CRONOSIM – UMA FERRAMENTA DE SIMULAÇÃO DISTRIBUÍDA DE EVENTOS DISCRETOS COM GERENCIAMENTO DE PROCESSOS

Luiz Fernando Nunes
Universidade Federal de Itajubá
AV. BPS, 1303 – Pinheirinho - Cep. 37.500-903 –Itajubá –MG
luizfernandolfn@gmail.com

Renan Curvello Faria
Universidade Federal de Itajubá
AV. BPS, 1303 – Pinheirinho - Cep. 37.500-903 –Itajubá –MG
renan.curvello@gmail.com

Edmilson Marmo Moreira
Universidade Federal de Itajubá
AV. BPS, 1303 – Pinheirinho – Cep. 37.500-903 – Itajubá – MG
edmarmo@unifei.edu.br

João Paulo Chaves Barbosa
Universidade Federal de Itajubá
AV. BPS, 1303 – Pinheirinho - Cep. 37.500-903 –Itajubá –MG
jpccomp@gmail.com

Otávio Augusto Salgado Carpinteiro
Universidade Federal de Itajubá
AV. BPS, 1303 – Pinheirinho - Cep. 37.500-903 –Itajubá –MG
otavio@unifei.edu.br

RESUMO

Este artigo apresenta CronoSim, uma ferramenta que permite a modelagem e a execução da simulação de um modelo de eventos discretos em uma infraestrutura computacional distribuída, através de uma máquina paralela ou de um sistema distribuído, proporcionando melhor desempenho. Ela também fornece recursos para gerenciar os processos lógicos do programa, permitindo otimizar a execução através de algoritmos de balanceamento de carga e escalonamento de processos. Em adição, esta ferramenta auxilia usuários, sem experiência em computação paralela, na paralelização do modelo de simulação.

PALAVRAS CHAVE. Simulação Distribuída. Ferramenta de Simulação. Monitoramento de Processos Lógicos.

Área princial: SIM – Simulação

ABSTRACT

This article presents CronoSim, a tool that allows the modeling and running of the simulation of discrete events in a distributed infraestructure using a parallel machine or a distributed system, providing better performance. It also provides resources to manage the logical processes of the program, allowing to optimize the run with resources of load balancing and processes scheduling. In addition, this tool helps users without experience in parallel computing, on the parallelization of the simulation model.

KEYWORDS. Distributed Simulation. Simulation Tool. Monitoring of Logical Processes.

Main area: SIM – Simulation

1. Introdução

Segundo Jagtap et al. (2012), a Simulação de Eventos Discretos (SED) é um tipo de simulação utilizado para estudar sistemas onde as mudanças de estado são discretas. Ela vem sendo empregada amplamente como ferramenta de auxílio a tomadas de decisões em diversas áreas. De acordo com a complexidade dos modelos, a simulação exige a manipulação de grande quantidade de dados para sua execução, sendo estes dados responsáveis pela representação do sistema real. Desta maneira, quanto maior o sistema a ser simulado, maior é o volume de dados requerido para sua simulação. A necessidade de simulações de sistemas grandes e complexos, que demandam a alocação de muitos recursos computacionais, pode tornar desvantajosa a execução de um programa de simulação em um ambiente sequencial.

A fim de minimizar o tempo de execução da simulação, foram adotadas abordagens paralelas com o objetivo de melhorar o desempenho e assim diminuir o tempo de execução da simulação. No cenário paralelo, os processos da simulação são divididos em vários processos menores, sendo chamados de processos lógicos, e cada um desses processos lógicos é enviado ao nó de uma máquina paralela ou de um sistema distribuído para que, dessa forma, seja executado em paralelo aos demais. Neste contexto, o conceito de Simulação Distribuída vem sendo desenvolvido com o objetivo de diminuir o tempo de execução de um programa de simulação (BRUSCHI *et al.*, 2004; JAGTAP *et al.*, 2012).

A realização de uma simulação, mesmo em um ambiente sequencial, é uma tarefa difícil, pois exige do usuário conhecimentos sobre: modelagem, programação e matemática (probabilidade e estatística), tornando assim, para novos ou inexperientes usuários, árdua a tarefa de criação, validação, execução e análise dos resultados obtidos.

Em um ambiente distribuído, essa atividade pode se tornar ainda mais difícil, já que, além dos conhecimentos citados anteriormente, é necessário que o usuário possua conhecimentos de Computação Paralela e de Sistemas Distribuídos, que não são fáceis nem rápidos de se obter.

Neste sentido, a utilização de um ambiente de simulação que apresente uma interface visual, suporte a execução distribuída de forma transparente e que permita aos seus usuários maior agilidade no processo de criação de modelos e facilidade para execução da simulação em um ambiente distribuído, é de grande interesse.

Da mesma forma, o desenvolvimento de ferramentas para o monitoramento de processos permite que os recursos alocados nos nós sejam monitorados e, eventualmente, reescalonados, com o objetivo de aumentar o desempenho do programa de simulação.

Procurando oferecer uma maior agilidade e facilidade para os usuários da simulação distribuída, este artigo apresenta uma ferramenta de simulação distribuída de eventos discretos denominada CronoSim. Esta ferramenta auxilia o usuário desde a fase de modelagem até o monitoramento dos processos, tratando, principalmente, dos aspectos relacionados com o desempenho da simulação, através do balanceamento de carga do sistema e da possibilidade de controlar os mecanismos de migração de processos. A ferramenta também permite interromper a simulação em tempo de execução para verificar os resultados parciais e, também, verificar os estados dos processos lógicos do programa de simulação.

Este artigo está estruturado da seguinte forma: a próxima seção apresenta a fundamentação teórica, focando em aspectos de simulação distribuída, a seção 3 abrange a especificação da ferramenta e suas principais características, a seção 4 traz os experimentos realizados utilizando o CronoSim e os resultados obtidos, a seção 5 disserta sobre trabalhos relacionados e, por fim, a última seção conclui o artigo.

2. Fundamentação Teórica

2.1. Simulação Distribuída

A simulação distribuída consiste na divisão do programa de simulação em processos lógicos menores que serão alocados e executados de forma que as operações tenham condições de serem processadas paralelamente. A implementação de uma simulação distribuída encontra obstáculos não existentes em um sistema centralizado, tais como, sincronização dos processos, balanceamento de carga e a sobrecarga na rede de comunicação (YILMAZ *et al*., 2014).

Segundo Yilmaz *et al*. (2014), em um ambiente de simulação distribuída, os processos lógicos devem ser executados de maneira que se evite a ocorrência de erros de causa e efeito, ou seja, a execução de eventos fora da ordem natural que seria obtida em um sistema centralizado. Para que isso ocorra, cada processo lógico tem seu próprio relógio lógico, que indica o progresso da simulação e, como não há um ambiente de memória compartilhada entre eles, as informações entre os processos são trocadas através de mensagens.

Uma maneira de garantir a ordem de execução cronológica dos eventos é utilizando protocolos que controlem o sincronismo entre os processos da simulação. Os protocolos de sincronismo que foram desenvolvidos são divididos em duas classes: conservativos e otimistas, evitando ou corrigindo erros de causa e efeito, respectivamente (PIENTA e FUJIMOTO, 2013).

Os primeiros protocolos desenvolvidos foram os conservativos. Basicamente, um evento é executado somente quando possuir o menor rótulo de tempo e não houver possibilidade do processo lógico receber algum evento com rótulo de tempo menor que ele (LIU, 2013). Nos protocolos otimistas, o sincronismo ocorre quando há identificação de um erro de causa e efeito. Nessa situação, os processos corrigem o problema através da restauração de um estado anteriormente armazenado, procedimento conhecido como *rollback*. Jefferson desenvolveu um mecanismo de sincronização denominado *Virtual Time* e propôs um protocolo que se tornou o protocolo otimista mais conhecido, chamado *Time Warp* (JEFFERSON, 1985; ALZRAIEE *et al*., 2012).

Apesar do *Time Warp* ser o protocolo otimista mais conhecido, Moreira (2005) apresentou um novo protocolo otimista, denominado *Rollback* Solidário e fundamentado na teoria dos *Checkpoints* Globais Consistentes, que utiliza a abordagem de linhas de recuperação para o sincronismo dos processos durante o procedimento de *rollback*, diminuindo o efeito cascata que ocorre durante o tratamento dos *rollbacks* pelos processos da simulação (MOREIRA; SANTANA; SANTANA, 2005). A ferramenta apresentada neste trabalho faz uso de ambos os protocolos: *Time Warp* e *Rollback* Solidário.

2.2. Monitoramento de Processos Lógicos na Simulação Distribuída

De acordo com Liu (2013), o escalonamento de processos é uma operação computacional que proporciona melhor desempenho em um sistema computacional distribuído. Para otimizar o desempenho, é indispensável uma distribuição equilibrada dos processos entre os processadores disponíveis. O escalonamento consiste, basicamente, na alocação de recursos computacionais para a execução de um programa formado por vários processos.

Segundo Gonsiorowski *et al*. (2012), um programa de simulação distribuída é afetado pela máquina paralela e pelas características do próprio modelo de simulação. Existem várias técnicas para otimizar o desempenho de um programa de simulação distribuída. As técnicas que mais se destacam são as de escalonamento e de balanceamento de carga. Neste contexto, algumas causas do desbalanceamento de carga em uma simulação distribuída, baseada nos protocolos otimistas, são:

- **avanço não homogêneo dos relógios lógicos:** os processos da simulação podem estar

executando em processadores com diferentes cargas e esta diferença pode levar ao avanço mais rápido de alguns processos da simulação. Os processos que se atrasam podem gerar *rollbacks* longos e frequentes;

- **influência de outras aplicações:** outras aplicações que concorrem com o uso dos recursos de *hardware* podem afetar o balanceamento de carga de um programa de simulação;
- **fatores internos da simulação:** a diferença de parâmetros entre os processos da simulação pode levar ao desbalanceamento de carga na simulação.

O monitoramento dos processos da simulação é essencial para que se obtenha dados necessários à tomada de decisão a respeito do escalonamento dos processos. A utilização de informações da simulação distribuída pode levar a um melhor desempenho quando é empregada para realizar o escalonamento de processos. Uma vez determinada uma situação de desbalanceamento, a migração de processos é utilizada para recuperar o equilíbrio na carga do sistema.

Por conseguinte, um programa de simulação distribuída pode tirar proveito de um sistema de monitoramento. Informações como o estado dos processos lógicos, quantidade de *rollbacks* realizados, processos ativos ou em migração, dados sobre o funcionamento da rede de interconexão, entre outras, podem ser usados para múltiplas tarefas, incluindo: detecção de falhas, escalonamento, depuração e análise dos resultados parciais da simulação. Além disso, é possível armazenar um histórico das execuções do programa, a fim de analisar o comportamento do sistema ao longo de um determinado período de tempo ou utilizar esses dados nos algoritmos de mapeamento dos processos da simulação, para que, em uma nova execução, o mapeamento possa ser melhor definido, diminuindo a necessidade de migrações durante a simulação.

3. A Ferramenta Desenvolvida

A ferramenta apresentada neste artigo está relacionada com outras pesquisas que têm sido desenvolvidas para proporcionar melhor desempenho em simulações. Neste contexto, Moreira *et al.* (2010) propuseram um *framework* para o desenvolvimento de aplicações de simulação distribuída, possuindo um conjunto de classes personalizadas e abstratas desenvolvidas para solucionar diversos problemas encontrados em ambientes distribuídos como, por exemplo, a escolha do protocolo de sincronismo e a possibilidade de trocar dinamicamente o protocolo durante a simulação. Dessa forma, tendo este *framework* como base, foi especificado um ambiente para simulação de eventos discretos com o objetivo de prover recursos para melhorar o desempenho da simulação, além de auxiliar o usuário iniciante em ambientes computacionais distribuídos a entender o funcionamento da arquitetura e conseguir utilizar, de uma maneira eficiente, os recursos que este tipo de plataforma pode oferecer.

As características específicas deste projeto levaram a separação de seus componentes em camadas e criou-se 3 categorias:

1. As questões relacionadas com computação paralela/distribuída, envolvendo os problemas inerentes ao escalonamento de processos, consistência dos dados e erros de causalidade na sincronização dos processos.

2. As questões relacionadas à aplicação em si, ou seja, modelagem, simulação e análise dos resultados.

3. A integração do programa de simulação em um ambiente distribuído com a interface de comunicação com o usuário da aplicação.

O objetivo desta arquitetura é separar as tarefas de acordo com suas pertinências. A estruturação da ferramenta segue o modelo em camadas definido para o *framework* desenvolvido no trabalho de Moreira *et al*. (2010).

Neste trabalho, foi utilizada a linguagem Java para a implementação da ferramenta. Os motivos desta escolha são as principais características desta linguagem. A linguagem Java é uma das mais poderosas ferramentas da atualidade para o desenvolvimento de vários tipos de aplicações. Isto porque ela é segura, estável, oferece uma enorme biblioteca de programação e é uma linguagem totalmente orientada a objetos. Além disso, a linguagem Java é caracterizada por sua independência de plataforma. Isso significa que o mesmo programa compilado pode ser executado em qualquer sistema operacional. Portanto, uma outra vantagem é a possibilidade de programas serem executados em uma rede de computadores com diferentes sistemas operacionais (DEITEL e DEITEL, 2014).

As subseções seguintes apresentam as principais características do projeto utilizando a UML (*Unified Modeling Language*) para especificação, em especial, o diagrama de classes.

3.1. Diagrama de Classes

A Figura 1 apresenta as principais classes do projeto e suas relações, onde é possível observar as diferentes camadas construídas. O pacote *editor* contém todas classes referentes ao ambiente visual e o pacote *model* abrange os elementos pertinentes aos dados da simulação. Os métodos correspondentes à execução da simulação distribuída estão no *framework* que utiliza os protocolos *Timewarp* e *Rollback Solidário*.

Na camada de visão, a classe *SimulEditor* é responsável por iniciar a ferramenta e realizar chamadas de métodos que configuram a interface gráfica com o usuário. As classes *EditorToolBar*, *EditorKeyboardHandler*, *EditorPopupMenu* e *EditorMenuBar* contém componentes para manipular as interações com usuário e ativar uma determinada ação no CronoSim. Por exemplo, quando um usuário deseja salvar um modelo de simulação, ele deverá interagir com a interface visual de alguma forma a fim de ativar a ação "salvar modelo". Todas ações ativadas pelo usuário são implementadas em subclasses de *EditorActions*.

A classe PerspectiveSwitch implementa a troca de perspectiva do ambiente de simulação, ou seja, possui métodos que controlam o que deve ser mostrado na tela principal. Através deste mecanismo é possível, por exemplo, durante a modelagem da simulação mudar para a tela onde é definido como será a arquitetura e configuração da rede.

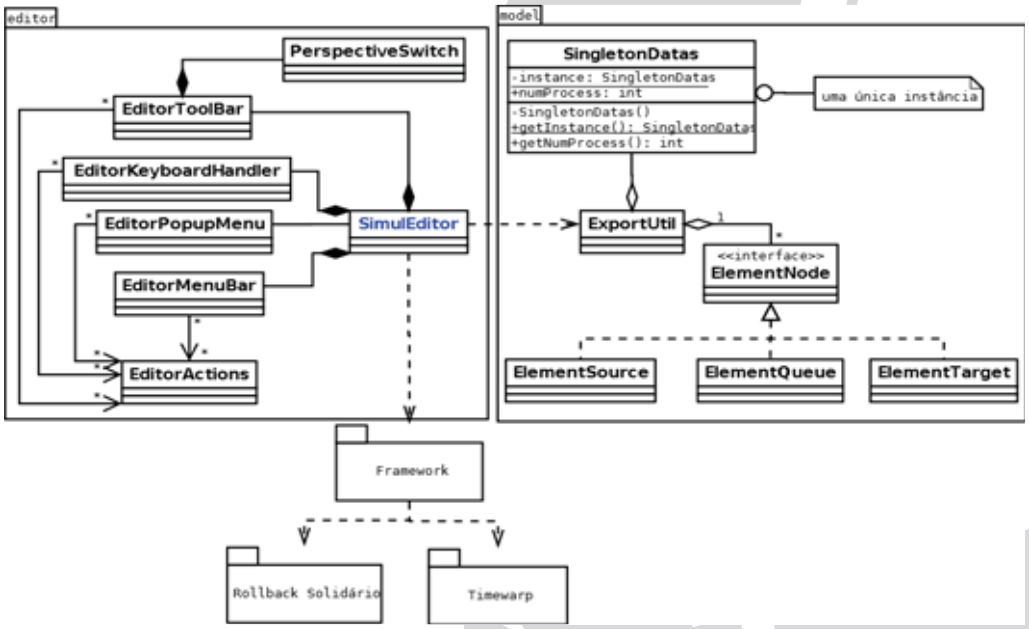

Figura 1. Diagrama de classes da ferramenta

O pacote *model* gerencia os dados da modelagem e resultados da simulação. As classes *ElementSource*, *ElementQueue* e *ElementTarget* representam os elementos do modelo de simulação baseado em Redes de Filas. A interface *ElementNode* foi criada para permitir futuras implementações referentes a outras técnicas de modelagem.

Tanto os dados do modelo criado pelo usuário quanto o resultado da simulação são tratados pela classe *ExportUtil*, que possui métodos para tradução da modelagem visual em códigos que o simulador possa interpretar, e também contém métodos para exportar dados a respeito do resultado. Além disso, foi utilizado um padrão de projeto conhecido por *Singleton*, que tem o objetivo de garantir que exista apenas uma instância de uma certa classe a qualquer momento. Este padrão foi aplicado à classe *SingletonDatas* para armazenar dados do modelo que devem ser únicos durante todo o processo da simulação.

3.2. Ambiente para Modelagem

CronoSim oferece um ambiente para modelagem de sistemas, permitindo ao usuário descrever o modelo utilizando Redes de Filas como forma de representação. Entretanto, a ferramenta está preparada para receber futuras implementações relacionadas a outras técnicas de modelagem, devido ao uso de classes abstratas e interfaces.

A Figura 2 apresenta um exemplo da especificação de um modelo baseado em Redes de Filas. Chwif e Medina (2014) explicam algumas características desta técnica de modelagem. Os componentes básicos desta representação são: **servidores**, **filas** e **clientes** (eventos). Os dados modelados do sistema correspondem aos servidores, os quais prestam serviços aos clientes. A fila é um campo de espera para clientes que não são atendidos prontamente por um serviço requisitado devido ao(s) servidor(es) estar(em) ocupado(s). O conjunto de um ou mais servidores e uma ou mais filas é denominado centro de serviço. Nesse sentido, uma rede de filas é composta por um conjunto de centros de serviço.

Utilizando o ambiente de modelagem, é possível adicionar **fila** (centro de serviço), **início** (gerador de eventos) e **fim** (saída do sistema). Um centro de serviço é um processo lógico e possui uma taxa de chegada de eventos que é definida por geradores externos conectados a ele. Além disso, podem existir centros de serviço em séries, ou seja, quando existe probabilidade dos eventos saírem de um e entrarem em outro. A ferramenta permite utilizar as seguintes variações de centros de serviço: (1) com uma fila e um servidor, (2) com uma fila e vários servidores e (3) com várias filas e um servidor.

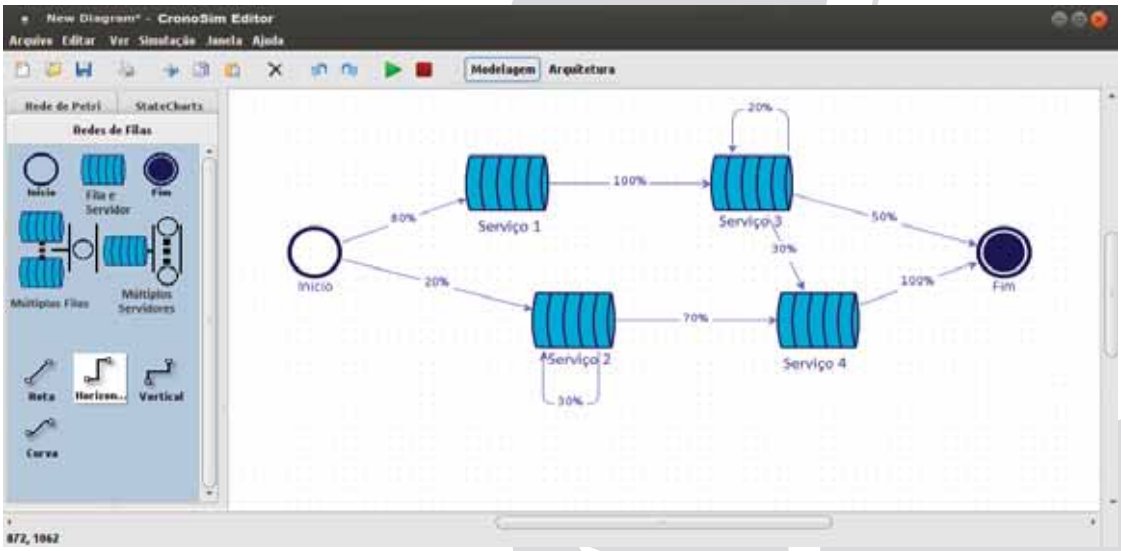

Figura 2. Ambiente gráfico de modelagem utilizando Redes de Filas

O usuário pode definir essas taxas após adicionar os componentes e configurar as conexões entre eles, através da aba propriedades, onde são exibidos os atributos do componente selecionado. Também é possível conectar as **filas** ao componente **fim** que indica a probabilidade dos eventos serem finalizados, ou seja, o elemento **fim** calcula a probabilidade do evento não entrar em outra fila.

3.3. Configuração da Arquitetura do Ambiente de Simulação

A Figura 3 exibe um exemplo de arquitetura de um ambiente computacional para ser utilizado na simulação distribuída. A arquitetura define as características dos *hardwares,* sendo possível adicionar estações (máquinas) e *switches* e conectá-los para representação da rede.

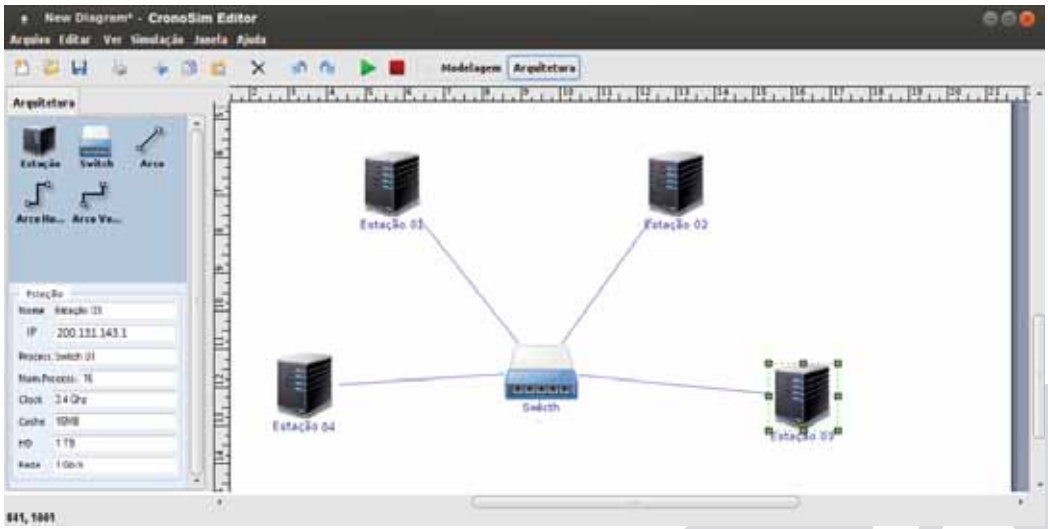

Figura 3. Configuração da arquitetura

O usuário pode definir os atributos pertinentes ao nó selecionado através do menu de configuração do componente à esquerda do editor. Dentre eles estão: endereço IP, o rótulo, processador, capacidade do *clock*, *cache*, disco e velocidade da rede. Através desta interface, o usuário poderá fazer o monitoramento dos processos da simulação durante a sua execução.

É importante destacar que o modelo criado para representar a configuração da arquitetura deve ser fiel às características do ambiente computacional onde será executada a simulação, caso contrário, os algoritmos de escalonamento não funcionarão adequadamente.

3.4. Execução da Simulação

Após a criação do modelo e a configuração da arquitetura, será possível executar a simulação distribuída. A Figura 4 ilustra o processo utilizando a interface gráfica, onde se identifica a transparência ao usuário, que não precisa escrever linhas de código nem conhecer os mecanismos particulares de simulação distribuída.

Figura 4. Processo de Simulação Distribuída utilizando o Ambiente Gráfico

Os dados do modelo gerado pelo usuário é utilizado juntamente com métodos do *framework* empregado para construção da ferramenta, o que permite a execução da simulação no ambiente distribuído.

A simulação é distribuída em um *cluster* utilizando a biblioteca de troca de mensagens *MPI* (*Message Passing Interface*) para comunicação entre os processos divididos entre os nós (BARNEY, 2014). Durante a simulação, o usuário pode visualizar informações referentes aos processos que estão sendo executados e após o seu término é possível analisar os resultados da simulação.

3.5. Mapeamento de Processos

Durante a execução da simulação, existe um sistema de monitoramento que coleta dados necessários à tomada de decisão a respeito do escalonamento dos processos. Estes dados são apresentados ao usuário através da arquitetura disposta no editor gráfico. Ao selecionar algum nó da arquitetura, são fornecidas informações a respeito dos processos que estão sendo executados nele.

Neste contexto, com o mapeamento dos processos, é possível obter dados que auxiliam o usuário na análise dos processos em execução. Quando uma situação de desbalanceamento é identificada, podem ser utilizados outros mecanismos que fazem a migração de processos, visando um melhor desempenho da simulação.

4. Experimentos realizados utilizando o CronoSim

A análise realizada neste artigo enfatiza o desempenho alcançado pela simulação em uma arquitetura distribuída de computadores e não os resultados estatísticos dos modelos. Esta escolha justifica-se pela necessidade de ilustrar, neste momento, as vantagens de se utilizar esta ferramenta na simulação de modelos grandes e complexos, ou seja, que apresentam um número elevado de componentes com várias conexões entre si.

Os experimentos utilizando o CronoSim foram executados em um *cluster* customizado da Universidade Federal de Itajubá composto por cinco máquinas Core 2 Duo CPU Q6600 @ 2.40GHz (L2, 4Mb), 4Gb DDR III, Rede Ethernet 10/100 com sistema operacional Linux. O objetivo é estabelecer um comparativo de desempenho entre simulações realizadas em um ambiente sequencial e simulações executadas de forma distribuída.

4.1. Modelos Simulados

Os experimentos foram realizados com modelos de 4, 50 e 100 processos lógicos, construídos e validados pelo CronoSim, de modo a calcular o tempo gasto para executar 10.000 eventos, para cada modelo. Nos três modelos criados para os testes, todas as filas tratavam de chegadas e atendimentos marcovianos com um único centro de atendimento (Notação Kendall-Lee M/M/1).

As simulações foram realizadas em um *cluster* com cinco máquinas. Cada modelo foi simulado em 5 arranjos de processadores (de 1 até 5) de modo a calcular os respectivos tempos médios de execução e desvios-padrões ao longo de 10 replicações para cada experimento. Entretanto, o modelo de 4 processos não necessitou de testes com 5 processadores, pois utilizaria, no máximo, 4 destes.

4.2. Resultados

O cálculo do *speedup* e da eficiência das configurações paralelas, em relação à execução sequencial, demonstra que por mais que a computação distribuída contribua com certa melhoria de desempenho na obtenção dos resultados da simulação, o *overhead* de mensagens sempre se faz presente, levantando questões como: balanceamento de carga e escalonamento de processos.

Segundo Tang *et al.* (2012), estas duas métricas, *speedup* e eficiência, são definidas respectivamente como a divisão do tempo gasto pela simulação no ambiente sequencial ($T_{sequencial}$) pelo tempo gasto pela simulação paralela ($T_{paralelo}$) e, a divisão do *speedup* pelo número de processadores. O resultado ideal do *speedup* seria que seu valor aumentasse na mesma proporção do aumento do número de processadores, ao passo que o valor ideal da eficiência deve ser 100%.

As tabelas a seguir apresentam os resultados da simulação distribuída. Os critérios adotados foram: o Tempo Médio da Simulação (TMS), que representa a média do tempo que cada processo levou para atingir os resultados esperados; o Tempo da Simulação (TSIM), que indica a quantidade de tempo que a simulação levou para executar, isto é, o tempo do processo que mais demorou para atingir seus resultados esperados e; por fim, os respectivos Desvios Padrões (DP) do TMS e do TSIM. O tempo apresentado na tabela está descrito em segundos.

A simulação do modelo de 4 processos, apesar de simples, foi criado para representar o diagrama de filas da Figura 2. Conforme os resultados apresentados na Tabela 1, pode-se notar que o ganho de desempenho deste modelo não foi significativo. Isto se deve às características do modelo que possui filas em série, criando uma dependência na comunicação entre os respectivos processos lógicos.

Tabela 1. Comparativo de desempenho da simulação distribuída em diferentes números de máquinas

Modelo	Nº de nós	TMS	DP (TMS)	TSIM	DP (TSIM)
4 processos	1	101	0,71	184	1,41
	2	98	0,71	180	0,96
	3	98	0,96	179	0,82
	4	97	0,75	179	0,89
	5	-	-	-	-
50 processos	1	230	4,95	1987	47,38
	2	191	4,18	1505	47,15
	3	194	5,69	1690	65,19
	4	190	5,16	1534	58,32
	5	187	5,32	1412	56,36
100 processos	1	89	2,83	412	44,55
	2	76	1,94	365	32,27
	3	75	2,16	390	41,63
	4	74	3,87	395	50,91
	5	74	1,41	370	30,41

Naturalmente, apesar da possibilidade de simular modelos com poucos componentes, as vantagens do CronoSim aparecem na simulação de modelos maiores. Para ilustrar este cenário,

foram criados dois modelos com 50 e 100 centros de serviços respectivamente. Como explicado anteriormente, cada centro de serviço foi tratado como um processo lógico do sistema distribuído. Os modelos foram criados respeitando-se as taxas de nascimento e de serviço do modelo M/M/1 e distribuindo-se, aleatoriamente, as conexões entre os elementos. Apesar de possuir o dobro de processos, o modelo com 100 centros de serviços foi gerado com uma quantidade significativamente menor de ligações entre os elementos, cerca de 50% do total de ligações do modelo com 50 processos. Esta modificação, ilustra o impacto da comunicação, mas ilustra o potencial de paralelismo da ferramenta.

A Tabela 2 mostra o *speedup* e a eficiência das execuções paralelas em relação ao experimento sequencial para avaliar o quanto a simulação distribuída traz ganhos em termos de desempenho para a simulação.

Tabela 2. *Speedup* e eficiência sobre a simulação

Modelos	Número de processadores	*TMS*		*TSIM*	
		Speedup	Eficiência	*Speedup*	Eficiência
4 processos	2	1,03	51,5%	1,02	51,2%
	3	1,03	34,3%	1,03	34,3%
	4	1,04	25,9%	1,03	25,7%
	5	-	-	-	-
50 processos	2	1,18	66,0%	1,32	66,0%
	3	1,19	39,2%	1,18	39,2%
	4	1,21	32,4%	1,30	32,4%
	5	1,21	28,1%	1,41	28,1%
100 processos	2	1,18	58,9%	1,13	56,4%
	3	1,19	39,8%	1,06	35,2%
	4	1,21	30,3%	1,04	26,0%
	5	1,21	24,2%	1,11	22,3%

Os resultados demonstram que essas métricas não atingem seus respectivos valores ideais, por conta do *overhead* de comunicação entre as máquinas e das conexões em série dos modelos, o que afeta diretamente o cálculo da eficiência. Mesmo assim, a diminuição do tempo total da simulação de modelos grandes é significativa, o que permite obter melhor desempenho que um programa de simulação sequencial.

5. Trabalhos Relacionados

Vários ambientes e ferramentas de simulação são encontrados na literatura, sendo a maioria de uso comercial. Exemplos destas ferramentas estão: ASDA (BRUSCHI *et al.*, 2004), Simul8 (CHWIF e MEDINA, 2006), SIPPO (Peixoto *et al.*, 2010), ExtendSim (KRAHL, 2013), ProModel (KHALILI e ZAHEDI, 2013), Simio (PEGDEN e STURROCK, 2013), Ururau (PEIXOTO *et al.*, 2013), SAS *Simulation Studio* (HUGHES *et al.*, 2014), Arena (KRANZ e ZUPICK, 2014) entre outros *softwares* disponíveis conforme uma pesquisa do Instituto de Pesquisa Operacional e Ciências da Administração - INFORMS (SWAIN, 2013).

Entretanto, a maioria destas ferramentas não suporta simulação distribuída. Além disso, nenhuma permite o gerenciamento dos processos durante a simulação. Desse modo, a ferramenta apresentada neste artigo auxilia o desenvolvimento de simulações em um ambiente distribuído. Seu principal diferencial, em relação às demais já existentes no mercado e também na academia, está na possibilidade do usuário gerenciar os processos lógicos do programa, permitindo otimizar

a execução com recursos de balanceamento de carga e escalonamento de processos, além de fornecer recursos para a modelagem e a execução da simulação.

6. Conclusões

Este artigo apresentou CronoSim, uma ferramenta que fornece aos usuários de simulação de eventos discretos recursos para utilizarem os benefícios da computação paralela na execução de grandes simulações. Isto elimina a necessidade de conhecimentos aprofundados na área de sistemas distribuídos para realizar as simulações.

A abordagem distribuída permite um aumento de desempenho da simulação em comparação com a execução de um programa de simulação sequencial. Além disso, a utilização de monitoramento dos processos possibilita a intervenção dos algoritmos de balanceamento de carga, a fim de se obter um melhor equilíbrio na carga dos processadores. Dessa forma, o desenvolvimento do CronoSim fornece uma importante contribuição para a área de simulação, permitindo que os usuários realizem simulações que demandam maior esforço computacional de forma eficiente.

Destaca-se que esta ferramenta está preparada para futuras implementações que possibilitarão a execução em uma arquitetura computacional mais dinâmica e flexível, como a utilização de *Cloud Computing*, que elimina a necessidade de adquirir equipamentos de computação de alto custo, tornando a simulação distribuída cada vez mais vantajosa aos usuários de simulação.

Agradecimentos

À CAPES pelo apoio financeiro recebido.

Referências

Alzraiee H.; Zayed T. e Moselhi O. Methodology For Synchronizing Discrete Event Simulation And System Dynamics Models. In: *Proceeding of the 2012 Winter Simulation Conference,* p. 606-616, 2012.

Barney, B. Message Passing Interface (MPI). *Training Materials - Lawrence Livermore National Laboratory*. Disponível em: computing.llnl.gov/tutorials/mpi [Acessado em: 20 de Abril de 2015].

Bruschi, S. M.; Santana, R. H. C.; Santana, M. J. e Aiza, T. S. An Automatic Distributed Simulation Environment. In: *Proceedings of the 2004 Winter Simulation Conference*. p. 378-385, 2004.

Chwif, L.; Medina, A.C. Modelagem e Simulação de Eventos Discretos: Teoria e Aplicações. São Paulo: Ed. dos Autores, 2014.

Chwif, L. e Medina, A. C. Introdução ao software de simulação Simul8. In: *Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional*, Goiânia, GO, 2006.

Deitel, H. M. e Deitel, P. J. *Java – How To Program*. 10th ed. Prentice Hall, 2014.

Gonsiorowski, E.; Carothers, C. e Tropper, C. Modeling Large Scale Circuits Using Massively Parallel Discrete-Event Simulation. In: *Proceedings of the 20th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. p. 127-133, 2012.

Hughes, E.; Lada, E.; Meanor, P. e Chen, H. Introduction To SAS Simulation Studio. In: *Proceedings of the 2014 Winter Simulation Conference.* p. 4202- 4212, 2014.

Jagtap, D.; Abu-Ghazaleh, N. e Ponomarev D. Optimization of Parallel Discrete Event

Simulator for Multi-core Systems. In: *Proceedings of the 26th International Parallel and Distributed Processing Symposium*, p. 520-531, 2012.

Jefferson, R. D. Virtual time. *ACM Transactions on Programming Languages and Systems*. v.7, n. 3, 405-425, 1985.

Khalili, M. H. e Zahedi F. Modeling and Simulation of a Mattress Production Line Using Promodel. In: *Proceedings of the 2013 Winter Simulation Conference.* p. 2598-2609, 2013.

Krahl, D. Extendsim 9. In: *Proceedings of the 2013 Winter Simulation Conference*, p. 4065-4072, 2013.

Kranz, R. e Zupick N. Arena Simulation Software: Introduction And Overview. In: *Proceedings of the 2014 Winter Simulation Conference*. p. 4213, 2014.

Liu, J. Real-time scheduling of logical processes for parallel Discrete-event simulation. In: *Proceedings of the 2013 Winter Simulation Conference*. p. 2959- 2971, 2013.

Moreira, E. M. *Rollback Solidário: Um novo protocolo otimista para Simulação Distribuída*. Tese (Doutorado) – Universidade de São Paulo, 2005.

Moreira, E. M.; Carpinteiro, O. A. S.; Seraphim, E.; Seraphim, T. F. P. e Cruz, L. B. Um framework para o desenvolvimento de programas de simulação distribuída. In: *Anais do XLII Simpósio Brasileiro de Pesquisa Operacional*. Bento Gonçalves-RS, 2010.

Moreira, M. E., Santana, R. H. C. e Santana, M. J. Using consistent global checkpoints to synchronize processes in distributed simulation. In: *Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real Time Applications*. 43-50, 2005.

Peixoto, T. A., Rangel, J. J. A. e Matias, I. O. SIPPO 0.1- Simulador Para Problemas de Pesquisa Operacional. In: *Anais do XLII. Simpósio Brasileiro de Pesquisa Operacional*. Bento Gonçalves-RS, 2010.

Peixoto T. A.; Rangel J. J. A.; Matias I. O.; Montevechi J. A. B. e Miranda, R. C. 2013 Ururau- Um Ambiente Para Desenvolvimento De Modelos De Simulação A Eventos Discretos. *Revista Eletrônica Pesquisa Operacional Para o Desenvolvimento*, v.5, n.3, p. 373-405, Rio de Janeiro, 2013.

Pienta, R. e Fujimoto, R. M. On the Parallel Simulation of Scale-Free Networks. In: *Proceedings of the 2013 ACM SIGSIM conference on Principles of advanced discrete simulation Pages 179-188*, 2013.

Pegden, C. D. e Sturrock, D. T. Introduction To Simio. In: *Proceedings of the 2011 Winter Simulation Conference*, p. 29- 38, 2011.

Swain J. J. INFORMS Simulation software survey. OR/MS Today. Institute for Operations Researchand the Management Sciences (INFORMS), USA, 2013. Disponível online: http://www.orms-today.org/surveys/Simulation/Simulation.html. [acessado em 20 de Abril de 2015].

Tang, S.; Lee, B. e He, B. Speedup for Multi-Level Parallel Computing. In: *Proceeding of the 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, p. 537-546, 2012.

Yilmaz, L; Taylor, S. J. E.; Fujimoto, R. e Darema, F. Panel: The Future of Research in Modeling & Simulation. In: *Proceedings of the 2014 Winter Simulation Conference*. p. 2797-2811, 2014.