Simulação de Sistemas *Exascale* usando *Time Warp*[*]

Ricardo Trivizan Fares[1], Aleardo Manacero[1], João Pedro R. Barroso[1], Lucas E. Genova[1], Gustavo T. Juiz[1], Renata S. Lobato[1], Roberta Spolon[2]

[1]Departamento de Ciências de Computação e Estatística
Universidade Estadual Paulista (UNESP) – São José do Rio Preto – SP – Brasil

[2]Departamento de Computação
Universidade Estadual Paulista (UNESP) – Bauru – SP – Brasil

{rt.fares,aleardo.manacero}@unesp.br

Abstract. *The analysis of exascale computing systems, when performed through simulation, requires large computational effort and time making its analysis unfeasible in sequential simulators. Using distributed simulation makes possible the analysis of such complex systems. Nonetheless, performing the simulation in a distributed system gives rise to synchronization problems. Thus, in this work we propose an extension of the simulator iSPD (iconic Simulator of Parallel and Distributed Systems) using the Time Warp optimistic synchronization protocol enabling the simulation of exascale systems in a timely manner.*

Resumo. *A análise de sistemas computacionais exascale, quando feita por sua simulação, exige enorme custo computacional e de tempo que inviabiliza sua análise em simuladores sequenciais. À vista disso, a simulação distribuída torna propício a análise de tais sistemas complexos. Entretanto, a partilha da simulação em um sistema distribuído origina o problema da sincronização. Assim, este trabalho propõe a extensão do simulador iSPD (iconic Simulator of Parallel and Distributed Systems) usando o protocolo de sincronização otimista Time Warp viabilizando a simulação de sistemas exascale em tempo hábil.*

1. Introdução

Os sistemas *exascale*, sistemas capazes de executar 10^{18} operações de ponto flutuante por segundo, tornou-se uma realidade recente com o advento do *Hewlett Packard Enterprise Frontier*. Assim, tal quebra de barreira computacional permitirá que pesquisadores conduzam seus estudos a novas etapas, antes inviáveis no universo *petascale* [Chang et al. 2023].

Além disso, tais sistemas apresentam custo de utilização altíssimo, de forma que identificar gargalos execução das aplicações se torna importante. Essa identificação pode ser realizada por simulação, com custo e precisão satisfatória, em comparação a *benchmarking* e modelos analíticos [Jain 1991].

No entanto, modelos de sistemas *exascale* apresentam milhares de componentes simuláveis, de maneira que os atuais simuladores sequenciais inviabilizam o estudo em tempo hábil. Logo, este trabalho propõe a extensão do simulador iSPD (*iconic Simulator of Parallel and Distributed Systems*) [Manacero et al. 2012] para tornar-se capaz de realizar simulações distribuídas utilizando o protocolo de sincronização otimista *Time Warp*.

[*]processo nº 2022/15807-6, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)

2. Simulação Distribuída

A simulação distribuída refere-se a execução de uma única simulação de eventos discretos em um computador paralelo. Contudo, a execução *concorrente* de eventos ocasiona o problema da sincronização, em que eventos são processados concorrentemente desprezando sua relação de *causa e efeito*, assim, produzindo resultados inesperados [Fujimoto 1990].

Até o momento, na literatura, os mecanismos resolutivos de problemas de sincronização caem em dois grandes grupos: *conservativos* e *otimistas*. O precursor dos protocolos conservativos é o *CMB (Chandy-Misra-Bryant)* [Chandy and Misra 1979]. Por outro lado, o mais famoso protocolo otimista é o *Time Warp* [Jefferson 1985].

Algoritmos de sincronização conservativos evitam, estritamente, qualquer ocorrência de quebra de restrição de *causa e efeito*. Esta técnica é obtida a partir da trava do processamento de um processo lógico até que este tenha certeza de que o próximo evento a ser executado tem marca de tempo menor que quaisquer marcas de tempos de eventos futuros neste mesmo processo lógico [Jafer et al. 2013].

De outra forma, algoritmos de sincronização otimistas permitem que processos lógicos executem eventos sem quaisquer garantias da restrição de causalidade. A corretude da simulação é garantida por meio de um mecanismo de *rollback*, que recupera o estado consistente da simulação toda vez que é detectado alguma violação da causalidade [Jefferson 1985].

De modo geral protocolos otimistas superam em *speedup* os protocolos conservativos, pois estes desperdiçam tempo bloqueados à espera de sincronismo. Entretanto, protocolos otimistas podem sofrer de *rollbacks* em cascata, de tal forma que o processamento será gasto realizando *rollbacks* em vez de progredir a simulação [Jafer et al. 2013]. Neste caso, a situação pode ser resolvida alterando o modelo sendo simulado e/ou a distribuição dos processos lógicos entre os núcleos físicos.

3. Metodologia

Como sabemos que para permitir que o iSPD[1] seja capaz de simular sistemas *exascale* é interessante sua paralelização, nosso trabalho envolve a implementação de protocolos de simulação paralela. Esse processo envolve as seguintes etapas:

1. Revisão bibliográfica sobre simuladores de eventos discretos paralelos;
2. Transposição do modelo de simulação do iSPD antes sequencial para suportar sua paralelização;
3. Otimizações do modelo de simulação paralelo;
4. Execução de testes de desempenho e comparação dos diferentes modos de execução.

A Etapa 1 foi concluída, resultando no entendimento do funcionamento do protocolo otimista *Time Warp* e na obtenção de uma implementação distribuída inicial. Essa implementação foi realizada por meio da utilização do simulador de eventos discretos de propósito geral ROOT-Sim[2] (*ROme OpTimistic Simulator*) [Pellegrini and Quaglia 2014], que é moderno e altamente otimizado.

[1] https://github.com/gspd-unesp/ispd
[2] https://github.com/ROOT-Sim/core

O ROOT-Sim permite três modos de execução, sendo eles: *sequencial*, *paralelo* e *distribuído*. O primeiro realizando a simulação utilizando apenas um núcleo de processamento, o segundo utilizando todos os núcleos de processamento por meio do uso de *pthreads* e, por fim, o terceiro utilizando todos os nós de um *cluster* por meio do uso de MPI (*Message Passing Interface*).

Neste momento estamos executando a Etapa 2, fazendo a transposição do motor de simulação sequencial do iSPD para um motor paralelo. Para isso, estão sendo feitas diversas modificações no tratamento de eventos, cálculo de métricas, escalonadores de tarefas, entre outros. Em suma, tais modificações são necessárias, pois as informações essenciais para o correto funcionamento estão agora dispersos na memória distribuída.

Durante a Etapa 3, otimizações no modelo de simulação serão realizados com o objetivo de: *aumento de desempenho* e *redução do uso de memória*, sendo esses alcançados pela redução de geração de eventos, diminuição do tamanho dos estados dos processos lógicos e utilização de algoritmos eficientes para o cálculo das métricas. Dessa forma, tais objetivos uma vez alcançados permitirão que a carga computacional do modelo *exascale* a ser simulado possa ser maior do que o esperado.

Após isso, durante a Etapa 4, realizaremos testes de desempenho entre os modos de execução dos simuladores, de tal forma que, uma vez obtido tais resultados, poderemos realizar mais adequações e/ou otimizações.

4. Resultados Preliminares

Diversos modelos de sistema *exascale* serão simulados tanto no modelo paralelo em construção quanto no modelo sequencial atualmente no simulador iSPD, tais modelos terão cargas computacionais diferentes, de modo que entenderemos os limiares entre qual modo de execução será mais vantajoso em cada caso, sendo os modos de execução: *sequencial*, *paralelo* e *distribuído*.

A partir disso, tomando-se em conta fatores como: capacidade de processamento, latência de comunicação e distribuição dos recursos, poderemos entender os fatores que produzem melhores desempenhos em um modo de execução do que em outro.

Nos modelos atualmente testados, obtivemos que as configurações do modelo sendo simulado como, por exemplo, a topologia de rede, algoritmos de escalonamento, quantidade de tarefas e a distribuição do tempo entre chegadas das tarefas alteram drasticamente o desempenho do simulador, podendo obter um *speedup* de 17x até nenhum *speedup* satisfatório, executando os modelos em um *cluster* local com 56 núcleos.

5. Observações Finais

Do realizado até agora se reconhece que a paralelização de um modelo de simulação complexo está longe de ser uma tarefa trivial. Entretanto, os objetivos neste trabalho estão sendo alcançados, indicando a possibilidade de extensão do simulador iSPD para a simulação de sistemas *exascale*.

Por fim, os próximos passos são o prosseguimento da Etapa 2, em que citamos a implementação dos tipos restantes de centros de serviço do modelo simulável e na implementação dos escalonadores presentes no simulador iSPD, tomando-se em conta agora a memória distribuída.

Por fim, embora usamos um simulador específico, o modelo sendo transposto será completamente neutro em relação a esse, de maneira que, sua substituição é possível com mínimas modificações, obtendo portabilidade. Para este fim, estamos construindo uma interface que situa-se entre o modelo simulável e o simulador em si, de tal forma que, o modelo não realizará quaisquer chamadas diretas a API (*Application Programming Interface*) do simulador.

Portanto, o uso de outro protocolo de sincronização tratar-se-á apenas pela implementação deste e fornecendo as devidas implementações da especificação da interface. Além disso, o uso de outro simulador de eventos discretos paralelo já existentes como o ROSS (*Rensselaer's Optimistic Simulation System*) [Carothers et al. 2000] ou o WARPED2 [Wilsey 2019] é realizado apenas adequando-os a nossa interface.

Agradecimentos

À Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo fomento do projeto através da bolsa de iniciação científica.

Referências

Carothers, C., Bauer, D., and Pearce, S. (2000). Ross: a high-performance, low memory, modular time warp system. In *Proceedings Fourteenth Workshop on Parallel and Distributed Simulation*, pages 53–60.

Chandy, K. and Misra, J. (1979). Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452.

Chang, C., Deringer, V., Katti, K., Speybroeck, V., and Wolverton, C. (2023). Simulations in the era of exascale computing. *Nature Reviews Materials*, 8(5):309–313.

Fujimoto, R. M. (1990). Parallel discrete event simulation. *Commun. ACM*, 33(10):30–53.

Jafer, S., Liu, Q., and Wainer, G. (2013). Synchronization methods in parallel and distributed discrete-event simulation. *Simulation Modelling Practice and Theory*, 30:54–73.

Jain, R. (1991). *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* Wiley professional computing. Wiley.

Jefferson, D. R. (1985). Virtual time. *ACM Trans. Program. Lang. Syst.*, 7(3):404–425.

Manacero, A., Lobato, R. S., Oliveira, P. H. M. A., Garcia, M. A. B. A., Guerra, A. I., Aoqui, V., Menezes, D., and Da Silva, D. T. (2012). Ispd: An iconic-based modeling simulator for distributed grids. In *Proceedings of the 45th Annual Simulation Symposium*, ANSS '12, San Diego, CA, USA. Society for Computer Simulation International.

Pellegrini, A. and Quaglia, F. (2014). The rome optimistic simulator: A tutorial. In an Mey, D., Alexander, M., Bientinesi, P., Cannataro, M., Clauss, C., Costan, A., Kecskemeti, G., Morin, C., Ricci, L., Sahuquillo, J., Schulz, M., Scarano, V., Scott, S. L., and Weidendorfer, J., editors, *Euro-Par 2013: Parallel Processing Workshops*, pages 501–512. Springer Berlin Heidelberg, Berlin, Heidelberg.

Wilsey, P. A. (2019). Time warp simulation on multi-core platforms. In *2019 Winter Simulation Conference (WSC)*, pages 1454–1468.