Ambiente para Estudo de Computação Paralela Baseado no Simulador Completo GEM5 e em Algoritmos de Ordenação Escritos com OpenMP

Leonardo B. A. Vasconcelos, Max V. Machado, Henrique C. Freitas
Grupo de Arquitetura de Computadores e Processamento Paralelo (CArT)
Programa de Pós-graduação em Informática (PPGINF)
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Belo Horizonte, Brasil
leonardo.athayde@sga.pucminas.br, maxm@pucminas.br, cota@pucminas.br

***Resumo*—Com o surgimento e disseminação de arquiteturas paralelas para computadores pessoais (e.g. processadores multi-core), a discussão sobre o ensino de programação paralela ganhou força e importância para uma melhor formação dos egressos em cursos de computação. Nesse contexto, em disciplinas de Arquitetura de Computadores, há também a necessidade de correlacionar programação e arquiteturas paralelas. Dessa forma, é possível explicar com mais clareza os impactos decorrentes de decisões de programação e escolhas arquiteturais no desempenho final de um sistema computacional. O principal problema é encontrar o momento adequado para tratar do assunto programação paralela e qual ambiente usar com flexibilidade para estudos e configurações de futuras arquiteturas. Neste artigo, apresenta-se um ambiente composto por um simulador de sistema completo e um *benchmark* de algoritmos de ordenação paralelos para práticas que envolvam avaliação da programação e projeto de arquiteturas paralelas. Os resultados apresentam relatos positivos de alunos que se envolveram com o *benchmark* em uma disciplina de algoritmos e estruturas de dados, e da aplicação do *benchmark* no simulador para uso em disciplinas de arquitetura de computadores.**

***Palavras-chave*-arquitetura e programação paralela; simulação; algoritmos de ordenação; OpenMP.**

I. Introdução

A computação paralela tem sido cada vez mais utilizada com o objetivo de atingir um melhor desempenho na execução de aplicações. Para que uma aplicação utilize uma arquitetura paralela de forma eficiente é necessário que o programador conheça bem o potencial paralelo do código. Com isso, há a necessidade de profissionais aptos a desenvolver aplicações capazes de tirar o proveito máximo de processadores com múltiplos núcleos. Esses profissionais serão cada vez mais demandados pelo mercado, pois a tendência é que as arquiteturas se tornem cada vez mais paralelas, um exemplo para essa tendência é o aumento do número de núcleos dentro do processador.

No entanto, a realidade da maioria das universidades com cursos de computação é o ensino da programação numa abordagem sequencial e não paralela. Tais cursos tem como tradição disciplinas focadas no desenvolvimento de aplicações sequenciais utilizando de forma ineficiente o *hardware* disponível. Normalmente, a computação paralela é tratada entre os educadores da área como um tema avançado, pois se torna complexa à medida que envolve várias áreas da computação como arquitetura de computadores, compiladores e algoritmos. Por esse motivo, é comum ter uma grade curricular na qual, ao final do curso, existe uma disciplina específica para abordar apenas assuntos relacionados à Computação Paralela e Distribuída (CPD). No entanto, existe um contrassenso em relação a essa estrutura curricular, pois há uma forte aceitação dos *stakeholders* da área do ensino da computação paralela que esse tipo de assunto seja abordado o mais breve possível nos cursos de graduação [1], utilizando uma abordagem de ensino espiral e sendo visto em diferentes disciplinas sobre diferentes aspectos. As disciplinas para inserção de tópicos de CPD não necessariamente precisam ser somente aquelas voltadas para programação, mas por exemplo, Arquitetura de Computadores, foco deste artigo.

Este artigo tem como o principal objetivo apresentar uma proposta de um ambiente para o ensino da computação paralela com a utilização de um simulador completo para disciplinas de Arquiteturas de Computadores e também de programação paralela. Dessa forma, com os resultados das simulações, o aluno pode visualizar os benefícios que se têm quando a programação paralela é aplicada de forma correta sobre uma determinada arquitetura paralela. Além disso, o *Technical Committee on Parallel Processing* (TCPP) possui um documento listando vários tópicos relacionados à computação paralela, que podem ser cobertos por cursos de computação. Um desses tópicos trata justamente da importância de se ter uma variedade de *benchmarks* como uma forma alternativa para a análise do impacto de uma determinada arquitetura paralela [2]. Este artigo visa cobrir um desses tópicos da lista, com um *benchmark* baseado em algoritmos de ordenação paralelos descrito na Seção III. As simulações apresentadas neste artigo possuem o intuito de ilustrar, de forma simples e até mesmo óbvia, o casamento entre programação paralela e arquitetura paralela através de um ambiente simples e viável, capaz de oferecer potencial para futuros experimentos de ensino/aprendizagem.

O artigo está estruturado na seguinte forma. A Seção II apresenta outros trabalhos que descrevem a atual situação do ensino da computação paralela nas universidades. Na Seção III, é descrito o ambiente de aprendizagem proposto neste artigo e também os algoritmos que compõem o *benchmark*. A Seção IV descreve os resultados das simulações feitas com o *benchmark*, com relatos de alunos que cursaram a disciplina Algoritmos e Estrutura de Dados

II (AEDII) no primeiro semestre de 2014. As conclusões e trabalhos futuros estão na Secção V.

II. Trabalhos Relacionados

Os trabalhos relacionados que são mencionados neste artigo são todos voltados para a educação em computação paralela.

A Universidade UTN-Bahia Blanca [3] inclui uma nova disciplina denominada *parallel processing* dentro do curso de Engenharia da Computação. É uma disciplina em que a grande maioria dos alunos estão no seu último ano e basicamente a estrutura da disciplina inclui MPI, Pthreads, OpenMP e OpenCL. A Universidade de Murcia na Espanha [4] também realiza um trabalho para a inclusão do paralelismo no curso de Engenharia de Computação, só que o foco é para alunos do segundo ano, no qual vários conceitos de CPD são distribuídos entre quatro disciplinas do curso. Além disso, o projeto conta com a colaboração do *Supercomputing Centre of the Scientific Park Foundation of Murcia (SCC)*, no qual os alunos têm acesso a um *cluster* para a atividade prática. A disciplina COMP8320 *Multicore Computing: Principles and Practice* é ofertada na *Australian National University* [5] que é uma universidade que já oferta outras disciplinas relacionadas ao CPD, como a COMP2310 *Concurrent and Distributed Systems*. A diferença entre uma e outra é o nível de profundidade no qual ambas ensinam OpenMP, mas a primeira numa profundidade maior. A disciplina cobre três paradigmas de programação: Memória Compartilhada, Troca de Mensagens e Unidade Gráfica utilizando CUDA. Richard Brown e Elizabeth Shoop [6] iniciaram um projeto chamado CSInParallel como uma forma de incentivar o paralelismo na educação do curso de Ciência da Computação. O projeto possui um site com o endereço *csinparallel.org*, que reúne materiais didáticos para o ensino de CPD organizado em módulos. A ideia do projeto é formar uma comunidade para que os *stakeholders* da área criem módulos e compartilhem esses módulos.

Os trabalhos relacionados retratam sobre o ensino de CPD, no qual cada projeto expõe iniciativas para injetar mais paralelismo nos cursos. De forma similar ao projeto *CSInParallel* que reúne uma série de materiais para a aprendizagem da computação paralela, este artigo apresenta um ambiente de estudo com a utilização de um simulador onde o *benchmark* proposto é composto especificamente por algoritmos de ordenação paralelos.

III. Proposta de Ambiente de Aprendizagem

Para que os estudantes de Arquitetura de Computadores se familiarizem com a computação paralela, é proposta a utilização de um simulador de sistema completo e um *benchmark* composto por algoritmos de ordenação paralelizados com OpenMP. O GEM5 [7], que é o simulador proposto pelo artigo, é um simulador de plataforma modular que suporta diferentes ISAs (*Instruction Set Architecture*) como Alpha, ARM, SPARC, MIPS, POWER e x86. Outro fator para a escolha desse simulador é que é possível executar as diretivas do OpenMP que são necessárias para a paralelização dos códigos que compõem o *benchmark*.

Existem duas formas de se executar um programa nesse simulador que são:

- *Syscall Emulation Mode* (SE): É a forma mais simples para a execução de uma simulação no GEM5. Nesse modo, é necessária apenas a aplicação, sendo desnecessária a emulação de um sistema operacional durante a simulação.
- *Full System Mode* (FS): Nesse modo, é necessário que o sistema operacional seja emulado, o que vai demandar de mais conhecimento sobre o GEM5, pois é necessário entender algumas peculiaridades do funcionamento do modo FS. Felizmente, o simulador GEM5 possui uma boa documentação em sua *web page* oficial para a utilização do modo FS. A vantagem desse modo é que possui um maior suporte para algumas funcionalidades que não há no modo SE. A simulação no *Full System* exige uma imagem do sistema operacional e do *kernel*, ambos vão compor um ambiente em *Linux* a ser emulado.

A arquitetura utilizada para a execução do *benchmark* é a x86, sendo mais prática a compilação das aplicações sem a necessidade de fazer a compilação cruzada. Outro ponto é que o modo SE do GEM5 não suporta as diretivas do OpenMP, que são necessárias para a paralelização dos algoritmos na execução do *benchmark*. Então, para esse ambiente de aprendizagem proposto, é utilizado apenas o modo FS.

Para executar o *benchmark* no modo FS, é necessário que se tenha no comando de execução um apontamento para a imagem do Linux e que nessa imagem tenha incluído nela o *benchmark*. Para isso, é necessário que se monte a imagem e inclua os executáveis e depois desmonte a imagem. Para adicionar um novo algoritmo ao *benchmark*, é necessário refazer esse procedimento. Além de apontar a imagem para o Linux, é necessário também indicar o *kernel* e também o arquivo *rcS* que é um arquivo de *script* que contém os comandos que vão ser executados dentro do ambiente Linux a ser emulado. Todos os arquivos necessários para a execução do *benchmark*, com a exceção do próprio simulador GEM5, estão disponíveis em [8].

A seguir, há um exemplo do comando a ser utilizado para a execução de um dos algoritmos do *benchmark*. O parâmetro *–disk-image* aponta para a imagem Linux a ser utilizada e o parâmetro *–kernel* aponta para o *kernel* a ser utilizado. Já os *scripts* são apontados pelo parâmetro *–script* e, finalizando, um último parâmetro *-n* indicando o número de núcleos que a arquitetura simulada terá. Esses são os parâmetros mínimos indicados para a simulação no ambiente de estudo.

/.../gem5.prof /.../fs.py –disk-image=/.../linux-x86.img –kernel=/.../x86_64-vmlinux-2.6.22.9.smp –script=/.../SelectionSort.rcS -n 2

A. Configurações da Arquitetura Simulada

Para que se faça algumas configurações no simulador GEM5 é necessário reescrever algumas linha de códigos Python. Sendo assim, para o ambiente a ser simulado

sugere-se a configuração padrão do GEM5 da arquitetura x86. No entanto, uma configuração importante e fácil para a execução dos algoritmos é a definição do número de núcleos. No caso, o estudante pode modificar facilmente o número de núcleos na linha de comando demonstrado anteriormente. A Tabela I demonstra a configuração padrão utilizada na arquitetura x86 do GEM5.

Tabela I
CONFIGURAÇÃO PADRÃO DA ARQUITETURA X86 GEM5

Componente	Parâmetro	Configuração
Processador	Quantidade de núcleos	1 - 4
	Frequência	2GHz
	Execução	Em Ordem
Cache L1	Tamanho da Linha	64 kB
	Tamanho - Dados(D)	64 kB
	Tamanho - Instruções (I)	32 kB
	Associatividade (D)	2-*way*
	Associatividade (I)	2-*way*
	Latência	2 ciclos
Cache L2	Tamanho da Linha	64 MB
	Tamanho	2 MB
	Associatividade	8-*way*
	Latência	20 ciclos

B. Benchmark

O *benchmark* é composto por seis programas paralelos de ordenação que são: *bublle sort*, *selection sort*, *odd-even sort*, *shell sort*, *merge sort* e *quick sort*. Para que a execução do *bublle sort* de forma paralela fosse possível, foi aplicada a seguinte lógica no programa. Primeiro se divide a lista entre várias *threads*. Depois, cada *thread* paraleliza um local da lista e em seguida, com a lista tendo suas partes ordenadas, reorganiza a lista para ficar toda ordenada. A Figura 1 ilustra a lógica descrita.

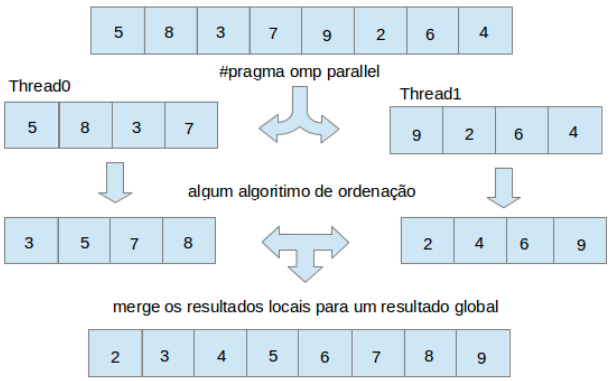

Figura 1. Lógica genérica para a paralelização de algoritmos de ordenação

Para escolher qual dos algoritmos será usado na simulação, basta indicar o arquivo de *script* correto, pois para cada algoritmo há um arquivo *rcS* correspondente. Nesse arquivo, existe um número inteiro que é um parâmetro indicando o tamanho da lista a ser ordenada pelo algoritmo. Caso haja a necessidade de aumentar ou diminuir o tamanho da entrada do algoritmo, é necessário modificar esse número. A Tabela II mostra o tamanho das entradas que cada algoritmo contém por padrão. Caso o aluno faça uma simulação utilizando o algoritmo *Selection Sort* e não tenha modificado o arquivo *rcS*, será realizada uma ordenação sobre uma lista com 10000 números inteiros.

Tabela II
CONFIGURAÇÕES PADRÃO DAS ENTRADAS DO *benchmark*

Algoritmo	Entrada	Algoritmo	Entrada
Bublle Sort	10000	Shell Sort	1000000
Selection Sort	10000	Merge Sort	1000000
Odd-Even Sort	10000	Quick Sort	1000000

IV. RESULTADOS

Na disciplina de Algoritmos e Estrutura de Dados II (AEDII), foram utilizados os algoritmos de ordenação paralelos que compõem o *benchmark* proposto neste artigo para o ambiente de estudo sem o uso do simulador. Esses algoritmos tradicionalmente são apresentados na versão sequencial. Com essa atividade, os alunos tiveram a oportunidade de visualizar a versão paralela, tendo ao mesmo tempo uma introdução à programação paralela. Os alunos que participaram dessa atividade, em sua maioria, são calouros, pois a disciplina é lecionada no primeiro ano do curso. A seguir, estão três relatos feitos por três estudantes que participaram da atividade.

Aluno 1: "*A Computação Paralela hoje não deve ser mais considerada uma utopia na computação, mas uma realidade que temos que nos adaptar. A introdução foi fundamental para o conhecimento da nova tecnologia, mas mais do que isso acho que mais atividades e investimentos em computação paralela é de grande importância para o nosso futuro profissional*"

Aluno 2: "*Programação paralela é fundamental para uma melhor utilização do processador, agilizando o tratamento de dados.*"

Aluno 3: "*A atividade foi interessante, estamos começando a nos preocupar com o tempo de execução e o paralelismo entra ai.*"

Com estas mensagens, percebe-se que os alunos tomaram consciência da computação paralela assim como seu papel e sua importância para os dias atuais. A evidência disso é que no semestre seguinte os alunos têm tomado a iniciativa de dialogar com o professor de AEDIII sobre possíveis paralelizações de outros algoritmos que aprendem na disciplina e até mesmo apresentam soluções paralelas prontas.

A. Execução do Benchmark

Para a simulação, foi utilizada a configuração padrão, apenas com a modificação no número de núcleos. Tendo em vista que o *speedup* é uma das principais métricas utilizadas na computação paralela, esta pode ser usada para demonstrar o potencial que o ambiente possui para a aprendizagem. As Figuras 2 e 3 contém os gráficos com os tempos de execução das simulações. Cada algoritmo possui duas barras verticais, sendo que a barra da esquerda representa o tempo de execução com um único núcleo

e a barra da direita o tempo de execução com quatro núcleos. Os resultados foram divididos em dois gráficos, pois os algoritmos inclusos na Figura 2 possuem diferentes tamanhos de entrada em relação aos algoritmos da Figura 3. Enquanto os algoritmos do primeiro gráfico receberam uma lista de tamanho *10000*, os algoritmos do segundo gráfico receberam uma lista de tamanho *1000000*, isso de acordo com a configuração padrão de entrada de dados mencionado na Seção III. Além desses resultados, é apresentado na Tabela III o *speedup* alcançado por cada algoritmo.

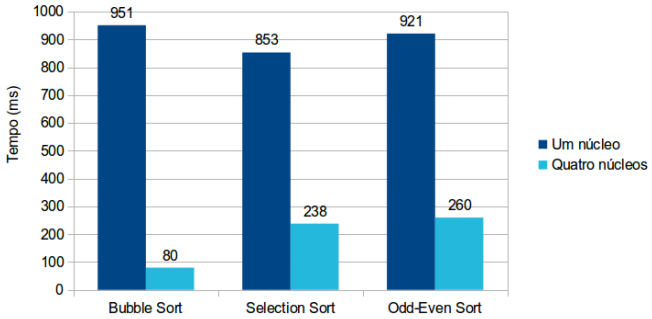

Figura 2. Tempo de execução do *Bubble Sort*, *Selection Sort* e *Odd Even Sort*, utilizando um núcleos e quatro núcleos

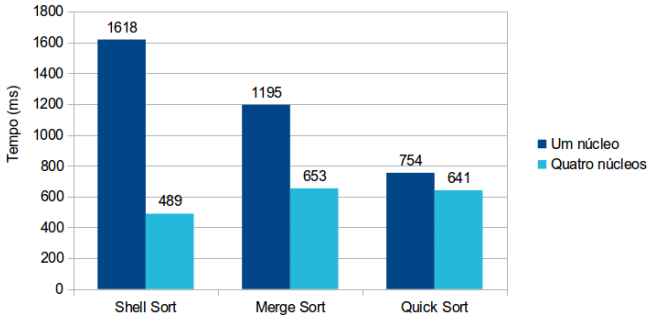

Figura 3. Tempo de execução do *Shell Sort*, *Merge Sort* e *Quick Sort*, utilizando um núcleos e quatro núcleos

Tabela III
SPEEDUP DOS ALGORITMOS AO UTILIZAR QUATRO NÚCLEOS.

Algoritmo	Speedup	Algoritmo	Speedup
Bublle Sort	11,8	Shell Sort	3,3
Selection Sort	3,58	Merge Sort	1,83
Odd-Even Sort	3,54	Quick Sort	1,17

Com os resultados dos gráficos e da tabela, fica visível a vantagem de desempenho que se tem na configuração de quatro núcleos em relação à configuração de um único núcleo. Embora esses resultados sejam óbvios, é importante que o aluno que está começando um contato inicial com a computação paralela em Arquitetura de Computadores tenha a sua hipótese lógica comprovada. Em outras palavras, esse é um ambiente de estudo em que o aluno do 2ºou 4ºperíodo pode ter um contato básico com um simulador de sistema completo e ao mesmo tempo com conceitos de programação paralela de forma integrada.

V. CONCLUSÕES E TRABALHOS FUTUROS

Com a importância que a computação paralela vem ganhando, é cada vez mais importante a aplicação dela em cursos como Ciência da Computação. Além disso, é importante que durante o curso, a computação paralela seja vista continuamente como um modelo espiral, pois assim o aluno poderá obter uma formação sólida para um futuro que tende a ter cada vez mais núcleos dentro de um *chip*. Esse ambiente de estudo visa propor a continuidade do estudo da computação paralela no curso, para que o aluno se sinta cada vez mais familiarizado com os termos da área, assim como a sua aplicação e sua importância. Fica como trabalho futuro a aplicação da proposta abordada nesse artigo em diferentes turmas de Arquitetura de Computadores para exploração do *benchmark* em projetos de arquiteturas *multi-core*.

REFERÊNCIAS

[1] R. Brown and et al, "Strategies for preparing computer science students for the multicore world," in *Proceedings of the 2010 ITiCSE Working Group Reports*, ser. ITiCSE-WGR '10. New York, NY, USA: ACM, 2010, pp. 97–115.

[2] S. K. Prasad and et al, "Nsf/ieee-tcpp curriculum initiative on parallel and distributed computing: Core topics for undergraduates," in *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '11. New York, NY, USA: ACM, 2011, pp. 617–618.

[3] J. Iparraguirre, G. Friedrich, and R. Coppo, "Lessons learned after the introduction of parallel and distributed computing concepts into ece undergraduate curricula at utn-bahìa blanca argentina," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, May 2012, pp. 1317–1320.

[4] M. Acacio and et al, "An experience of early initiation to parallelism in the computing engineering degree at the university of murcia, spain," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, May 2012, pp. 1289–1294.

[5] P. Strazdins, "Experiences in teaching a specialty multicore computing course," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, May 2012, pp. 1283–1288.

[6] R. Brown and E. Shoop, "Csinparallel and synergy for rapid incremental addition of pdc into cs curricula," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, May 2012, pp. 1329–1334.

[7] N. Binkert and et al, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[8] (2014) Leonardoathayde/ordenacaoparalelizada. [Online]. Available: https://github.com/LeonardoAthayde/Ordenacao Paralelizada