

# SciMulator: Um Ambiente de Simulação de Workflows Científicos em Redes P2P\*

Jonas Dias<sup>1</sup>, Carla Rodrigues<sup>1</sup>, Eduardo Ogasawara<sup>1</sup>, Daniel de Oliveira<sup>1</sup>,  
Vanessa Braganholo<sup>2</sup>, Esther Pacitti<sup>3</sup>, Marta Mattoso<sup>1</sup>

<sup>1</sup>Programa de Engenharia de Sistemas e Computação – COPPE / UFRJ  
Caixa Postal 68.511, Rio de Janeiro, RJ, 21941-972

<sup>2</sup>Departamento de Ciência da Computação – UFRJ  
Caixa Postal 68.530, Rio de Janeiro, RJ, 21941-590

<sup>3</sup>INRIA & LIRMM, Montpellier, France

{jonasdias, carlarod, ogasawara, danielc, marta}@cos.ufrj.br

braganholo@dcc.ufrj.br

pacitti@lirmm.fr

**Abstract.** The growth of large-scale scientific experiments motivates the search for computing environments that support the parallelization of computing activities, particularly those that comply to the Many Task Computing (MTC) paradigm. Peer-to-Peer (P2P) environments can meet this demand due to the easy access and distributed control. However, building a real P2P infrastructure to evaluate this solution is very costly. Within this context, we present the simulator SciMulator developed to evaluate P2P architectures. We present the modeling of the simulator and an initial assessment of its performance when using the SciMule architecture for submission of scientific workflows activities on P2P networks.

**Resumo.** O crescimento dos experimentos científicos em larga escala motiva a busca por ambientes computacionais que apoiem a paralelização de atividades computacionais, particularmente, as que atendem o paradigma *Many Task Computing* (MTC). Ambientes *Peer-to-Peer* (P2P) podem atender esta demanda, pelo fácil acesso e controle distribuído. Porém, construir uma infraestrutura P2P real para avaliar tal solução é muito custoso. Neste contexto, apresentamos o simulador SciMulator, desenvolvido para avaliar arquiteturas P2P. Neste trabalho, apresentamos a modelagem do simulador e uma avaliação inicial do seu desempenho ao utilizar a arquitetura SciMule para submissão de atividades de workflows científicos em redes P2P.

## 1. Introdução

Recentemente, o avanço do processamento de alto desempenho motivou a realização de experimentos científicos (Deelman et al. 2009). Tais experimentos são caracterizados pela grande movimentação de dados, dados heterogêneos e execução de um grande número de atividades com potencial de paralelização. Nestes experimentos, o encadeamento de atividades é popularmente modelado como um workflow científico

---

\* Esse trabalho foi parcialmente financiado pelo CNPq e pelo INRIA dentro do projeto SARAVA, equipe associada.

(Brown et al. 2007). As atividades podem requerer alta complexidade computacional, o que faz com que a paralelização torne-se necessária para realização do experimento em tempo hábil. Em muitos casos, esta paralelização pode ocorrer por meio de varredura de parâmetros e fragmentação de dados. Estas características se encaixam no novo paradigma de computação conhecido como *Many Task Computing* (MTC) (Raicu et al. 2008).

Existem soluções para paralelizar atividades do workflow seguindo o paradigma MTC e executá-las em ambientes homogêneos como *clusters* (Abramson et al. 2008, Ogasawara et al. 2009). Entretanto, existem muitos outros ambientes heterogêneos que podem ser explorados para execução de workflows científicos como, por exemplo, *grids*, *desktop grids* (Anderson 2004) ou nuvens híbridas (Grid4All Consortium 2009). O principal problema é que cada um destes ambientes requer diferentes esforços, recursos e habilidades do cientista para definir e avaliar o paralelismo nas atividades do workflow.

A abordagem P2P também se mostra promissora para o processamento de atividades MTC pela alta escalabilidade, capacidade de lidar com distribuição de dados, além da tolerância a falhas. Segundo Pacitti et al. (2007), técnicas P2P também se mostram úteis em *grids* computacionais de larga escala. Recentemente, iniciativas como SciMule (Ogasawara et al. 2010) propõem a paralelização de atividades de workflows científicos em redes *peer-to-peer* (P2P). No entanto, faltam estudos que avaliem soluções para o processamento de atividades MTC de workflows científicos em redes P2P.

Uma rede P2P envolve milhares de computadores com arquiteturas distintas interligadas por uma rede também heterogênea. Construir uma infraestrutura real para realizar estudos iniciais de viabilidade é muito custoso e inviável para a grande maioria dos pesquisadores (Almeida et al. 2008). Portanto, antes de se realizar estudos em uma rede real, é necessário avaliar se há indícios de que a distribuição de atividades de workflows científicos em redes P2P é realmente vantajosa e em quais cenários há essa vantagem, uma vez que devem ser consideradas diversas características como latência entre pontos da rede e a dificuldade de transmissão de dados através de pontos com baixa largura de banda. A avaliação destes indícios pode ser feita através de estudos de simulação.

O objetivo deste trabalho é apresentar o modelo construído para o ambiente de simulação SciMulator, ressaltando os desafios encontrados, tais como: a modelagem de atividades MTC de um workflow; sua decomposição e escalonamento em uma rede com controle distribuído; seu processamento e retorno de resultados; captura de dados de proveniência (Davidson and Freire 2008); além do suporte à tolerância a falhas. Tais características não estão presentes em simuladores P2P tradicionais. O SciMulator foi construído a partir do PeerSim (Jelasity et al. 2010), um simulador de ambientes P2P muito utilizado em outros trabalhos como base na construção de outros cenários de pesquisa (Boudani et al. 2008, Dick et al. 2009). No SciMulator, cada nó da rede é capaz de submeter atividades de workflows científicos para serem executadas distribuídas pelos outros nós da rede. As atividades são decompostas em tarefas seguindo estereótipos comuns no paradigma MTC.

Numa primeira avaliação, o SciMulator foi utilizado para avaliar a arquitetura do SciMule (Ogasawara et al. 2010). Ele mostrou-se ágil na simulação de cenários custosos e robusto no consumo de memória. Os resultados obtidos com o simulador foram positivos, uma vez que foi possível identificar componentes na arquitetura que precisam

ser aperfeiçoados, como, por exemplo, o escalonador de tarefas, o sistema de busca dos dados do experimento e o mecanismo de distribuição destes dados para execução de atividades, bem como possibilitar o desenvolvimento e avaliação de outras arquiteturas para apoiar MTC em ambientes P2P.

Este artigo está organizado da seguinte forma: a seção 2 descreve o SciMulator, um ambiente de simulação de workflows científicos em redes P2P, desenvolvido sobre o PeerSim; a seção 3 descreve sucintamente a arquitetura SciMule utilizada como primeiro estudo de caso do SciMulator; a seção 4 apresenta a análise de resultados; a seção 5 apresenta os trabalhos relacionados e a seção 6 conclui este artigo e apresenta trabalhos futuros.

## 2. SciMulator

O SciMulator é um simulador de um ambiente P2P onde cada nó da rede (*peer*) é capaz de submeter e executar tarefas distribuídas de atividades de workflows científicos seguindo o modelo MTC. O objetivo do simulador é permitir avaliações iniciais de arquiteturas voltadas ao processamento de atividades em redes heterogêneas com controle distribuído como as redes P2P. Por estas características, além do elevado grau de dinamismo, a solução P2P para processamento de tarefas deve se comportar de forma diferente de outras soluções para *clusters* e *grids*.

O SciMulator foi desenvolvido como uma extensão do PeerSim (Jelasity et al. 2010). O PeerSim é um simulador de redes P2P desenvolvido na linguagem Java. Ele possui quatro componentes principais extensíveis: (i) o nó, que representa uma máquina (*peer*) na rede P2P e é modelado pela classe *GeneralNode*; (ii) uma abstração do *overlay* que mantém informação sobre como os *peers* estão conectados; (iii) protocolos que são executados pelos nós e (iv) elementos de controle que executam ações globais na rede, tais como eventos de *churn* e registro de *log*. Estes componentes foram estendidos e outros foram criados para apoiar o modelo de submissão e execução de tarefas. Um desafio na construção do simulador foi, justamente, acrescentar componentes que permitissem a distribuição de atividades em tarefas MTC pela rede P2P, além de sua execução nos nós através de protocolos PeerSim. O PeerSim não oferece um modelo de controle de largura de banda dos *peers*, nem um sistema de *download* e *upload* entre eles. Portanto, o processo de transferência de dados também precisou ser modelado, pois influencia no tempo total de execução de uma atividade. Na Internet, a largura de banda entre nós da rede não pode ser prevista. O SciMulator busca simular esta característica estabelecendo valores estocásticos para a largura de banda disponível para cada nó da rede.

O SciMulator fornece um arcabouço base para avaliação de cenários e arquiteturas para submissão e paralelização de atividades MTC em redes P2P. Avaliar uma arquitetura no SciMulator significa avaliar a execução de um conjunto de protocolos pelos nós da rede dispostos em uma determinada topologia. Tal avaliação envolve a possível extensão de componentes referentes aos nós, à topologia e a alguns protocolos. Além dos componentes gerais do PeerSim, todo arcabouço de fragmentação e distribuição de atividades de workflow, escalonamento de tarefas, transmissão de dados, processamento, monitoramento, proveniência e recuperação de falhas pode ser reutilizado para modelar outras arquiteturas P2P para execução de workflows.

## 2.1 Submissão de Atividades

No SciMulator, os *peers* podem submeter atividades de workflows científicos na rede. Para isso, foi modelado um escalonador de tarefas que é responsável por todo o processo de decomposição e distribuição das atividades. Este componente entra em ação em um *peer* no momento em que este possui uma atividade para submeter.

Atividades são decompostas em tarefas de acordo com o seu tipo de paralelismo atrelado a MTC. Uma primeira implementação do SciMulator permite avaliação de atividades do tipo fragmentação de dados (*DataFragmentation*), nas quais as tarefas processam um fragmento do conjunto de dados da atividade, e varredura de parâmetros (*ParameterSweep*), nas quais as tarefas têm diferentes conjuntos de parâmetros para processar. O escalonador seleciona os *peers* para processar as tarefas considerando o balanceamento de carga. Características como o tamanho da fila de tarefas que cada *peer* possui, dados a serem transmitidos, uso do processador e a banda de rede utilizada, são levadas em consideração para que não haja sobrecarga em pontos da rede. O *peer* que submeteu a atividade também pode ser eleito para executar algumas tarefas de sua atividade. A Figura 1 apresenta um trecho do modelo UML do SciMulator exibindo os componentes do processo de submissão de uma atividade.

Os *peers* recebem tarefas de vizinhos, as quais precisam de um ou mais conjuntos de dados (*DataPackage*) para serem processadas. Se o *peer* não possuir os dados, ele efetua o *download* pelo nó que submeteu a atividade na rede. A transmissão de dados de uma tarefa é realizada através da troca de pacotes de dados simbólicos entre *peers* de acordo com a largura de banda do *link*. A taxa de *download* depende da velocidade do *link*, identificada pela menor largura de banda entre o cliente e o nó de execução. Um *peer* pode efetuar o *download* de várias tarefas simultaneamente. Assim, sua largura de banda é compartilhada entre todas as transmissões, segundo uma estratégia circular (*round-robin*). Os *peers* armazenam os pacotes de dados localmente, pois uma futura tarefa pode precisar deste mesmo conjunto de dados.

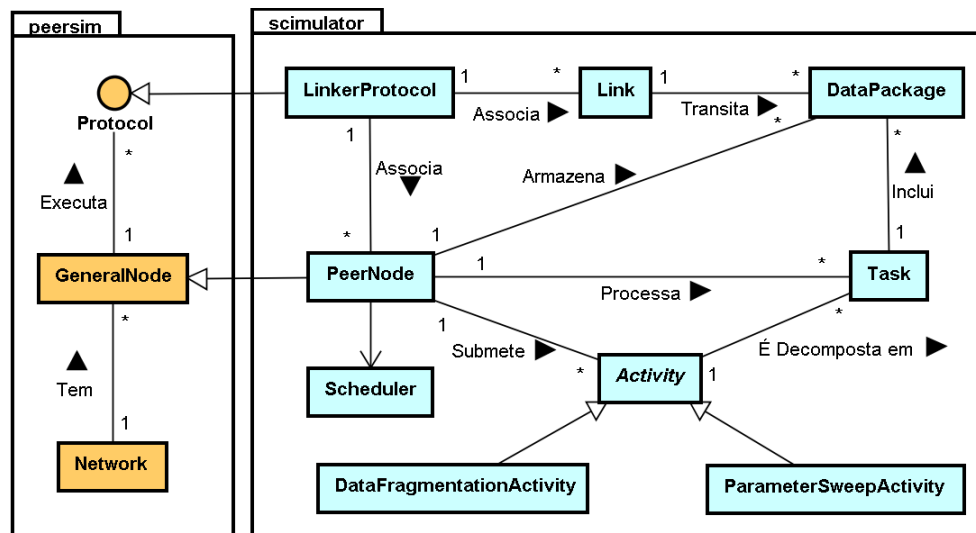


Figura 1: Modelo UML do SciMulator dos componentes de submissão

## 2.2 Execução de Atividades

Quando um *peer* detecta que foi escolhido para a execução de uma tarefa, ele verifica se precisa efetuar o *download* de algum conjunto de dados. Após obter todos os dados que compõem a tarefa, o *peer* começa a processá-la. Ele pode receber tarefas de diferentes

vizinhos, porém o protocolo de processamento é FIFO, isto é, as primeiras tarefas obtidas são as primeiras a serem processadas. Quando termina de processar a tarefa, o *peer* envia os resultados e informação de proveniência ao *peer* que submeteu a atividade.

A simulação do processamento de tarefas envolve o conceito de unidade de processamento. Cada *peer* possui  $x$  unidades de processamento por ciclo de simulação e cada tarefa possui um custo  $y$  de processamento. Sendo assim, uma tarefa deve demorar  $y/x$  ciclos para ser processada por um *peer* ocioso.

### 2.3 Configuração da Rede

Como o SciMulator é modelado para redes heterogêneas, durante o processo de configuração da rede, características como poder de processamento e largura de banda dos *peers* devem variar. Além disso, o *peer* não é uma máquina dedicada e tem seus recursos utilizados para outros fins. Ou seja, a capacidade total de processamento e largura de banda de um *peer* não é exclusiva da rede P2P. Portanto, ao longo da simulação, a medida de quanto uma máquina está ociosa em um determinado momento também varia. A porcentagem de quanto estes recursos estão disponíveis é estimada a cada ciclo. Os eventos que ocorrem durante a simulação, como a submissão de novas atividades e a entrada e saída de nós da rede (*churn*) também devem seguir uma regularidade variável. Para modelar este cenário, estudou-se um conjunto de distribuições estatísticas capazes de representar a variação destas características e eventos. Tais distribuições foram escolhidas com base no comportamento observado em experimentos reais realizados em clusters (Ogasawara et al. 2009) e parametrizadas para o modelo do simulador. A Tabela 1 mostra as distribuições escolhidas na modelagem do SciMulator. A capacidade de processamento de um *peer* varia seguindo uma distribuição Gamma, com fator de escala 30 e forma 2 (Freedman et al. 2007), com uma média de 80 unidades de processamento por ciclo. A largura de banda segue uma distribuição Gamma com média 1,5 sob uma função logarítmica de base dois definida de 7 (128 kbps) a 16 (64Mbps). A ociosidade da máquina segue uma distribuição normal com uma média de 0,5 e desvio padrão de 0,1875, indicando que cerca de cinquenta por cento dos recursos de cada *peer* estão disponíveis para a rede P2P em um determinado ciclo. As médias das distribuições de Poisson para Submissão de Atividades ( $x$ ) e Ocorrência de Churns ( $y$ ) são fatores da simulação e definem a frequência de submissão de atividades e de saída e entrada de *peers* na rede.

**Tabela 1: Distribuições adotadas para modelar características heterogêneas dos nós e eventos da simulação.**

Característica	Distribuição	Média	Desvio Padrão	Escala	Forma
Capacidade do Processador	Gamma	80,0	-	30	2
Largura de Banda	Gamma	1,5	-	1	2
Ociosidade da Máquina	Normal	0,5	0,1875	-	-
Submissão de Atividades	Poisson	$x$	-	-	-
Ocorrência de <i>Churns</i>	Poisson	$y$	-	-	-

Além da configuração interna dos nós e dos controles, a topologia da rede também é um fator decisivo no bom desempenho de uma rede P2P. Entretanto, esse fator é muito atrelado à arquitetura que está sendo avaliada pelo simulador. O PeerSim já oferece um arcabouço que facilita a implementação de novas topologias. Nos primeiros estudos com o SciMulator, foi avaliada a arquitetura SciMule (Ogasawara et al. 2010). As medições realizadas com o simulador são apresentadas na seção 4 e foram obtidas utilizando a arquitetura SciMule, que é descrita na seção 3.

## 2.4 O processo de simulação

A simulação foi modelada usando uma abordagem síncrona (Barbosa 1996), baseada em ciclos, do PeerSim. Esta abordagem foi escolhida por garantir maior escalabilidade, embora possua simplificações na camada de transporte. Entretanto, o modelo de transferências de dados desenvolvido no SciMule visa compensar tais simplificações. O processo de simulação transcorre em um número fixo de ciclos. A Figura 2 mostra o diagrama de atividades do simulador. A simulação inicia com a configuração de uma topologia, que dispõe os *peers* na rede. Em seguida, cada ciclo inicia com o processamento dos controles, que são ações globais realizadas na rede, tais como: registros de observação dos eventos (*Observadores*), definição de quais nós saem e ingressam na rede (*Churn*), e quais nós submetem atividades (*Controle de Submissão de Atividades*) naquele ciclo. Logo após, segue a execução dos protocolos por cada um dos *peers*.

Cada *peer* inicia cada ciclo calculando o quanto de seus recursos está disponível para aquele ciclo. Em seguida, verifica se os vizinhos que estão processando alguma de suas tarefas ainda estão ativos. Caso o *peer* detecte a saída de um vizinho, a tarefa é reescalada para outro nó. O próximo passo é a submissão de possíveis novas atividades, com o escalonamento das tarefas. Cada *peer*, então, requisita os dados de entrada para processar as tarefas recebidas e, após, responde às requisições iniciando o envio dos pacotes de dados. Cada *peer* segue com a execução do protocolo de *upload* e, depois, o de *download*. Ao final, cada *peer* processa o que for possível de suas tarefas e envia os resultados e dados de proveniência das tarefas concluídas naquele ciclo.

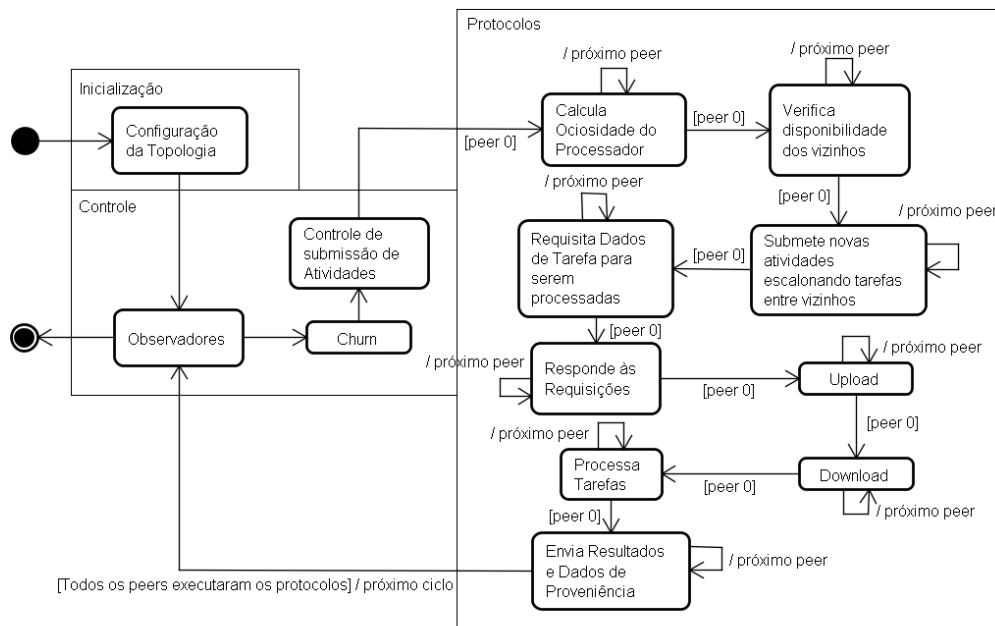


Figura 2: Diagrama de Atividades do Simulador

## 3. A Arquitetura SciMule

O SciMule é uma arquitetura projetada para distribuir atividades de workflows científicos em ambientes P2P. Em linhas gerais, as redes P2P podem ser classificadas quanto ao seu grau de centralização ou a forma de estruturação. Uma rede pode ser: centralizada, na qual um nó (*super-peer*) armazena informações sobre os outros nós e controla as buscas; descentralizada, caracterizada pela ausência do *super-peer* ou outro mecanismo de controle centralizado; e híbrida, que é uma combinação de ambas as

abordagens (Oram 2001). Já no que tange a topologia das redes descentralizadas, as redes podem ser não estruturadas, cujo armazenamento de dados e a vizinhança são aleatórios; ou estruturadas, cuja topologia é definida por algoritmos determinísticos, utilizados para otimizar a localização de recursos (Balakrishnan et al. 2003). Sob esta perspectiva, a arquitetura SciMule é dita híbrida, pois combina certas funcionalidades dos *super-peers* para facilitar o mecanismo de entrada de novos *peers* na rede, porém o escalonamento e a execução de tarefas são feitos de maneira distribuída sobre uma topologia não estruturada.

O objetivo do SciMule é estabelecer uma rede colaborativa entre cientistas para execução de tarefas de experimentos. Os principais desafios enfrentados pelo SciMule envolvem o escalonamento de tarefas, tolerância a falhas, o balanceamento de carga e a elaboração de uma topologia eficiente para troca de dados e processamento de tarefas de experimentos científicos de diferentes domínios, tudo em uma rede com controle distribuído. O SciMule é composto por três camadas: (i) uma camada de submissão que distribui atividades na rede, (ii) uma camada de execução que processa os pacotes de atividades recebidos, e (iii) a camada de *overlay* que armazena informações de posição dos nós na rede. Cada *peer* possui as três camadas e pode se comportar como cliente, quando submete atividades, como nó de execução e como um nó especial denominado *gate peer* (GP), que orienta o ingresso de novos *peers* na rede na camada de *overlay*.

O SciMule provê a análise de dois tipos de paralelização: o paralelismo de dados (Meyer et al. 2007) e o paralelismo por varredura de parâmetros (Samples et al. 2005). Para tal, a camada de submissão é capaz de decompor uma atividade MTC em um conjunto de tarefas e escaloná-las na rede. A Figura 3 apresenta a arquitetura em três camadas que caracteriza o SciMule. Tais camadas são detalhadas a seguir.

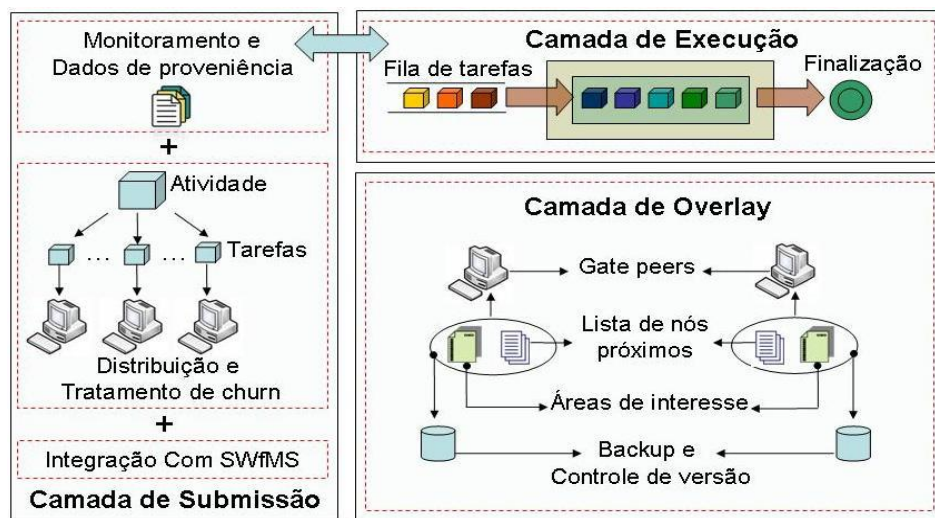


Figura 3: Arquitetura SciMule

**Camada de Submissão.** A camada de submissão é composta por componentes de workflow, que são módulos genéricos incluídos no SWfMS, tal como Kepler (Altintas et al. 2006), Taverna (Oinn et al. 2004) ou VisTrails (Callahan et al. 2006), e por componentes do mecanismo MTC do SciMule, responsáveis por distribuir tarefas, capturar dados de proveniência e tratar eventos de *churn* (entrada e saída intermitente dos nós na rede) (Wu et al. 2008). As atividades são divididas em tarefas de acordo com o tipo de paralelização que é definido pelo pesquisador no arquivo de configurações antes de iniciar o experimento.

**Camada de Execução.** Na camada de execução, as tarefas são recebidas e colocadas em uma fila. Quando os pacotes de dados de uma tarefa são recebidos por completo, elas são processadas. O *peer* confirma a conclusão da tarefa quando a execução tiver sido finalizada, retornando o controle para a camada de submissão. A saída de um *peer* de execução implica o reescalonamento da tarefa a outro *peer* ativo na rede.

**Camada de Topologia.** A camada de topologia (*overlay*) é apoiada por dois componentes: os *gate peers* (GP) e a lista de GP ativos. GP possuem a mesma estrutura que *peers* normais, porém armazenam uma lista de nós próximos (vizinhança) e suas áreas de interesse (*subjects*) (Dick et al. 2009), que estão relacionadas a programas e ferramentas em comum utilizados pelos *peers*. A lista de GP fica disponível na rede para facilitar a busca por GP próximos. Na topologia do SciMule, os *peers* são posicionados em um modelo unidimensional, de modo que a proximidade possa ser medida pela menor diferença entre dois pontos em uma circunferência. Um novo *peer* entra na rede sem qualquer conexão com os demais. O primeiro passo é obter a lista de GP para registrar-se em um deles. O nó recém-chegado se registra no GP mais próximo e, em seguida, adquire a lista de vizinhança. Ele também recebe a lista de um segundo GP adjacente para ter mais opções. Dado que, em uma rede com  $n$  nós e  $g$  GP, onde cada GP possui, em média, uma lista com  $n/g$  *peers*, um nó ingressante tem, inicialmente, no máximo  $2n/g$  opções de vizinhos. Posteriormente, ele também deve estabelecer conexões com outros nós ingressantes registrados no mesmo GP e nos outros dois GP adjacentes, podendo atingir um máximo de  $3n/g$  vizinhos. O SciMule também limita o número de conexões iniciais, de forma que *peers* novos precisam permanecer mais tempo na rede para adquirir mais vizinhos. Esta abordagem visa evitar nós que consomem muitos recursos, porém compartilham pouco (*free riders*).

Para um primeiro estudo do comportamento do SciMulator, a arquitetura SciMule foi utilizada como objeto de simulação. Cada componente descrito nesta seção foi codificado no ambiente SciMulator. A próxima seção traz os resultados dessa avaliação com detalhes. Apesar da avaliação do SciMulator ter sido realizada com a simulação dos componentes da arquitetura SciMule, o simulador não é restrito a esta arquitetura. Na prática, simulações de execuções de workflows científicos em redes P2P podem ser modeladas e executadas no SciMulator, independente da arquitetura proposta/utilizada.

#### 4. Avaliação do SciMulator

As camadas da arquitetura SciMule foram utilizadas para avaliar o comportamento do SciMulator. Para realizar uma avaliação inicial do desempenho do SciMulator, executamos diversas simulações com ele, medindo o tempo de execução, a memória utilizada pelo simulador e quantas tarefas submetidas foram finalizadas variando o número de *peers* e a frequência de *churn*. A variação do tamanho da rede é interessante para avaliarmos a escalabilidade do simulador e a variação do *churn* indica quanto o dinamismo da rede influencia no desempenho do simulador ao processar as atividades. Os dados foram registrados por elementos (classes) de controle específico para registro de *log*, que registraram o tempo decorrido na simulação, a memória máxima utilizada pela máquina virtual Java e a taxa de atividades submetidas e executadas.

As simulações ocorreram por 14.400 ciclos, que equivalem a quatro dias. As atividades, submetidas na rede a uma frequência de 1%, custavam 8000 unidades de processamento com um conjunto de dados de 192MB. Estas atividades foram fragmentadas em 128 tarefas e escalonadas na rede. O número médio de vizinhos foi 32 e a conectividade máxima foi configurada para 64 *peers*. As instâncias do simulador

executaram em uma Altix ICE 8200 com processadores Intel Xeon 5355 de 2.66GHz com cerca de 1GB de memória RAM por núcleo de processamento. Cada instância do simulador foi executada por um núcleo. A Figura 4 apresenta os resultados obtidos em tempo de simulação e memória utilizada.

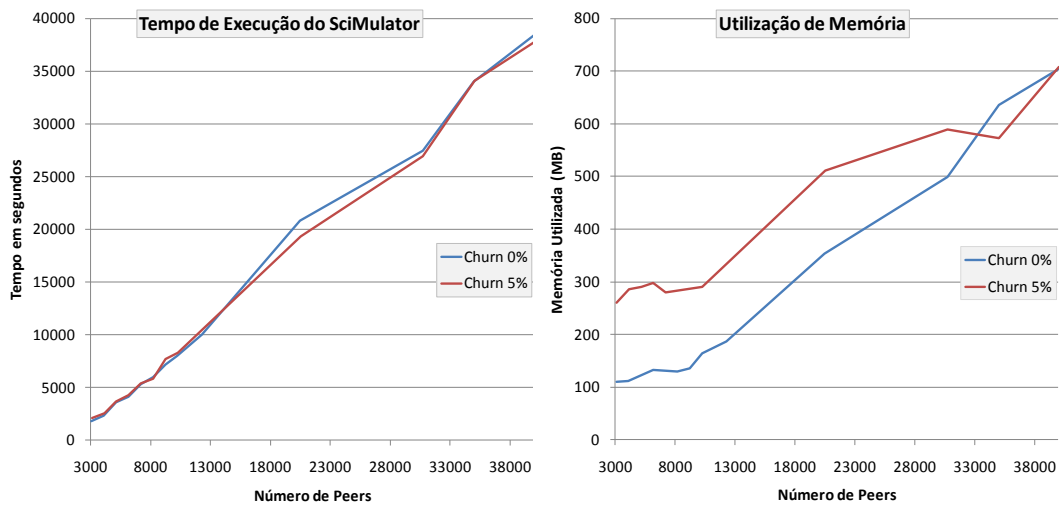


Figura 4: Medidas de desempenho do SciMulator

Ambos os resultados indicam que o simulador apresenta um comportamento linear no tempo de execução e no consumo de memória com o aumento do número de nós da rede. O cenário com *churn* apresenta um consumo maior de memória em função da estrutura de dinamismo do PeerSim, já que os objetos da classe *GeneralNode* não são descartados durante a simulação, mas apenas desligados da rede. Dessa forma, a memória não é liberada, pois é possível que o *peer* desligado volte à rede mais tarde. A aproximação do consumo de memória nos casos com maior número de *peers* na rede se deve à coleta de lixo da máquina virtual Java, que foi configurada para limitar o uso de memória em 768MB. Quando a utilização se aproxima deste valor, o sistema de coleta de lixo torna-se mais rigoroso e frequente.

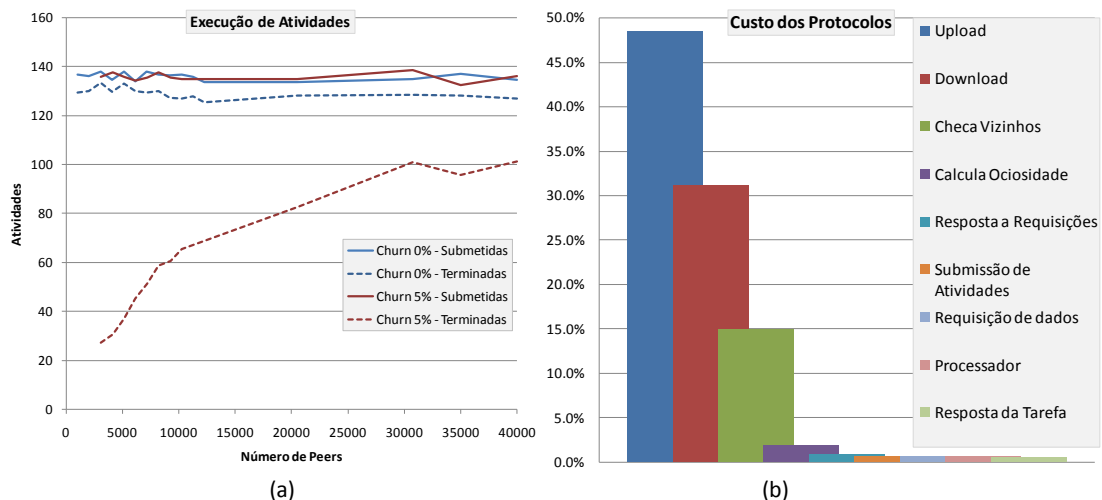


Figura 5: Medidas de (a) Submissão e finalização de atividades e (b) percentual de tempo de simulação gasto com cada protocolo.

Já nas medidas de atividades submetidas e finalizadas, os eventos de *churn* causam um grande impacto, reduzindo a taxa de finalização a cerca de 20% nas simulações com menos *peers*. A Figura 5 (a) mostra o gráfico de finalização de tarefas

variando o número de nós na rede. Nas simulações maiores, a taxa de finalização de atividades cresce, pois, como a frequência de submissão de atividades é a mesma, há mais *peers* ociosos. Dessa forma, a probabilidade de um *peer* ativo no processamento de uma tarefa sair da rede é menor. A queda de um *peer* ativo implica o reescalonamento de uma tarefa, o que pode aumentar significativamente o tempo de execução de uma atividade em função da retransmissão dos dados da tarefa.

A Figura 5 (b) mostra o percentual de tempo gasto com cada protocolo. O protocolo de *upload* é o mais custoso, pois o *peer* que submeteu uma tarefa necessita enviar o conjunto de dados para cada um dos vizinhos que deve processá-la. No protocolo de *download*, o *peer* faz o *download* apenas do dado da tarefa que precisa processar. Ainda assim, fica claro que os protocolos de transmissão de dados são os mais custosos para o simulador representando mais de 75% do tempo de execução de protocolos. Como os processos de transmissão de dados influenciam tanto no tempo de simulação como na taxa de finalização de tarefas, acreditamos que a melhoria de tais processos traga benefícios ao simulador. Técnicas de compactação, replicação e melhor distribuição de dados devem trazer um efeito positivo no desempenho do SciMulator.

## 5. Trabalhos Relacionados

A implantação de ambientes de computação distribuída em larga escala é bastante custosa, comprometendo a viabilidade de estudos para avaliar novos cenários e soluções. Como alternativa, simuladores vêm sendo cada vez mais utilizados. O GridSim (Buyya and Murshed 2002) é um simulador baseado em eventos e orientado para *grids*. Suas principais entidades são: (i) o usuário, que define parâmetros da simulação, tais como frequência de submissão de tarefas e estratégias de escalonamento, (ii) o escalonador, (iii) o recurso compartilhado, (iv) o serviço de informação, que armazena a lista de recursos disponíveis na *grid* e (v) as entradas e saídas que caracterizam as tarefas. Embora simule uma rede heterogênea, o GridSim não oferece uma rede dinâmica com controle totalmente distribuído, dificultando a construção de um simulador P2P.

CloudSim (Buyya et al. 2009) é um arcabouço para a simulação de computação em nuvem que provê componentes que simulam servidores, políticas de escalonamento e alocação de recursos. O CloudSim é independente de plataforma e pode ser caracterizado como uma extensão do GridSim, cuja infraestrutura principal é modelada pelo componente *DataCenter* que recebe as requisições e aloca os recursos, de acordo com a política definida que pode ser do tipo espaço compartilhado ou tempo de processamento compartilhado. Entretanto, o CloudSim também não oferece um ambiente de controle distribuído, pois centraliza o gerenciamento no componente *DataCenter*. Esta abordagem o descaracteriza como simulador para um ambiente P2P.

O PeerSim (Jelasity et al. 2010) é dedicado à simulação de redes P2P e oferece duas abordagens: o modelo síncrono, baseado em ciclos, nos quais cada nó obtém o controle e executa ações (protocolos) de forma sequencial comunicando-se diretamente com outros nós sem uma camada de transporte e, o modelo assíncrono, baseado em eventos, que controla a ordem em que os nós executam os protocolos através da troca de mensagens. O PeerSim oferece uma arquitetura P2P básica satisfatória, mas a avaliação de cenários mais complexos exige a extensão do simulador. A ausência da camada de transporte e dos mecanismos de execução de tarefas nos *peers* tornou necessária a construção do SciMulator.

## 6. Conclusões

A utilização de redes P2P para processamento distribuído de atividades pode ser uma solução interessante, em função da sua maior acessibilidade. Entretanto, a complexidade estrutural para construir uma rede deste tipo para realizar avaliações iniciais de arquiteturas é muito elevada. Por tal motivo, apresentamos o SciMulator, um simulador de ambientes P2P voltado à submissão e execução de atividades de workflows científicos seguindo o paradigma de paralelização MTC.

O SciMulator é uma extensão do PeerSim e modela a paralelização, o escalonamento, a distribuição e a execução de tarefas em uma rede P2P. As características da rede heterogênea e os eventos estocásticos, como a submissão de atividades e eventos de *churn*, variam seguindo distribuições estatísticas com o objetivo de representar a realidade. Inicialmente, modelamos a arquitetura do SciMule no SciMulator para, então, avaliá-lo.

Na avaliação inicial de desempenho, o SciMulator mostrou-se escalável no tempo de execução e consumo de memória, mesmo nos cenários com *churn*. A taxa de finalizações de atividade, como esperado, sofre impacto negativo na presença de *churn*, mas o efeito é amenizado com o aumento da rede. Com estes resultados, acreditamos que o SciMulator é um arcabouço promissor para avaliações de arquiteturas para execução de atividades em redes P2P.

Como trabalhos futuros, pretendemos aprimorar o mecanismo de *churn* do simulador, permitindo que nós que tenham saído da rede anteriormente, possam reingressar e dar continuidade ao processamento de tarefas. Também planejamos implementar soluções para compactação, replicação e distribuição de dados mais eficiente com mecanismos de indexação para melhorar a taxa de finalização de tarefas, uma vez que a retransmissão de dados é o maior gargalo da abordagem.

## Referências

- Abramson, D., Enticott, C., Altintas, I., (2008), "Nimrod/K: towards massively parallel dynamic grid workflows". In: *SC 08*, p. 1-11, Austin, Texas.
- Almeida, E. C. D., Sunyé, G., Traon, Y. L., Valduriez, P., (2008), "A Framework for Testing Peer-to-Peer Systems". In: *Proceedings of the 2008 19th International Symposium on Software Reliability Engineering*, p. 167-176
- Altintas, I., Barney, O., Jaeger-Frank, E., (2006), "Provenance Collection Support in the Kepler Scientific Workflow System", *Provenance and Annotation of Data*, , p. 132, 118.
- Anderson, D., (2004), "BOINC: a system for public-resource computing and storage". In: *Proceedings. Fifth IEEE/ACM International Workshop on Grid Computing*, p. 10, 4
- Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., Stoica, I., (2003), "Looking up data in P2P systems", *Commun. ACM*, v. 46, n. 2, p. 43-48.
- Barbosa, V. C., (1996), *An Introduction to Distributed Algorithms*. The MIT Press.
- Boudani, A., Chen, Y., Straub, G., Simon, G., (2008), "Internet-Scale Simulations of a Peer Selection Algorithm". In: *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, p. 531-535, Los Alamitos, CA, USA.
- Brown, D. A., Brady, P. R., Dietz, A., Cao, J., Johnson, B., McNabb, J., (2007), "A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis", *Workflows for e-Science*, Springer, p. 39-59.
- Buyya, R., Ranjan, R., Calheiros, R., (2009), "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges". In: *Proc.*

- of *HPCS 2009* HPCS 2009, Leipzig, Germany.
- Buyya, R., Murshed, M., (2002). GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *ArXiv Computer Science e-prints*. Disponível em: <http://adsabs.harvard.edu/abs/2002cs.....3019B>. Acesso em: 17 Mar 2010.
- Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., Vo, H. T., (2006), "VisTrails: visualization meets data management". In: *Proc. SIGMOD 2006*, p. 745-747, USA.
- Davidson, S. B., Freire, J., (2008), "Provenance and scientific workflows: challenges and opportunities". In: *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, p. 1345-1350, Vancouver, Canada.
- Deelman, E., Gannon, D., Shields, M., Taylor, I., (2009), "Workflows and e-Science: An overview of workflow system features and capabilities", *Future Generation Computer Systems*, v. 25, n. 5, p. 528-540.
- Dick, M. E., Pacitti, E., Kemme, B., (2009), "Flower-CDN: a hybrid P2P overlay for efficient query processing in CDN". In: *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, p. 427-438, Saint Petersburg, Russia.
- Freedman, D., Pisani, R., Purves, R., (2007), *Statistics, 4th Edition*. 4 ed. W. W. Norton.
- Grid4All Consortium, (2009), "Towards Hybrid Clouds - A Grid4All perspective on cloud computing". , White paper. [www.grid4all.eu](http://www.grid4all.eu).
- Jelasy, M., Montresor, A., Jesi, G. P., Voulgaris, S., (2010), *The PeerSim simulator*, <http://peersim.sourceforge.net>.
- Meyer, L., Scheftner, D., Vöckler, J., Mattoso, M., Wilde, M., Foster, I., (2007), "An Opportunistic Algorithm for Scheduling Workflows on Grids", *High Performance Computing for Computational Science - VECPAR 2006*, , p. 1-12.
- Ogasawara, E., Dias, J., Oliveira, D., Rodrigues, C., Pivotto, C., Antas, R., Braganholo, V., Valduriez, P., Mattoso, M., (2010), "A P2P Approach to Many Tasks Computing for Scientific Workflows". In: *9th International Meeting on High Performance Computing for Computational Science*, Berkeley, CA, USA.
- Ogasawara, E., Oliveira, D., Chirigati, F., Barbosa, C. E., Elias, R., Braganholo, V., Coutinho, A., Mattoso, M., (2009), "Exploring many task computing in scientific workflows". In: *MTAGS 09*, p. 1-10, Portland, Oregon.
- Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M. R., et al., (2004), "Taverna: a tool for the composition and enactment of bioinformatics workflows", *Bioinformatics*, v. 20, p. 3045-3054.
- Oram, A., (2001), *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc.
- Pacitti, E., Valduriez, P., Mattoso, M., (2007), "Grid Data Management: Open Problems and New Issues", *Journal of Grid Computing*, v. 5, n. 3, p. 273-281.
- Raicu, I., Foster, I., Yong Zhao, (2008), "Many-task computing for grids and supercomputers". In: *Workshop on Many-Task Computing on Grids and Supercomputers*, p. 1-11
- Samples, M. E., Daida, J. M., Byom, M., Pizzimenti, M., (2005), "Parameter sweeps for exploring GP parameters". In: *2005 workshops on Genetic and evolutionary computation*, p. 212-219, Washington, D.C.
- Wu, D., Tian, Y., Ng, K., Datta, A., (2008), "Stochastic analysis of the interplay between object maintenance and churn", *Computer Communications*, v. 31, n. 2 (Feb.), p. 220-239.