Grid Simulators

Sérgio Faria
Departamento de Ciências de Computadores
Faculdade de Ciências da Universidade do Porto
Email: up200902611@alunos.dcc.fc.up.pt

***Resumo*—Em sistemas distribuídos, a validação, avaliação e comparação de performance de algoritmos e aplicações é uma dificuldade para os investigadores. A complexidade e propensão a falhas obriga ao uso de métodos experimentais, em detrimento dos analíticos. A experimentação em ambientes reais não é facilmente reproduzível pelos pares, nem desejável, por motivos económicos, em sistemas de larga escala.**

Estes problemas levaram ao aparecimento de várias ferramentas de simulação, que modelam os sistemas permitindo aos investigadores testar seus algoritmos de forma reproduzível. Este paper pretende descrever o estado da arte neste campo, comparando sucintamente as diversas soluções disponíveis.

I. Introdução

Clouds e outros sistemas distribuídos como Grids e sistemas P2P continuam, nos dias de hoje, como tópicos de forte investigação no sentido de os tornar mais eficientes. Algoritmos de escalonamento, gestão de recursos, comunicação na rede e tolerância a falhas são algumas das linhas de investigação nesta área.

A modelação de comportamento e análise de desempenho é uma tarefa importante para a validação de aplicações e algoritmos. Embora métodos analíticos tenham seus méritos, não são normalmente adequados a sistemas paralelos e distribuídos, devido à complexidade necessária para descrever um destes sistemas adequadamente. Neste tipo de sistemas torna-se necessário recorrer a métodos experimentais.

Embora possa-se testar em sistemas de produção, seus resultados são de utilidade limitada dado a imprevisibilidade e falta de controla em sistemas distribuídos de larga escala, susceptíveis a delays e falhas não determinísticas[1]. Uma boa metodologia de teste permite reproduzir os resultados experimentais, não só para validação mas também para comparação com outros algoritmos e implementações. As metodologias experimentais devem ser extensíveis a ambientes computacionais diferentes e seus resultados aplicáveis no mundo real[2].

Quatro classes de metodologias experimentais foram propostas[2] para ambientes distribuídos. Como pode ser observado na tabela I, as metodologias são qualificadas tendo em conta se é o ambiente (nós e middleware) e/ou a aplicação e suas interacções na rede a serem modeladas.

Dado o amplo leque de soluções para as metodologias de teste apresentadas na tabela I, este paper pretende focar-se no estado da arte da simulação, onde quer o ambiente como a componente aplicacional são modeladas. O ambiente em estudo são sistemas distribuídos em larga escala, mais especificamente as Grids.

		Ambiente	
		Real	Modelado
Aplicação	Real	In-situ	Emulação
	Modelada	Benchmarking	Simulação

Tabela I
Classes de metodologias experimentais[2].

Para tal começar-se-á, na secção II, por discutir sobre o papel de ferramentas de simulação e das testbeds para a validação experimental. De seguida serão analisados quatro simuladores para sistemas distribuídos de larga escala: SimGrid (generalista), GridSim (apenas para Grids), CloudSim (apenas para Clouds) e GroudSim (para Clouds e Grids).

II. Simulação como alternativa a Testbeds

Experiências no "lugar"(in-situ) refere-se à utilização de ambientes reais para testar a implementação de algoritmos e aplicações. Como a reprodutibilidade dos resultados é importante, estes ambientes reais não são Grids de produção, mas sim testbeds. Testbeds são sistemas dedicados para a experimentação em ambientes controlados.

Um bom exemplo é a Grid'5000: Uma testbed com 5000 cores, distribuídos por 9 clusters na França, interligados por redes de alta velocidade[3]. Esta testbed destaca-se de outras semelhantes não só pela sua dimensão, mas também por permitir aos investigadores usar a sua própria imagem de sistema (OS) sem virtualização. A preparação para o deployment da experiência é um processo bastante moroso, especialmente se for necessário o teste de diferentes cenários[1]. Além disso construir um sistema destes exclusivamente para testes é um investimento caro, logo as testbeds existentes tendem a ser de acesso partilhado mas controlado através de reservas atempadas (um investigador pode reservar o uso exclusivo de toda a infraestrutura).

A simulação aparece naturalmente como uma forte alternativa a testbeds. Embora testbeds possam permitir resultados mais próximos da realidade, a sua escalabilidade é limitada e seu uso controlado. Simulações são facilmente reproduzíveis permitindo a comparação entre soluções.

III. SimGrid

SimGrid[1] é uma ferramenta genérica, escrita em C, para a simulação de sistemas distribuídos. É capaz de simular recursos time-shared e simular o load das máquinas a partir de

traces reais[4]. Ao contrário das restantes soluções, SimGrid usa modelos analíticos para modelar ligações TCP como um fluxo[1]. No entanto é incapaz de modelar QOS e mecanismos de congestionamento TCP[5][2], revelando-se demasiado optimista em certos casos[1].

A framework é baseada em eventos discretos, permitindo que o tempo de execução cresça linearmente com o número de eventos[2]. A escalabilidade foi avaliada na ordem dos 10000 nós[1][2].

Enquanto que simuladores P2P ignoram o uso de CPU e disco, para suportar milhões de nós[6], ferramentas para Grids tentam balancear os vários factores atenuantes[1]. Os problemas que pretendem tratar definem um simulador: o modo como balanceiam precisão, performance, escalabilidade e complexidade.

Embora genérico o SimGrid está mais virado para a computação distríbuida, com extensões para modelar programas MPI e IaaS clouds[1], sendo incapaz de tratar SaaS clouds[2] e Data Grids.

É de notar que a literatura para esta framework se encontra desactualizada. A ultima versão é a `3.9` (lançada Fevereiro de 2013), no entanto o último paper, de 2008, trata sobre as novidades na versão `3.0`.

IV. GridSim

GridSim é uma solução específica para Grids que permite a modelação e simulação do uso de vários tipos de grids, incluindo o comportamento de schedulers e brokers.

Têm como funcionalidades chave:

- Capaz de simular não só jobs computacionalmente intensivos, como também jobs que usam e transferem grandes quantidades de dados[5][4].
- Permite modelar recursos heterogéneos bem como falhas e o uso de traces reais para simular o uso da Grid[4].
- Suporta Data Grids incluindo catálogos hierárquicos, replicas e políticas de alocação[5].
- A simulação da comunicação em rede é ao nível de pacotes, podendo ter em conta a topologia das entidades e com suporte para delays e QOS[2][5].
- Inclui Grid Information Services, permitindo a criação de VO's e a adição de recursos a meio da simulação[4][7].
- Dispõe de vários pontos de extensão bem definidos permitindo a implementação de diferentes estratégias, estáticas e dinâmicas, para brokers e escalonadores[4][8].

Implementado em Java, GridSim assenta na biblioteca de simulação genérica SimJava, para realizar todo o processo simulação.

Embora SimJava usa um modelo de simulação baseado em eventos discretos. Cada entidade, no SimJava têm o seu próprio thread e além de poder gerar eventos para outras entidades, pode esperar por um determinado evento ou tempo simulado[8][9]. Como o tempo progride de forma discreta e não contínua, o uso de threads em nada contribui para a funcionalidade de espera, apenas permite concorrência no processamento de eventos.

[1]Infrastructure as a Service clouds

[2]Storage as a Service clouds

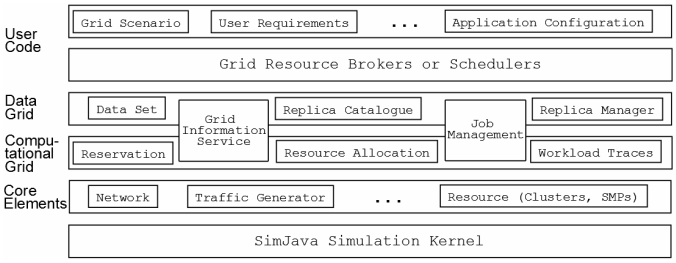

Figura 1. Arquitectura do GridSim[4], que usa o SimJava[9] como base.

Um thread por entidade é uma forte limitação para o GridSim, visto que uma Grid pode ser composta por milhares de recursos e utilizadores. Além de forçar context-switching excessivo, o uso de um thread por entidade provoca um overhead muito significativo em termos de memória[7]. A disparidade entre resultados de benchmarks de escalabilidade[1][2][7], pode ser atribuído a este problema, pois numa máquina com um processador x86, o performance vai-se deteriorar muito mais rapidamente, do que numa máquina x64 usando bastante memória virtual.

É de notar que dado a incompatibilidade no domínio temporal entre os threads e o modelo de simulação, não deveria existir nenhuma limitação que forçasse o uso de um thread por entidade. Após uma breve code review do GridSim e do SimJava, o uso de threads aparenta ser motivado simplesmente pela re-utilização do SimJava. Esta funcionalidade pode ser facilmente removida, procedendo a modificações relativamente pequenas em cada entidade (quebrando API's) e na classe `GridSim`.

V. CloudSim

Já o CloudSim é um simulador para clouds que na sua versão inicial usava o GridSim como base, provando a sua extensibilidade. A Cloud é um paradigma diferente de Grids, onde uma das principais diferenças é o provimento dinâmico de máquinas virtuais, para computação na Cloud.

A virtualização é suportada por serviços de alto nível que permitem alocar máquinas virtuais em máquinas físicas, normalmente num contexto comercial com *Service Level Agreements*. A criação do CloudSim deveu-se à falta de suporte em simuladores de Grid, para modelar os serviços e gestão necessária ao funcionamento das Clouds[10].

CloudSim é útil para Cloud Providers, ao permitir simular algoritmos de escalonamento e de alocação de recursos, mas também é útil para seus utilizadores ao modelar o comportamento aplicacional e custos associados[10]. O uso Clouds federadas ou de diversos Data Centers, com diferentes características, custos e algoritmos de escalonamentos também são suportados pelo CloudSim.

Dado os problemas de performance do GridSim[7], devido ao uso excessivo de threads pelas entidades do SimJava, tentou-se minimizar o número de entidades no CloudSim[11],

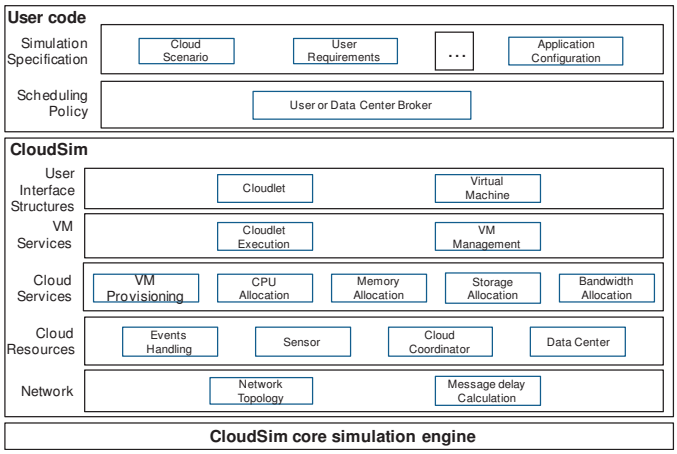

Figura 2. Arquitectura do CloudSim[4], que já não é dependente do GridSim como na 1ª versão.

para que um Datacenter com 5000 máquinas não obrigasse ao uso de 5000 threads.

Na versão `2.0` um novo componente, single-threaded, foi desenvolvido para substituir o SimJava, pois além das limitações já descritas não suportava a criação de novas entidades após iniciar a simulação[12].

O paper[12] que introduz a 2ª versão peca por não identificar claramente, além da remoção do SimJava, o trabalho desenvolvido para a nova versão. Na parte da validação da framework, é mencionado que na 1ª versão o uso de memória era exponencial com o número de hosts. No entanto a sua causa não é especificado nem aparente, visto que é afirmado em trabalho prévio[10][11], que os hosts de um DataCenter não são entidades do SimJava.

Features em destaque neste ultimo paper[12]:

- O uso do formato BRITE para descrever a topologia de rede.
- Detecção da violação de service level agreements.
- O uso de sensores (como violações de SLA e over provision) para ajustar os recursos contratados, num modelo de cloud federada.
- Data Centers com políticas baseadas no consumo energético.

VI. GroudSim

GroudSim[13], o ultimo simulador a ser analisado, também é implementado em Java, pretendendo oferecer uma solução escalável tanto para Grids como Clouds. A framework é um tanto simplista e desenhada para ser facilmente extendida. GroudSim foi integrada no back-end do middleware Askalon (middleware para Grids e Clouds), possibilitando a simulação e execução de aplicações reais no mesmo ambiente integrado[13].

GroudSim têm como pontos chave:

- A simulação, em simultâneo, de Grids e Clouds. A mesma classe é usada para submeter um job numa Grid ou Cloud.
- Tracing do estado do sistema a nível de entidades e eventos[13] (como output da simulação).
- O escalonamento de entidades é time-shared[13] e a visão de um GridSite é homogénea.
- Uma forte aposta no uso de distribuições probabilísticas, várias são incluídas por defeito.
- Suporta reserva de recursos, geração de falhas em Jobs e transferência de dados. Recursos podem ser adicionados no decorrer da simulação.
- Ao contrário dos outros simuladores apresentados, permite descrever cenários em XML[13], em vez de obrigar à escrita de código.
- Permite o uso de traces rais, no formato Grid WorkLoad Archive, para simular o uso de recursos por outros utilizadores[13].

Os autores do GroudSim justificaram a necessidade de um novo simulador[13], exclusivamente nos problemas de performance do GridSim[5] e CloudSim[10], devido ao uso de um thread por entidade. No entanto não descrevem as limitações da sua solução em relação às descritas nos outros papers.

Quer os autores quer os reviewers falharam ao não descobrirem que os problemas de performance descritos, tinham sido resolvidos no CloudSim[12]. Embora o artigo citado tenha sido publicado por volta da mesma altura que GroudSim foi apresentado, a versão `2.0` estava disponível à cerca de um ano (CloudSim é open source). Esta nova versão removeu a dependência directa no GridSim e SimJava, implementando o seu próprio simulador de eventos discretos (single-threaded)[12].

Usando como ponto de referência os outros simuladores estudados, cheguei às seguintes conclusões quanto às limitações da simplicidade do GridSim:

- A visão duma Grid é como um aglomerado de CPU's homogéneos. Não existe a noção de máquina física.
- A simulação da rede é muito básica e está limitada ao controlo de bandwith e geração de falhas.
- Assume que uma Grid e VMs apenas têm uma única porta de rede. O utilizador têm que estender `NetworkLink` se pretender diferenciar tráfego externo do interno, mas mesmo assim não pode diferenciar o custo de transferência de dados.
- Sem suporte para simular workflows (dependências de jobs).
- Todo o processo de brokering têm que ser implementado pelo utilizador. Clouds normalmente incluem a pratica de preços dinâmicos e a escolha duma Cloud não deve ser feita pela sua posição numa lista.
- Apenas apropriado para IaaS clouds e Grids computacionais. Mais virado para utilizadores que providers, dado a falta de schedulers e polítcas de alocação, para além das por defeito (time shared, first come first served).

VII. Conclusão

Neste artigo foram analisados quatro simuladores open-source associados a Grid e Cloud Computing. No entanto a falta de análise experimental e o uso de ferramentas para

simulação especializadas e desenvolvidos *in-house* ainda são práticas comuns, que dificultam a comparação de vários papers académicos[1][2].

Vários simuladores foram desenvolvidos, no contexto de sistemas distríbuidos, para permitir a validação de algoritmos e aplicações, de forma rigorosa. Simuladores apresentam uma alternativa atrativa, de baixo custo e menos morosa do que o uso de testbeds de larga escala.

REFERÊNCIAS

[1] H. Casanova, A. Legrand, and M. Quinson, "SimGrid: a generic framework for large-scale distributed experiments," in *Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on*. IEEE, 2008, pp. 126–131.

[2] J. Gustedt, E. Jeannot, and M. Quinson, "Experimental methodologies for large-scale systems: a survey," *Parallel Processing Letters*, vol. 19, no. 03, pp. 399–418, 2009.

[3] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jégou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab *et al.*, "Grid'5000: a large scale and highly reconfigurable grid experimental testbed," in *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*. IEEE Computer Society, 2005, pp. 99–106.

[4] R. Buyya and A. Sulistio, "Service and utility oriented distributed computing systems: challenges and opportunities for modeling and simulation communities," in *Simulation Symposium, 2008. ANSS 2008. 41st Annual*. IEEE, 2008, pp. 68–81.

[5] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, and R. Buyya, "A toolkit for modelling and simulating data grids: an extension to GridSim," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 13, pp. 1591–1609, 2008.

[6] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," in *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*. IEEE, 2009, pp. 99–100.

[7] W. Depoorter, N. Moor, K. Vanmechelen, and J. Broeckhove, "Scalability of grid simulators: An evaluation," in *Euro-Par 2008 – Parallel Processing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, vol. 5168, pp. 544–553.

[8] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, pp. 1175–1220, 2002.

[9] F. Howell and R. Mcnab, "simjava: A discrete event simulation library for java," in *In International Conference on Web-Based Modeling and Simulation*, 1998, pp. 51–56.

[10] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*. IEEE, 2009, pp. 1–11.

[11] R. N. Calheiros, R. Ranjan, C. A. F. D. Rose, and R. Buyya, "CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services," *CoRR*, vol. abs/0903.2525, 2009.

[12] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[13] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer, "GroudSim: an event-based simulation framework for computational grids and clouds," in *Euro-Par 2010 Parallel Processing Workshops*. Springer, 2011, pp. 305–313.