Aplicações de Computação Paralela em Jogos Digitais

Paulo Renato Conceição Mendes[1], **Pedro Vinícius Almeida de Freitas**[1], **Francisco Ylderlan Chaves de Oliveira**[1], **Alexandre César Muniz de Oliveira**[2]

[1]TeleMídia – Universidade Federal Maranhão (UFMA)
São Luís, Brasil

[2]DEINF – Universidade Federal Maranhão (UFMA)
São Luís, Brasil

Abstract. *With the increasing complexity and scale of digital games nowadays, new means of optimizing the use of computational resources and solutions that allow the carrying of even more ambitious and complex projects while in commercially viable computers are sought. Therefore, this article seeks to investigate recent applications and advances in the field of parallel computing applied to digital games. In one of the works studied, its proposed a parallel programming* framework *for* engines *of digital games called* Cascade *and their comparison with other methods of parallelizing the procedures of running a digital game.*

Resumo. *Com o aumento da complexidade e escala dos jogos digitais atualmente, são buscados novos meios de otimização do uso de recursos computacionais e soluções que permitam o porte de projetos ainda mais ambiciosos e complexos em computadores comercialmente viáveis. Em vista disso, esse artigo busca investigar as recentes aplicações e avanços no campo da computação paralela aplicada a jogos digitais. Em um dos trabalhos estudados, temos a proposta de um* framework *de programação paralela para* engines *de jogos digitais chamada* Cascade *e a sua comparação com outros métodos de paralelização dos procedimentos de execução de um jogo digital.*

1. Introdução

De acordo com [Coelho 2012], um dos principais desafios da Ciência da Computação atualmente é viabilizar soluções computacionais que reduzam o tempo de processamento e forneçam respostas cada vez mais precisas. Uma vez que a evolução dos processadores possui uma barreira, que é o espaço ocupado pelos componentes, a indústria enxergou a utilização de processadores *multi-core* como uma forma de contornar esse problema. Com o avanço dessa solução, foi percebido que esses processadores poderiam ser utilizados de modo paralelo, com a finalidade de tornar mais rápida a execução de tarefas. A partir desse cenário, surge a computação paralela.

Segundo [Araújo 1999], "A computação paralela é um modo de processamento de informação que foca na manipulação concorrente de elementos de dados, capazes de solucionar um ou mais processos de um único problema". Para [Pacheco 2011], um programa em computação paralela é aquele em que diversas tarefas cooperam de maneira próxima para solucionar um problema. Diversas áreas da Ciência da Computação têm utilizado a computação paralela para melhorar o tempo

de execução de tarefas. Dentre essas áreas pode-se citar as seguintes: processamento de imagens[Saito et al. 2007], otimização[PEDROSO Jr and SCHIOZER 2000], e processos estocásticos[Breitman 2012].

A indústria de jogos digitais movimenta bilhões por ano e vem conquistando um importante espaço na vida de crianças, jovens e adultos e é um dos setores que mais cresce na indústria de mídia e entretenimento[Savi and Ulbricht 2008]. Além do fator entretenimento, jogos digitais vêm sendo objeto de estudo nas mais diversas áreas[Oliveira et al. 2016, Clua 2014, Savi and Ulbricht 2008].

Os jogos digitais estão se tornando cada vez mais realistas, e estão crescendo em complexidade e espaço ocupado. Com o advento da computação ubíqua e dos processadores *multi-core*, tornou-se importante que os jogos digitais fizessem uso desse recurso de maneira eficiente[AlBahnassi et al. 2012]. Ainda segundo [AlBahnassi et al. 2012], em anos recentes, alguns dos jogos existentes passaram por um processo de transição para suportar diversos processadores. Esse processo usualmente vem sendo feito através da conversão de alguns códigos específicos desses jogos para rodar em paralelo.

Este trabalho tem o objetivo de investigar recentes aplicações e avanços no campo de computação paralela aplicada a jogos digitais. Com este fim, este trabalho está dividido da seguinte maneira: a Seção 2 detalha alguns trabalhos que envolvem a computação paralela aplicada ao desenvolvimento de jogos digitais, a Seção 3 faz algumas considerações finais sobre este trabalho.

2. Jogos e Computação Paralela

Um jogo digital consiste de vários módulos que, apesar de estarem relacionados, podem ser tratados separadamente e de modo paralelo. Uma maneira comum de implementação de jogos é o uso de uma função central que gera cada quadro(do inglês *frame*) do jogo [Valente et al. 2005]. Essa função, por sua vez, realiza chamadas a diversos módulos que podem estar presentes no jogo, como por exemplo: animação, física, controle de personagens, testes de colisões, controle de inimigos, etc. A Figura 1 mostra um exemplo de execução de um jogo em que os módulos de animação e física são executados de maneira paralela a cada quadro do jogo e, então, são centralizados novamente para a renderização do quadro.

Os parágrafos seguintes têm o objetivo de apresentar alguns trabalhos que envolvem a aplicação de técnicas de computação paralela para o desenvolvimento de jogos digitais.

O trabalho de [Tagliasacchi et al. 2008] propõe um *framework* de programação paralela para *engines* de jogos digitais, o *Cascade*. No *Cascade*, o processamento de tarefas do sistema são inclusos em uma *Cascade Task*. Tarefas são vinculadas por dependências em um *grafo de dependências de tarefas*. Tal grafo é percorrido em tempo de execução pelo *Cascade Job Manager* que delega tarefas para *threads* para execução. O *Job Manager* deve corretamente satisfazer as dependências enquanto maximiza a performance. O objetivo do *Cascade* é contemplar sistemas complexos, como *engines* de jogos. Tais sistemas são feitos como múltiplos subsistemas interativos, com dependências não triviais. Outros *frameworks* de programação paralela, enquanto são adequados para a paralelização de subsistemas individuais, não se adéquam à estru-

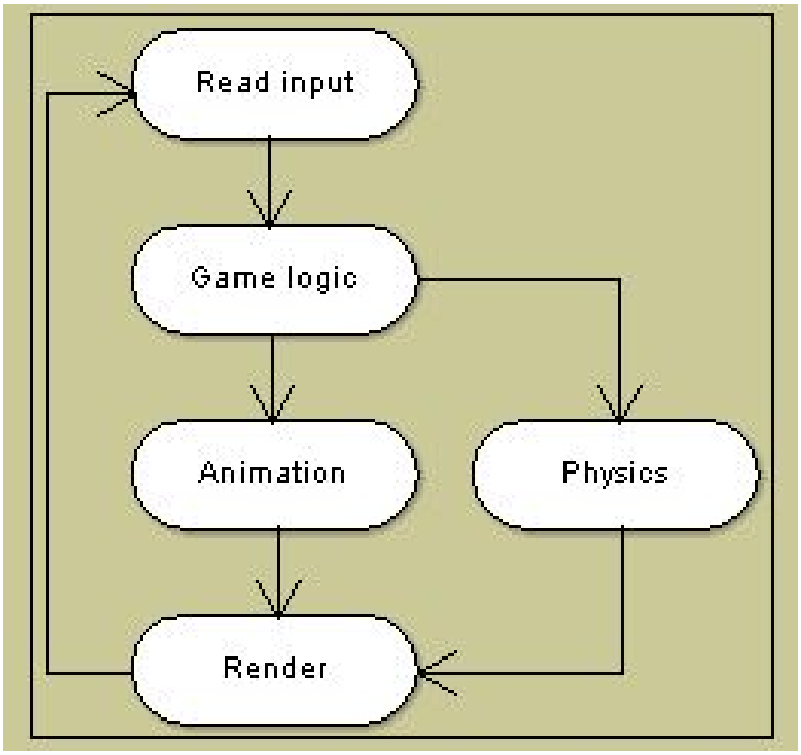

Figura 1. Exemplo de execução de um jogo
Fonte: https://www.geeks3d.com/20100418/game-engine-multi-threading-programming-resources

tura de vários subsistemas funcionando juntos, como explicado no próprio trabalho de [Tagliasacchi et al. 2008].

O trabalho de [AlBahnassi et al. 2012] apresenta um padrão de projeto, o *Sayl*, para a programação paralela de jogos e aplicações de simulação em tempo real para o uso efetivo de múltiplos núcleo de CPUs que, atualmente, estão se tornando comuns em máquinas servidoras. *Sayl* é uma especialização do padrão de projeto *Task Parallelism*, mais genérico. A especialização adiciona três importantes requisitos de programas desse domínio: heterogeneidade de tarefas, dependência do tipo de fluxo de dados entre tarefas, e a mudança dinâmica do conjunto de tarefas ativas em diferentes *frames* do jogo. A principal contribuição dos autores é que o uso do *Sayl* envolve menos trabalho por parte do programador.

Com o objetivo de comparar o *Sayl* com o *Cascade*, [AlBahnassi et al. 2012] propuseram um jogo de teste com o tema de guerra espacial. Tal jogo envolve um jogador que controla uma nave espacial que dispara tiros contra um grande número de partículas. Essas partículas tentam se agrupar, e o jogador deve atirar nelas para evitar que elas consigam se agrupar. O código do jogo foi escrito de maneira comum: *game loop*[1]. Para o paralelismo, algumas mudanças foram feitas no código de simulação de partículas. A Figura 2 mostra a comparação do *speedup*, razão entre o tempo de execução utili-

[1]Em jogos digitais, o *game loop* é o código central do jogo, que é comumente dividido em duas partes: *start*(código executado quando o jogo inicia), e *update*(código executado a cada frame do jogo)

zando n unidades de processamento e o tempo de execução com apenas uma unidade de processamento[Eager et al. 1989], em relação a cada uma das abordagens e o número de partículas no jogo.

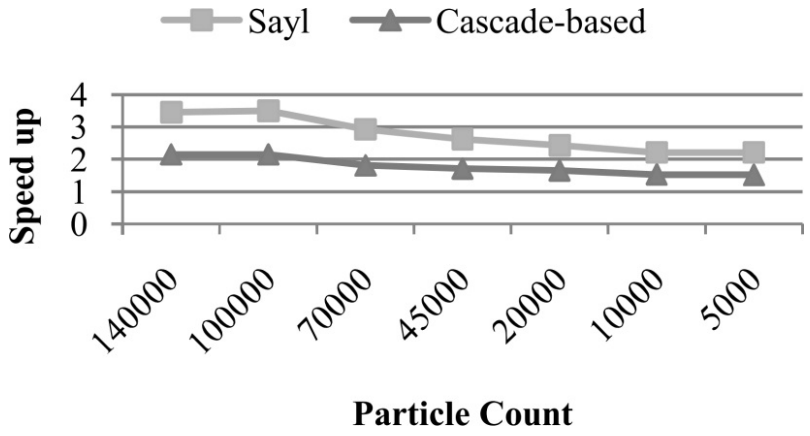

Figura 2. Gráfico que mostra a comparação das abordagens em relação ao *speedup*
Fonte: Imagem retirada de [AlBahnassi et al. 2012]

A Figura 3 mostra o tempo necessário para executar um quadro do jogo em milissegundos, onde as abordagens que utilizam paralelismo são comparadas com a abordagem sem paralelismo.

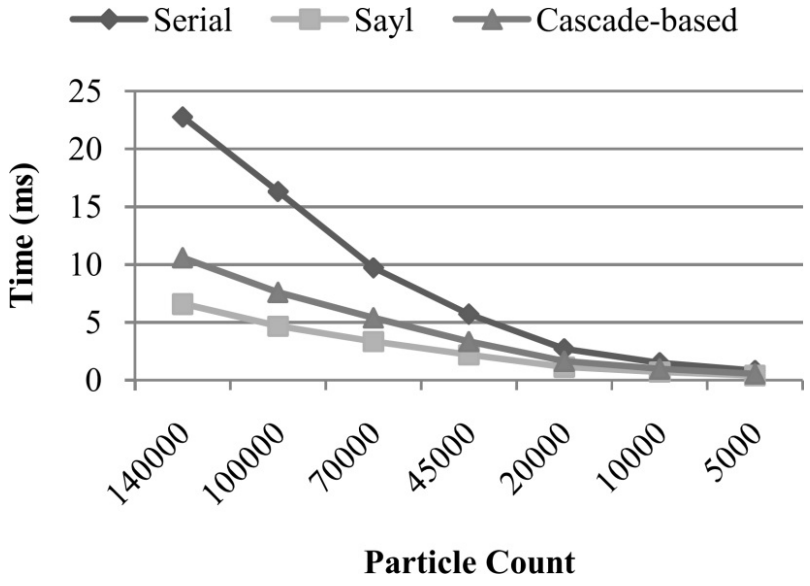

Figura 3. Gráfico que mostra a comparação das abordagens em relação ao tempo de execução

Fonte: Imagem retirada de [AlBahnassi et al. 2012]

Segundo [Andblom and Sjöberg 2018], que fazem uma comparação entre padrões de projeto para o desenvolvimento de jogos digitais, os desenvolvedores de jogos devem utilizar o paralelismo disponível atualmente para reduzir ciclos e o tempo de renderização de jogos. Um dos avanços no que diz respeito a esse potencial é a criação de *multithreaded engines* de games que aproveitam as unidades de processamento adicionais. Nessas

engines, alguns ramos do *game loop* são paralelizados. Entretanto, as especificidades dos padrões de projeto para desenvolvimento paralelo usados e ideias de como combiná-los não são descritos. O trabalho de [Andblom and Sjöberg 2018] especifica esses fatores, para fornecer um guia de quando usar cada um dos seguintes padrões de projeto para desenvolvimento paralelo: *fork-join* e *pipeline parallelism*. Usando um coleção de dados e comparando em relação às métricas de *speedup* e eficiência, o trabalho deles conclui que a paralelização do *game loop* de jogos pode ser melhor organizada com o uso de diferentes padrões de projeto para desenvolvimento paralelo.

3. Conclusão

Como apresentado na Seção 2, pode-se observar que a combinação de múltiplos padrões de projeto para desenvolvimento de jogos digitais pode propiciar o aumento da eficiência do uso dos recursos de *hardware*. Tendo em vista os avanços na complexidade e escala dos jogos atuais, a paralelização dos elementos do *gamelooop* oferece uma possibilidade de continuar esse aumento com maior distância entre a inviabilidade de projetos de grande escala e os recursos necessários para executa-los. Esse artigo apresentou algumas das recentes técnicas no campo da computação paralela aplicada a jogos digitais. Porém é válido citar que métodos para otimizar o uso de recursos nas mais diversas áreas computacionais estão em voga e, portanto, é valida a busca continuada de métodos mais atuais e eficientes.

Referências

AlBahnassi, W., Mudur, S. P., and Goswami, D. (2012). A design pattern for parallel programming of games. In *High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*, pages 1007–1014. IEEE.

Andblom, R. and Sjöberg, C. (2018). A comparison of parallel design patterns for game development.

Araújo, A. P. F. d. (1999). *DPWP-Uma nova Abordagem para o Escalonamento Dinâmico em Computação Paralela Virtual*. PhD thesis, Universidade de São Paulo.

Breitman, K. K. (2012). *Arcabouço para desenvolvimento de serviços baseados na Simulação de Monte Carlo na Cloud*. PhD thesis, PUC-Rio.

Clua, E. W. G. (2014). Jogos sérios aplicados a saúde. *Journal of Health Informatics*, 6.

Coelho, S. A. (2012). Introdução a computação paralela com o open mpi. *Simpósio Brasileiro de Computação Paralela com o Open MPI*, pages 24–44.

Eager, D. L., Zahorjan, J., and Lazowska, E. D. (1989). Speedup versus efficiency in parallel systems. *IEEE Transactions on Computers*, 38(3):408–423.

Oliveira, R. G. S. G. d., Mendes, P. R. C., and Soares-Neto, C. S. (2016). Framework dirigido por modelos para monetizacão de jogos free-to-play. *Proceedings of SBGames*.

Pacheco, P. (2011). *An introduction to parallel programming*. Elsevier.

PEDROSO Jr, C. and SCHIOZER, D. (2000). Otimização de locações de poços usando simulação numérica de reservatórios e computação paralela (pvm). In *Rio Oil and Gas Expo and Conference. Rio de Janeiro, Brasil*.

Saito, P. T., Sabatine, R. J., Nunes, F. L., and BRANCO, K. (2007). Uso da computação paralela distribuída para melhoria no tempo de processamento de imagens médicas. *In Anais da XIV Escola Regional de Informática do Paraná-ERI-PR*, pages 36–47.

Savi, R. and Ulbricht, V. R. (2008). Jogos digitais educacionais: benefícios e desafios. *RENOTE*, 6(1).

Tagliasacchi, A., Dickie, R., Couture-Beil, A., Best, M. J., Fedorova, A., and Brownsword, A. (2008). Cascade: A parallel programming framework for video game engines. In *Proceedings of the Workshop on Parallel Execution of Sequential Programs on Multi-core Architectures. Institute of Computing Technology, Chinese Academy of Sciences*, pages 47–54.

Valente, L., Conci, A., and Feijó, B. (2005). Real time game loop models for single-player computer games. In *Proceedings of the IV Brazilian Symposium on Computer Games and Digital Entertainment*, volume 89, page 99.