286	310	2,6507	2,8362	1,3958	1,5064	0,0536	0,1116	0,1337	0,0235	0,1116	0,1337
23	1	21	293	310	2,5925	2,7694	1,5143	1,6131	0,0637	0,1109	0,1329	0,0189	0,1109	0,1329
24	1	22	296	310	2,5788	2,7617	1,3230	1,4676	0,0515	0,1106	0,1325	0,0120	0,1106	0,1325
25	1	23	297	310	2,4287	2,6052	1,3187	1,3810	0,0522	0,1105	0,1324	0,0121	0,1105	0,1324
26	2	0	276	282	2,0881	2,2436	1,1691	1,1880	0,0569	0,1126	0,1349	0,0102	0,1126	0,1349
27	2	1	280	282	1,8765	2,0252	1,1098	1,1916	0,0776	0,1122	0,1344	0,0092	0,1122	0,1344
28	2	2	280	282	1,8879	2,0230	1,0307	0,9729	0,0692	0,1122	0,1344	0,0254	0,1122	0,1344
29	2	3	280	282	1,6893	1,8308	0,9096	0,9596	0,0457	0,1123	0,1345	0,0257	0,1122	0,1344
30	2	4	281	282	1,6780	1,8110	0,9045	0,9672	0,0638	0,1121	0,1343	0,0147	0,1121	0,1343
31	2	5	281	282	1,4815	1,6213	0,9365	1,0168	0,0839	0,1121	0,1343	0,0249	0,1121	0,1343

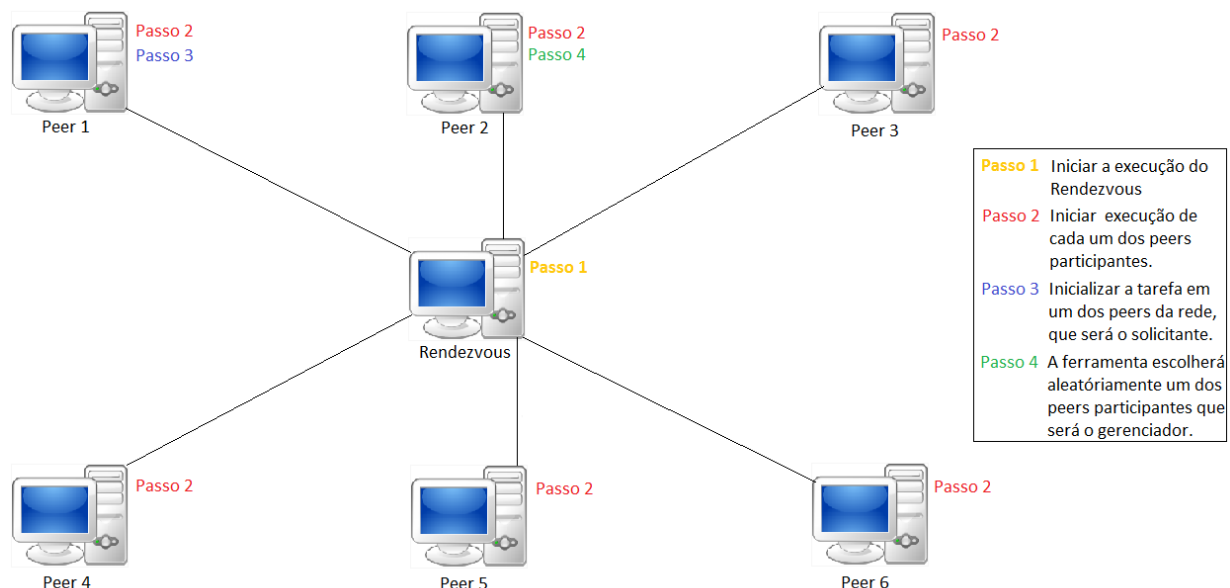
Fonte: O autor.

4.2.1 Esquema do Funcionamento no P2PComp

A Figura 4.4 exibe a estrutura e os passos fundamentais de inicialização do P2PComp e de uma tarefa. Num esquema como esse, onde existem 7 peers em funcionamento, apenas 5 trabalham na execução de uma tarefa, sendo um como gerenciador, responsável pela distribuição da tarefa e do recebimento processado e 4 como trabalhadores, executando o processamento e devolvendo ao gerenciador. Os outros dois peers (solicitante e rendezvous) não executam nenhum processamento da tarefa, apenas são parte da estrutura do P2PComp. O solicitante dá início a execução da tarefa e o rendezvous é responsável pelo roteamento das mensagens trocadas e por armazenar os anúncios dos participantes da rede.

É importante ressaltar que para melhor desempenho e utilização dos recursos disponíveis, foram colocados em execução um Peer por núcleo físico (processador), sendo que poderiam ser iniciados mais de um Peer por núcleo, mas esse procedimento poderia comprometer o desempenho e a medição dos resultados.

Figura 4.4: Funcionamento do P2PComp na execução de uma tarefa com os passos para a inicialização.



Fonte: O autor.

Nesse caso, o 'Peer 2' será o gerenciador e fará no final da execução, a junção das partes simuladas e a gravação da nova base de dados.

No P2PComp, cada Peer é uma entidade independente, não existindo gerenciamento centralizado da rede P2P, sendo que todos conhecem a todos, podendo se comunicar mutuamente através de seus nomes e PeerIDs únicos, conforme descrito na Seção 3.2. Essa comunicação é intermediada pelo *Peer Rendezvous (Super-Peer)* que armazena numa DHT (*Distributed Hash Table*) os resumos dos anúncios de cada participante, o que permite conhecer a rota para cada Peer conectado a essa rede.

A tarefa executada no P2PComp é gerenciada por um Peer definido pelo *framework*, ou seja, apesar de não existir um Peer específico, fixo, responsável pelo gerenciamento da tarefa, um deles será destinado a gerenciar a execução da mesma.

O método de comunicação utilizado pelo P2PComp na troca de mensagens entre as tarefas é o *JavaSocket*, que simula um Pipe Bidirecional, conforme descrito na Seção 3.2.

4.3 Ferramenta JXTA Venthor

Com a finalidade de comprovar e comparar a eficiência do *Framework* P2PComp, foi criada uma nova rede P2P com os Protocolos JXTA na linguagem Java e preparada para executar como tarefa, o modelo do Venthor paralelizado (o mesmo testado no P2PComp).

Essa nova rede foi criada seguindo padrões diferentes dos utilizados no P2PComp, como a execução do processo gerenciador, que aqui é definida manualmente por quem executa a simulação, mas que pode ser iniciada em qualquer Peer participante. Já no P2PComp essa decisão é tomada pelo *framework* de forma aleatória.

Outra diferença na estrutura está no padrão de construção das tarefas, que nesta ferramenta não segue o padrão MPI/PVM, sendo construídos e inicializados o gerenciador e o trabalhador, separadamente. A seguir serão descritos alguns detalhes do funcionamento desta ferramenta.

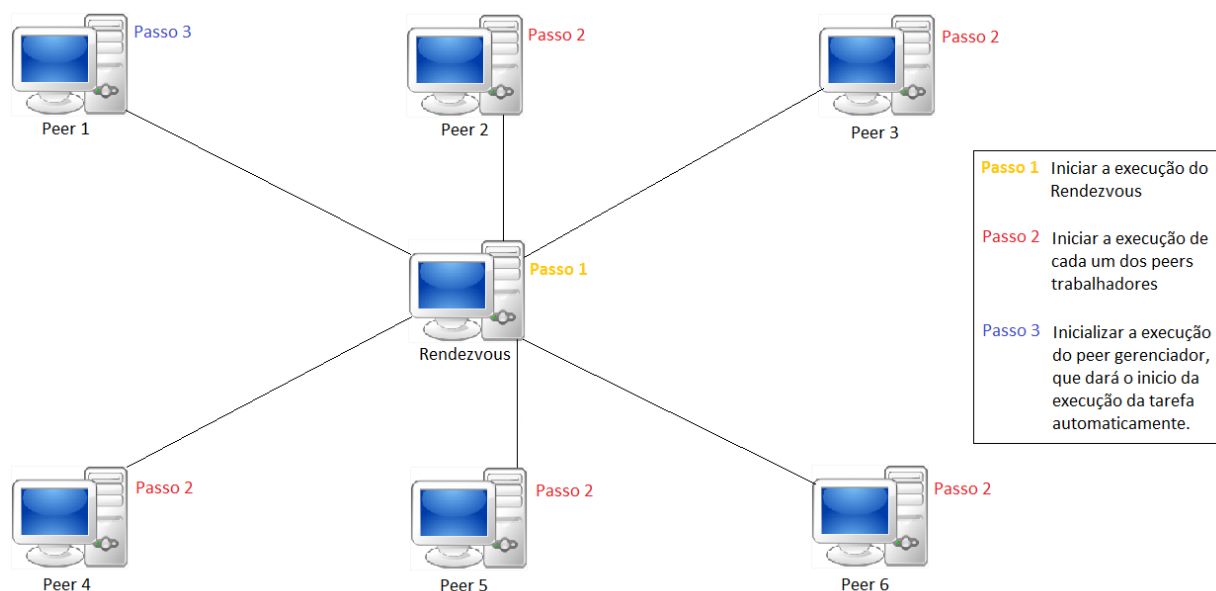
4.3.1 Esquema do Funcionamento no JXTA Venthor

Como é mostrado na Figura 4.5, no 'Passo 1' é iniciado o Rendezvous da mesma forma e com as mesmas funcionalidades que no P2PComp.

No 'Passo 2' são inicializados os trabalhadores, que ficam aguardando pela conexão com o gerenciador que lhes enviará trabalho.

No 'Passo 3' é inicializado o gerenciador, que ao se conectar com os trabalhadores iniciará automaticamente o envio das partes a serem simuladas e ficará aguardando o retorno para que ao final seja gravada a nova base de dados simulados.

Figura 4.5: Funcionamento do JXTA Venthor na execução de uma tarefa com os passos para a inicialização.



Fonte: O autor.

A comunicação entre os Peers é feita pelo gerenciador, que faz uma busca descobrindo os Peers conectados ao *Rendezvous* e envia as partes a serem executadas para a lista de Peers trabalhadores.

O método de comunicação utilizado nesta ferramenta foi o *Unicast Pipes* que é de mais baixo nível que os *JxtaSockets*. Essa escolha não resulta em redução significativa no tempo de execução da tarefa, mas leva vantagem sobre os *JxtaSockets* como será mostrado no próximo capítulo. *Unicast Pipes* tem maior grau de complexidade na sua construção e controle, já que o desenvolvedor precisa controlar o fluxo nos dois sentidos.

4.4 Tempos de Execução

Para obtenção dos tempos de execução e comparação entre versões foi estabelecido como padrão a mesma base de dados e 4 testes por versão, sendo eles para: 10 anos, 20 anos, 50 anos e 100 anos de dados simulados. Para cada versão foram executados mais de 10 testes em cada série de anos simulados, onde foi constatado uma variação média de menos de 1% entre elas, não representando valores significativos.

Em cada teste realizado, estavam em execução o sistema operacional, a ferramenta utilizada e a tarefa, não sendo executado nenhum outro software durante os testes.

A seguir serão descritos os resultados obtidos para todas as versões do simulador do Venthor, sendo elas: Venthor Original, Venthor Java, Venthor P2PComp e JXTA Venthor. Os tempos exibidos seguem o padrão: 'hh:mm:ss' (horas, minutos e segundos).

Na Tabela 4.1 são mostrados os resultados obtidos na execução do Venthor original, executado em ambiente Windows, sendo que nas demais versões é utilizado o Linux.

Tabela 4.1: Tempos de execução para o Venthor original.

Total de Anos	Tempo de Execução
10	00:08:16
20	00:19:16
50	00:25:24
100	00:57:24

Fonte: O autor.

Na Tabela 4.2 são mostrados os resultados para a versão transcrita para a linguagem Java (sem paralelização).

Nota-se aqui uma redução expressiva no tempo de execução das simulações como, por exemplo, na simulação de 10 anos, de 00:08:16 (original) para 00:00:06 (versão java). Essa redução se deve a fatores adotados a partir desta versão, onde foram suprimidos o acesso ao banco de dados (eliminando as chamadas frequentes ao mesmo e para desobrigar a instalação do SGBD nas máquinas participantes nas versões paralelizadas), utilização de arquivos texto,

Tabela 4.2: Tempos de execução para o Venthor Java (sem paralelização).

Total de Anos	Tempo de Execução
10	00:00:06
20	00:00:17
50	00:01:34
100	00:08:54

Fonte: O autor.

organização dos algoritmos transcritos e a utilização somente do núcleo do simulador, descartando outros procedimentos e ferramentas presentes na versão original. Foram realizados testes desta versão, tanto em sistema operacional Windows como Linux, alcançando resultados muito próximos, variando de 1 a 2 segundos para menos a favor do Linux.

Nas Tabelas 4.3, 4.4 e 4.5 são mostrados os resultados obtidos na execução da versão paralelizada do Venthor com o P2PComp. Nesta versão, bem como na do JXTA Venthor, a estrutura da tabela incluirá o número de Peers utilizados em cada teste, sendo que foram realizados testes com 6, 14 e 26 Peers. Nesta versão é utilizado um Peer que serve apenas para iniciar a execução, como mostrado na Figura 4.4, que não será contabilizado nos cálculos de tempo e de *speedup*⁴ que serão exibidos ainda neste capítulo, devido ao mesmo não executar cálculo algum e permanecer estático durante a execução da tarefa.

Tabela 4.3: Tempos de execução para o Venthor P2PComp (paralelizado) utilizando 6 Peers.

Total de Anos	Tempo de Execução
10	00:00:12
20	00:00:23
50	00:01:36
100	00:06:39

Fonte: O autor.

Tabela 4.4: Tempos de execução para o Venthor P2PComp (paralelizado) utilizando 14 Peers.

Total de Anos	Tempo de Execução
10	00:00:10
20	00:00:14
50	00:00:43
100	00:02:24

Fonte: O autor.

Nas Tabelas 4.6, 4.7 e 4.8 são exibidos os resultados da execução da versão JXTA Venthor, também realizados com 6, 14 e 26 Peers, sendo que nessa versão não existe um Peer específico para comandar o início da execução da tarefa, a qual é iniciada automaticamente ao ser inicializado o gerenciador.

⁴conforme descrito na Seção 4.5.

Tabela 4.5: Tempos de execução para o Venthor P2PComp (paralelizado) utilizando 26 Peers.

Total de Anos	Tempo de Execução
10	00:00:12
20	00:00:15
50	00:00:33
100	00:01:36

Fonte: O autor.

Tabela 4.6: Tempos de execução para o JXTA Venthor (paralelizado) utilizando 6 Peers.

Total de Anos	Tempo de Execução
10	00:00:07
20	00:00:18
50	00:01:27
100	00:06:07

Fonte: O autor.

Tabela 4.7: Tempos de execução para o JXTA Venthor (paralelizado) utilizando 14 Peers.

Total de Anos	Tempo de Execução
10	00:00:05
20	00:00:11
50	00:00:32
100	00:02:12

Fonte: O autor.

Tabela 4.8: Tempos de execução para o JXTA Venthor (paralelizado) utilizando 26 Peers.

Total de Anos	Tempo de Execução
10	00:00:06
20	00:00:07
50	00:00:20
100	00:01:16

Fonte: O autor.

Na Tabela 4.9 são mostrados os tempos de execução de todas as versões paralelamente, destacados em negrito os melhores tempos.

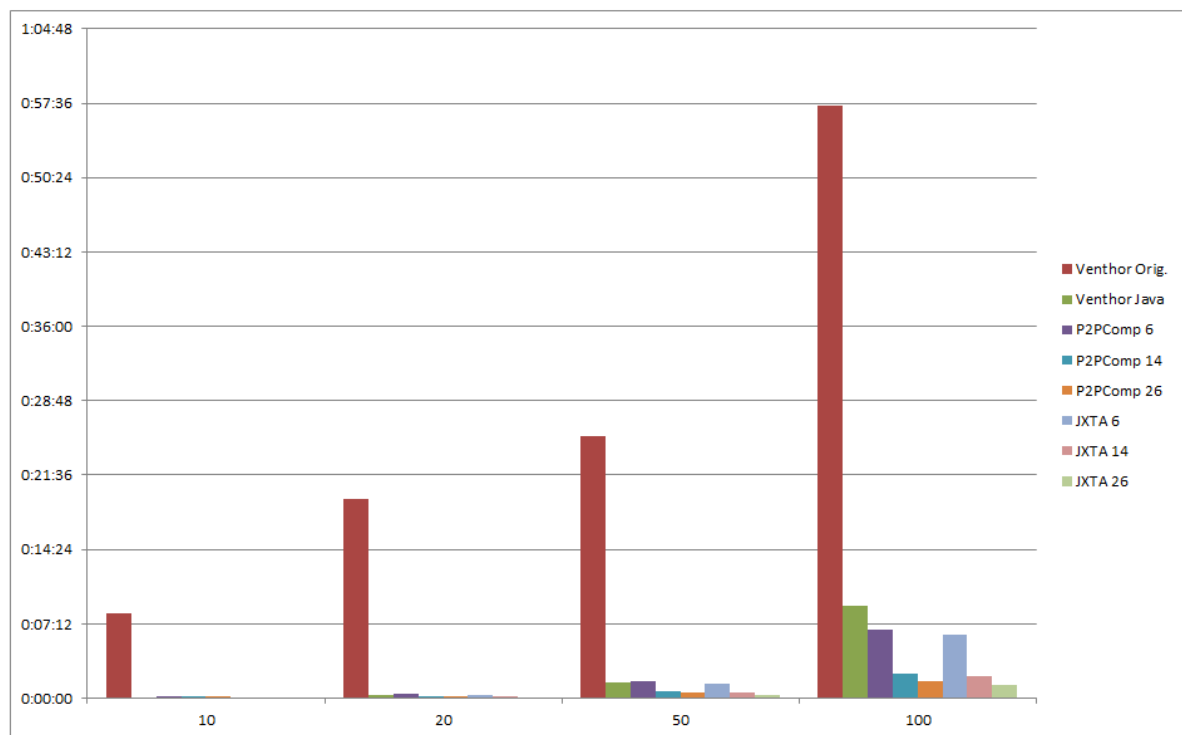
Tabela 4.9: Tempos de execução para todas as versões do Venthor.

Ano	Venthor	VenthorJava	P2PComp6	P2PComp14	P2PComp26	Jxta6	Jxta14	Jxta26
10	00:08:16	00:00:06	00:00:12	00:00:10	00:00:12	00:00:07	00:00:05	00:00:06
20	00:19:16	00:00:17	00:00:23	00:00:14	00:00:15	00:00:18	00:00:11	00:00:07
50	00:25:24	00:01:34	00:01:36	00:00:43	00:00:33	00:01:27	00:00:32	00:00:20
100	00:57:24	00:08:54	00:06:39	00:02:24	00:01:36	00:06:07	00:02:12	00:01:16

Fonte: O autor.

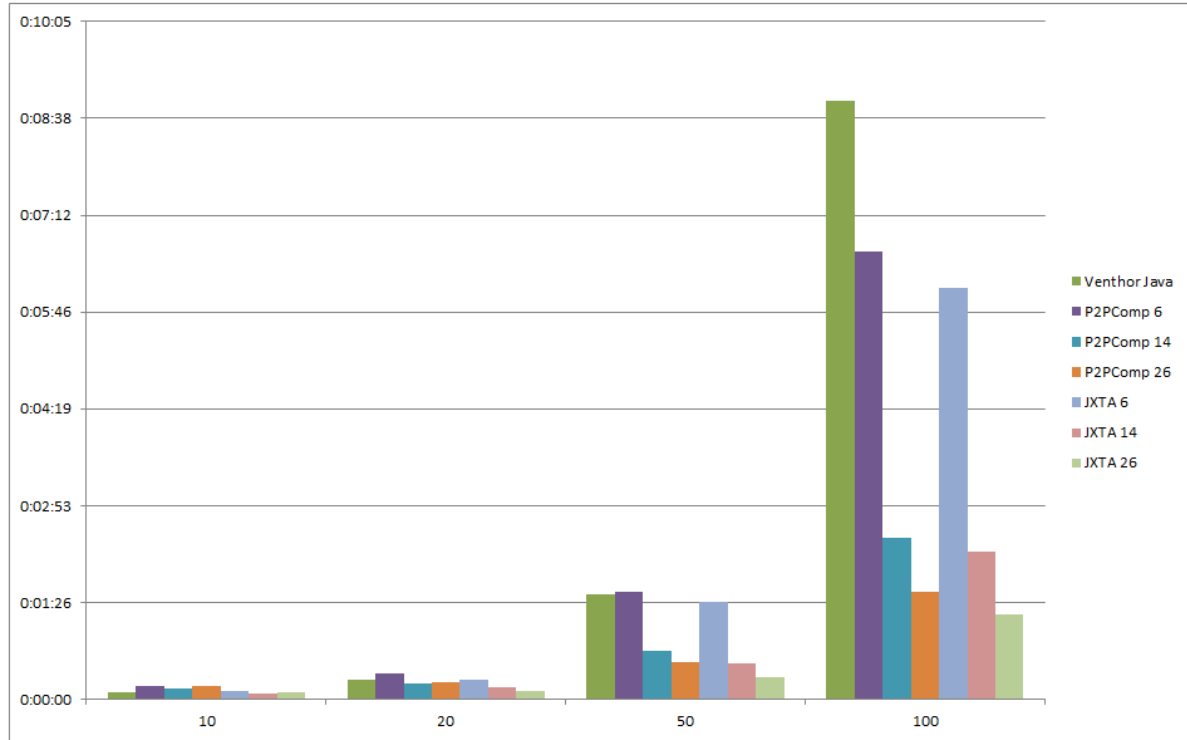
As Figuras 4.6 e 4.7 mostram graficamente os tempos de execução, sendo a 4.6 incluindo o tempo do Venthor original e a 4.7 excluindo-o.

Figura 4.6: Gráfico com os tempos de execução incluindo o Venthor Original.



Fonte: O autor.

Figura 4.7: Gráfico com os tempos de execução sem o Venthor Original.



Fonte: O autor.

4.5 Resultados de Speedup

O cálculo do *speedup* é definido pela Equação 4.1:

$$S_p = \frac{T_1}{T_p} \quad (4.1)$$

onde, ' p ' = número de processadores, ' T_1 ' = tempo de execução do algoritmo sequencial e ' T_p ' = tempo de execução do algoritmo paralelo com ' p ' processadores. Dessa forma, o *speedup* ideal é obtido quando: ' $S_p = p$ '. Ao executar um algoritmo com *speedup* linear, duplicando o número de processadores, dobra-se a velocidade. Como isto é ideal, considera-se a escalabilidade muito boa. Na prática, é difícil alcançar níveis de *speedup* ideais ou bem próximos dele. Isso se deve, na maioria das vezes, ao grau de paralelismo de um algoritmo, já que parte dele (parte gerencial) continua sendo executada de forma sequencial e que se somado a outros fatores como *overhead* de rede, influenciarão no resultado final do *speedup* [37].

Para exemplificar o cálculo de *speedup*, temos como exemplo um algoritmo executando numa única máquina de forma sequencial e que levou 00:02:00 para terminar. Seu *speedup* sempre terá como base o valor '1'. Ao executar o mesmo algoritmo de forma paralela, dividindo-o

e executando em duas máquinas, o tempo gasto foi de 00:01:10. Por ter sido executado em duas máquinas, o *speedup* ideal seria '2'. Ao calcular o *speedup* para o tempo gasto, obtem-se o valor de 1,71, sendo que esse valor somente chegaria a 2 se o algoritmo paralelo tivesse executado num tempo de 00:01:00, ou seja, a metade do tempo do algoritmo sequencial.

Nas Tabelas 4.10 e 4.11, são mostrados os valores de *speedup* com até duas casas de precisão, sendo que nas linhas de cabeçalho constam, quando for o caso, o valor ideal para o *speedup* como nos casos onde são utilizados 6, 14 ou 26 Peers.

Nas Tabelas 4.10 e 4.11 serão exibidos os cálculos de *speedup* para os tempos mostrados na seção anterior e serão divididos em duas partes: a primeira, comparando as versões criadas neste estudo com a versão original do Venthor; e a segunda, comparando as versões paralelizadas com a versão transcrita para o Java (sem paralelização).

4.5.1 Venthor Original versus Java e P2P

Os dados mostrados na Tabela 4.10 comparam o *speedup* do Venthor original com os valores obtidos nas demais versões.

Tabela 4.10: Comparativo entre o *speedup* do Venthor original e as demais versões.

Ano	Venthor	VenthorJava	P2PComp6	P2PComp14	P2PComp26	Jxta6	Jxta14	Jxta26
10	1	82,67	41,33	49,60	41,33	70,86	99,20 *	82,67
20	1	68,00	50,26	82,57	77,07	64,22	105,09	165,14 *
50	1	16,21	15,88	35,44	46,18	17,52	47,63	76,20 *
100	1	6,45	8,63	23,92	35,88	9,38	26,09	45,32 *

Fonte: O autor.

Na Tabela 4.10 estão destacados em negrito os menores (piores) e negrito com um (*) os maiores (melhores) valores de cada série (10, 20, 50 e 100). Observa-se também que, apesar de alguns experimentos (6 e 14 Peers) não obterem os melhores resultados, podem alcançar uma relação custo/benefício satisfatória, já que requerem menor número de máquinas. Isso possibilita flexibilidade na execução dos experimentos, mas requer uma análise para cada tipo de experimento a ser realizado, para que haja um balanceamento entre necessidade e disponibilidade.

4.5.2 Venthor Java versus P2P

Na Tabela 4.11 são exibidos os valores de *speedup* da versão Venthor Java em comparação com as versões paralelizadas (P2PComp e JXTA). Os valores mostrados nesta tabela são os valores de referência para este trabalho, considerando os dados da Tabela 4.10 apenas como

ilustração dos ganhos obtidos ao término deste estudo em relação ao software original e será considerado como melhoria do algoritmo, já que houve uma redução considerável no tempo de execução quando o modelo foi transcrito para a linguagem Java (modelo não paralelizado).

Tabela 4.11: Comparativo entre o *speedup* do Venthor Java e das versões paralelizadas.

Anos	Venthor Java	P2PComp 6	P2PComp 14	P2PComp 26	JXTA 6	JXTA 14	JXTA 26
10	1	0,50	0,60	0,50	0,86	1,20 *	1,00
20	1	0,74	1,21	1,13	0,94	1,55	2,43 *
50	1	0,98	2,19	2,85	1,08	2,94	4,70 *
100	1	1,34	3,71	5,56	1,46	4,05	7,03 *

Fonte: O autor.

Na prática é difícil alcançar valores de *speedup* ideais, já que sofrem influência de diversos fatores como *overhead* de rede e o grau de paralelização do algoritmo (que acaba executando parte dele sequencialmente no processo gerenciador), mas esses ganhos por menores que sejam, são de grande importância para as áreas envolvidas, onde o tempo gasto em excesso pode levar além do atraso em pesquisas e cronogramas, à perda de recursos financeiros.

Apesar dos valores obtidos não se aproximarem do valor ideal, os resultados são considerados bons, já que ao final de algumas execuções, teremos acumulado horas não desperdiçadas, aguardando o seu término.

5 Conclusão

Diante das condições em que esta pesquisa foi elaborada, os materiais e métodos utilizados e devido aos resultados obtidos, pode-se concluir que:

(1) Houve um ganho satisfatório no tempo de execução das simulações quando foi transcrito o modelo original do Venthor para a linguagem Java, através da reconstrução do algoritmo (parte gerenciadora).

(2) A utilização do modelo paralelizado, juntamente com o *Framework* P2PComp permitiu resultados significativos quando há um volume maior de dados a serem processados, associado a um número significativo de anos a simular.

(3) Nas simulações com bases de dados empíricas pequenas e com poucos anos a simular, é viável a utilização de menos Peers, ou até mesmo alguns Peers numa única máquina *multicore*, podendo ser utilizando o P2PComp.

(4) Fica comprovada a eficiência do *Framework* P2PComp para execução de tarefas paralelizadas, ao serem comparados seus resultados de execução com os da rede P2P (JXTA Venthor). Apesar de ter tido resultados inferiores ao JXTA Venthor (que tem menos recursos embarcados), o P2PComp obteve índices satisfatórios, sendo perfeitamente aplicável a outras tarefas, como na simulação das demais variáveis climáticas.

(5) Conclui-se ainda que devido à previsão de aumento no volume de dados para os próximos anos, tanto pelo volume acumulado no padrão horário, que é coletado nas estações meteorológicas, como pelos novos padrões a cada 15 ou 5 minutos, o que gerará um enorme volume de dados, será impraticável a simulação destes sem os benefícios da computação paralela.

Por fim, os resultados obtidos com os testes demonstram a viabilidade de aplicação da computação paralela e de suas ferramentas em simulações climáticas, bem como podem ser utilizadas em outras vertentes da agricultura, que demandem grande poder de processamento como: a mineração de dados agrícolas, estimativas de produtividade, genética de plantas, entre outras.

Referências Bibliográficas

- 1 PERIN FILHO, C. *Introdução a Simulação de Sistemas*. [S.l.]: Editora da Unicamp, 1995.
- 2 OZSU, M. T.; VALDURIEZ, P. *Principles of Distributed Databases*. [S.l.]: Prentice-Hall, 2007. ISBN 0-13-041212-0.
- 3 MUKHOPADHYAY, Debajyoti et al. Architecture of a scalable dynamic parallel webcrawler with high speed downloadable capability for a web search engine. *CoRR*, abs/1102.0676, 2011.
- 4 VIRGENS FILHO, J. S. et al. Simulador estocástico baseado em modelos probabilísticos, para geração de séries horárias de vento. *XVI Congresso Brasileiro de Agrometeorologia - Belo Horizonte-MG*, Editora da SBAGRO, v. 1, p. 1–5, 2009.
- 5 VIRGENS FILHO, J. S.; LEITE, M. L. Ajuste de modelos de distribuição de probabilidade a séries horárias de velocidade do vento para o município de ponta grossa, estado do paraná. *Acta Scientiarum. Technology*, v. 33, n. 4, p. 447–455, 2011.
- 6 VIRGENS FILHO, J. S. *Ferramenta computacional para simulação de séries climáticas diárias, baseada na parametrização dinâmica das distribuições de probabilidade*. 92 p. Tese (Doutorado) — Universidade Estadual Paulista, 2001.
- 7 SILVA, F. G.; BERGAMASCO, A. F.; VENDITE, L. L. Modelos de simulação para análise e apoio a decisão em agrossistemas. *Revista Biociências de Taubaté-SP*, v. 8, n. 2, p. 7–17, 2002.
- 8 SEMENOV, M. A.; BARROW, E. M. Use of a stochastic weather generator in the development of climate change scenarios. *Climatic Change*, n. 35, p. 397–414, 1997.
- 9 SEMENOV, M. A. et al. Comparison of the wgen and lars-wg stochastic weather generators for diverse climates. *Climate Research*, n. 10, p. 95–107, 1998.
- 10 EVANGELISTA, Adão W. P. et al. Avaliação do desempenho do modelo cligen em gerar dados climáticos para modelos hidrológicos e agrônômicos. *Revista Brasileira de Engenharia Agrícola e Ambiental*, scielo, v. 10, p. 119–127, 2006. ISSN 1415-4366.
- 11 SOLTANI, A.; HOOGENBOOM, G. A statistical comparison of the stochastic weather generators wgen and simmeteo. *Climate Research*, v. 24, n. 3, p. 215–230, 2003.
- 12 PICKERING, N. B. et al. Weatherman: a utility for managing and generating daily weather data. *Agronomy Journal*, v. 86, p. 332–337, 1994. ISSN 00021962.
- 13 VIRGENS FILHO, J. S. *Modelo computacional para simulação de dados climáticos*. 86 p. Dissertação (Mestrado) — Universidade Estadual Paulista, 1997.

- 14 CAMARGO, Marcelo Bento Paes de; ORTOLANI, Altino Aldo; ARRUDA, Hermano Vaz de. Ocorrência mensal de rajadas máximas diárias de vento em Campinas (SP). *Bragantia*, sciELO, v. 53, p. 107 – 112, 1994. ISSN 0006-8705.
- 15 MARQUES JUNIOR, S. et al. Análise de dados de vento para a região de botucatu-sp utilizando a distribuição beta. *Revista Brasileira de Agrometeorologia*, n. 3, p. 129–132, 1995.
- 16 SANSIGOLO, C. A. Distribuições de probabilidade de velocidade e potência do vento. *Revista Brasileira de Meteorologia*, v. 20, n. 2, p. 210–217, 2005.
- 17 ASSIS, F. N.; ARRUDA, H. V.; PEREIRA, A. R. *Aplicações de estatística à climatologia: teoria e prática*. [S.l.]: Pelotas: Editora Universitária/UFPel, 1996. 161 p.
- 18 BUSSAB, W. O.; MORETIN, P. A. *Estatística básica*. [S.l.]: Saraiva, 2004.
- 19 COULOURIS, G. et al. *Distributed Systems - Concepts and Design*. [S.l.]: Addison Wesley, 2011. ISBN 0-13-214301-1.
- 20 LORIN, H. R. Review of "highly parallel computing" by g. s. almasi and a. gottlieb, benjamin-cummings publishers, redwood city, ca, 1989. *IBM Syst. J.*, IBM Corp., Riverton, NJ, USA, v. 29, n. 1, p. 165–166, 1990. ISSN 0018-8670.
- 21 SHEN, J. P.; MIKKO, H. L. *Modern Processor Design: Fundamentals of Superscalar Processors*. [S.l.]: McGraw-Hill Professional, 2005. ISBN 0070570647.
- 22 SNIR, M. et al. *MPI. The Complete Reference: Volume 1, The MPI Core*. [S.l.]: MIT Press, 1998. ISBN 0-262-69215-5.
- 23 BURNS, G.; DAOUD, R.; VAIGL, J. Lam: An open cluster environment for mpi. In *Proceedings of Supercomputing Symposium*, p. 379–386, 1994.
- 24 GEIST, A. et al. Pvm: A users' guide and tutorial for networked parallel computing. MIT Press, 1994.
- 25 BEGUELIN, A. et al. *PVM: Parallel Virtual Machine: User's Guide and tutorial for Networked Parallel Computing*. [S.l.]: MIT Press, 1994.
- 26 BUYYA, R. *High Performance Cluster Computing: Architectures And Systems*. [S.l.]: Prentice Hall NJ - USA, 1999.
- 27 PITANGA, M. *Computação em Cluster*. [S.l.]: Brasport, 2004.
- 28 ANDERSON, D. P. et al. Seti@home: An experiment in public-resource computing. *Communications of the ACM*, v. 45, n. 11, p. 56–61, 2002.
- 29 BALATON, Z. et al. Sztaki desktop grid: a modular and scalable way of building large computing grids. *Proceedings of the 21st International Parallel and Distributed Processing Symposium - Long Beach, California-USA*, p. 26–30, 2007.
- 30 FEDAK, G. et al. Xtremweb: A generic global computing system. In *Proc. of CCGRID2001 Workshop on Global Computing on Personal Devices - IEEE Press*, 2001.
- 31 SOTOMAYOR, B.; CHILDERS, L. Lisa. *Globus Toolkit 4: Programming Java Services*. [S.l.]: Morgan Kaufmann, 2005. ISBN 0123694043.

- 32 GRIMSHAW, A. S. et al. Metasystems. *Comm. of the ACM*, v. 41, n. 11, 1998.
- 33 CALIN, M. et al. Involving grid computing in agricultural research. *Research Journal of Agricultural Science*, v. 39, n. 2, 2007.
- 34 GRADECKI, J. D. *Mastering JXTA: building Java Peer-to-Peer Applications*. [S.l.]: Wiley Publishing Inc, 2002.
- 35 WILSON, B. J. *JXTA*. [S.l.]: New Riders, 2002.
- 36 SENGER, L. J.; SOUZA, M. A.; FOLTRAN JUNIOR, D. C. Towards a peer-to-peer framework for parallel and distributed computing. *International Symposium on Computer Architecture and High Performance Computing*, IEEE Computer Society, v. 24, p. 127–134, 2010.
- 37 ALMASI, G. S.; GOTTLIEB, A. *Highly parallel computing*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1989. ISBN 0-8053-0177-1.
- 38 JUSTUS, C. G. et al. Methods for estimating wind speed frequency distributions. *Journal of Applied Meteorology*, IEEE Computer Society, v. 17, n. 3, p. 350–353, 1978.
- 39 FALLS, L. W. The beta distribution: a statistical model for world cloud cover. *NASA Technical Memorandum, TMX-64714*, NASA, p. 1–6, 1973.
- 40 PEARSON, K. *Tables of the incomplete Beta function*. [S.l.]: University College of London, Biometriks Office, 1934.
- 41 GREENWOOD, J. A.; DURAND, D. Aids for fitting the gamma distribution by maximum likelihood. *Technometrics*, v. 2, n. 1, p. 55–65, 1960.

ANEXO A – Equações do Modelo

A.1 Distribuições de Probabilidade

A.1.1 Distribuição de Weibull

A distribuição de Weibull 2 parâmetros para a velocidade do vento é expressa pela função de densidade de probabilidade:

$$f(v) = \left(\frac{k}{c}\right) \left(\frac{v}{c}\right)^{k-1} \exp \left[- \left(\frac{v}{c}\right)^k \right] \quad (\text{A.1})$$

em que a função cumulativa de probabilidade dada por:

$$F(v) = 1 - \exp \left[- \left(\frac{v}{c}\right)^k \right] \quad (\text{A.2})$$

em que: c é o fator de escala em unidades de velocidade do vento, k , o fator de forma adimensional e v , a variável aleatória velocidade do vento. O fator de forma k está inversamente relacionado à variância σ^2 das velocidades eólicas em torno da média. Os parâmetros c e k , conforme discutidos em Justus et al.[38], podem ser determinados a partir da transformação da equação A.2 na forma linear.

$$\ln(-\ln(1 - F(v))) = -k \ln(c) + k \ln(v) \quad (\text{A.3})$$

que pode ser representada pela reta: $Y = a + bX$

em que:

$$Y = \ln [- \ln (1 - F(v))];$$

$$X = \ln (v);$$

$$a = - k \ln (c) \text{ e}$$

$$b = k$$

Assim, a determinação dos parâmetros c e k fica condicionada aos cálculos dos coeficientes a e b da reta. Esses podem ser obtidos pelo método dos mínimos quadrados aplicado ao conjunto de dados $X = \ln(v)$ e $Y = \ln[-\ln(1 - F(v))]$ obtidos dos valores de v e $F(v)$ que, por sua vez, são determinados a partir das séries observadas da velocidade do vento distribuídas em n intervalos de classe com suas respectivas frequências.

A.1.2 Distribuição de Rayleigh

A expressão matemática da função de densidade de probabilidade de Rayleigh é dada por:

$$f(v) = -\frac{v}{a^2} \exp\left(-\frac{v^2}{2a^2}\right) \quad (\text{A.4})$$

em que:

$$a = \sqrt{\frac{2}{\pi}} E(v) \quad (\text{A.5})$$

e $E(v)$ é a esperança matemática da variável aleatória v (velocidade do vento). A função cumulativa de probabilidade Rayleigh é:

$$F(v) = 1 - \exp\left(-\frac{\pi v^2}{4E(v)^2}\right) \quad (\text{A.6})$$

A.1.3 Distribuição de Beta

Essa função de densidade de probabilidade pode ser expressa da seguinte forma [39]:

$$F(v) = \frac{1}{(b-a)} \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} \left(\frac{v-a}{b-a}\right)^{p-1} \left(1 - \frac{v-a}{b-a}\right)^{q-1} \quad (\text{A.7})$$

em que: a e b correspondem ao menor e maior valor da série de dados, respectivamente, Γ é o símbolo da função Gama das respectivas variáveis, p e q são parâmetros da distribuição Beta e v é um valor qualquer da variável em análise. A estimativa dos parâmetros p e q pode ser realizada a partir do método dos momentos [40].

$$p = \frac{\mu_1(\mu_1 - \mu_2)}{[\mu_2 - (\mu_1)^2]} \quad (\text{A.8})$$

$$q = \frac{(1 - \mu_1)(\mu_1 - \mu_2)}{[\mu_2 - (\mu_1)^2]} \quad (\text{A.9})$$

em que: μ_1 corresponde ao momento de ordem 1 para a variável v e μ_2 ao momento de ordem 2 para a variável v , dentro de uma série de n dados. Estes termos podem ser estimados a partir da seguinte análise:

$$\mu_1 = \frac{\sum_{i=1}^N v_i}{N} \quad (\text{A.10})$$

$$\mu_2 = \frac{\sum_{i=1}^N v_i^2}{N} \quad (\text{A.11})$$

Para a estimativa dos valores de ocorrência de probabilidade, por meio da distribuição Beta, a equação A.7 deve ser adimensionalizada para um intervalo compreendido entre [0 e 1]. A variável adimensionalizada v toma então a seguinte forma:

$$v' = \frac{v - a}{b - a} \quad (\text{A.12})$$

sugerindo que a função de densidade de probabilidade Beta assuma a seguinte forma:

$$f(v') = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} (v')^{p-1} (1 - v')^{q-1} \quad (\text{A.13})$$

em que: $0 \leq v' \leq 1$, para $p > 1$ e $q > 1$. A integração numérica da equação A.13 confere os valores da probabilidade de ocorrência para um valor de v qualquer dentro do intervalo considerado.

A.1.4 Distribuição de Gama

A função de densidade de probabilidade Gama é expressa por:

$$f(v) = \frac{v^{\alpha-1} e^{-\frac{v}{\beta}}}{\beta^\alpha \Gamma(\alpha)} \quad (\text{A.14})$$

em que: α = parâmetro de forma, estimado pelo método de Greenwood e Durand[41], dado por:

$$\alpha = \frac{8,898919 + 9,05995y + 0,9775373y^2}{y(17,79728 + 11,968477y + y^2)} \quad (\text{A.15})$$

para $0 \leq y \leq 0,5772$. Ou

$$\alpha = \frac{0,5000876 + 0,1648852y - 0,0544274y^2}{y} \quad (\text{A.16})$$

para $0,5772 < y \leq 17,0$ em que:

$$y = \ln \left(\frac{\frac{1}{N} \sum_{i=1}^N v_i}{\left(\prod_{i=1}^N v_i \right)^{\frac{1}{N}}} \right) \quad (\text{A.17})$$

β = parâmetro de escala estimado a partir de:

$$\beta = \frac{\sum_{i=1}^N v_i}{N\alpha} \quad (\text{A.18})$$

$\Gamma(\alpha)$ = Função Gama ordinária de α . A função cumulativa de probabilidade é:

$$F(v) = \frac{1}{\Gamma(\gamma)\beta^\gamma} \int_0^v v^{\gamma-1} e^{-\frac{v}{\beta}} dv \quad (\text{A.19})$$

A.2 Teste de Kolmogorov-Smirnov

O teste de Kolmogorov-Smirnov é aplicado para verificar se os valores de uma certa amostra de dados podem ser considerados como provenientes de uma população com distribuição teórica pré-estabelecida, sob uma hipótese: a hipótese de nulidade (H_0). O teste confronta duas distribuições de frequência acumuladas, uma $F(X)$, teórica, e outra, $F(X)$, derivada dos dados amostrais, tal que: - Seja $F(X)$ uma função de distribuição de probabilidade teórica com seus parâmetros especificados; - Seja $F(X)$ uma distribuição de probabilidade empírica, ou seja, para uma determinada classe de frequência $F(X) = fa/N$, onde fa é a frequência acumulada da classe;

Determina-se:

$$D_{max} = MAX|F'(X) - F(X)| \quad (A.20)$$

Se, ao nível de significância estabelecido, o valor observado de $D_{m\acute{a}x}$ (calculado) for maior ou igual ao valor crítico de $D_{m\acute{a}x}$ (tabelado), a hipótese de nulidade, ou seja, a hipótese de que os dados amostrais provêm de uma população com distribuição teórica, $F(X)$ é rejeitada [17].

A.3 Cálculo da Direção do Vento

Já para a análise da direção predominante do vento, os dados foram descritos probabilisticamente através da frequência relativa simples (fr_d) para cada mês do ano do período em estudo, cuja fórmula esta descrita abaixo:

$$fr_d = \frac{f_d}{N} \quad (A.21)$$

onde f_d é a frequência de cada direção e N é o numero de observações dentro do mês analisado. Para determinar as direções dos ventos, foram utilizados os pontos cardeais Norte, Sul, Leste e Oeste e também os pontos colaterais Nordeste, Sudoeste, Sudeste e Noroeste, através da rosa-dos-ventos.