Processamento Paralelo com Hyper-Threading: Uma Análise de Desempenho Como Instrumento de Ensino e Aprendizagem em Sistemas Operacionais

Diego J. Hoss[1], Gil Eduardo de Andrade[1], Roberta Suero[1], Adriano C. V. C. Yasuda[2], Jonatas S. do Nascimento[2], Josepher J. C. da Silva[2], Levy J. F. Da Silva[2]

[1]Docente – Eixo de Informação e Comunicação
Instituto Federal do Paraná (IFPR) – Campus Paranaguá, PR – Brazil

[2]Discente – Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal do Paraná (IFPR) – Campus Paranaguá, PR – Brazil

{gil.eduardo, roberta.suero, diego.hoss}@ifpr.edu.br, {adriano__kira, jonatas-silveira, josephercastrosilva, levy_jorge}@hotmail.com

Abstract. *A computer program's performance is affected by the operational system (OS). It's possible to improve the performance by making use of the parallelism approach. To use this technique, the developer must know the operational system's functionalities and how to utilize them. This article proposes the development of a program that uses parallelism with Hyper-Threading technology and its equivalent without parallelism. Therefore, a performance analysis is inferred by tracing a relationship between program performance and knowledge regarding operational systems. The results showed that the performance of an application is directly related to the programmer's knowledge about the OS.*

Resumo. *O desempenho de um programa de computador é influenciado pelo sistema operacional (SO). É possível melhorar o desempenho por meio da técnica de paralelismo. Para utilizar esta técnica, o desenvolvedor deve conhecer as funcionalidades do sistema operacional. Neste sentido, este trabalho propõe o desenvolvimento de um programa que utiliza paralelismo com a tecnologia Hyper-Threading e seu equivalente sem paralelismo. A partir disso, infere-se uma análise de desempenho traçando uma relação entre o rendimento do programa e o conhecimento acerca de sistemas operacionais. Os resultados mostraram que o desempenho de uma aplicação está diretamente relacionado com o conhecimento do programador sobre SO.*

1. Introdução

Os programas de computador, também chamados de aplicações dos usuários, são qualificados pelo tempo gasto ao realizar uma tarefa. Em outras palavras, entende-se que um programa possui bom desempenho se executa uma tarefa em pouco tempo [Deitel 2005].

O desempenho de um programa de computador pode ser influenciado por diversos fatores. Entre eles, destacam-se dois: os componentes físicos do computador (*hardware*: processador, memória e periféricos) e o sistema operacional (SO). Isto é, o computador deve possuir poder de processamento aliado ao uso eficaz do mesmo.

É comum encontrar nos computadores atuais processadores dotados de tecnologias que visam melhorar o desempenho das aplicações. Uma destas tecnologias é o Hyper-Threading, da Intel®, que possibilita ao processador executar mais de um processo simultaneamente. Esta técnica é conhecida como processamento paralelo, ou paralelismo.

O uso desta tecnologia, entretanto, está sujeita ao sistema operacional. Isto porque o processador, bem como todo o *hardware*, é acessado pelas aplicações por intermédio do SO. Cabe portanto, ao desenvolvedor, determinar de maneira explícita, como e se o programa utilizará o recurso. Para que isto seja possível, o programador deve conhecer sobre sistemas operacionais.

O conhecimento sobre o funcionamento dos sistemas operacionais é obtido, geralmente, em disciplinas específicas sobre o assunto. Estas disciplinas costumam ser ofertadas nos cursos de graduação e/ou técnicos pertencentes a área de informática.

Neste sentido, o trabalho proposto apresenta uma análise de desempenho para um programa que aplica os conceitos de paralelismo, estudado na disciplina de SO. Neste contexto, o objetivo é demonstrar a relação entre o desempenho do programa e os conhecimentos acerca de SO. Este trabalho foi desenvolvido por alunos do IFPR – Campus Paranaguá sob orientação do professor da disciplina. As próximas seções descrevem, em detalhes, o desenvolvimento deste trabalho.

2. Fundamentação Teórica

Nesta seção são descritos os conceitos necessários para o desenvolvimento do trabalho proposto, que incluem: sistemas operacionais e seu funcionamento; gerência de processos e paralelismo; a tecnologia Hyper-Threading e; abordagens alternativas empregadas no ensino de sistemas operacionais.

2.1. Sistemas Operacionais

Os sistemas operacionais (SO) modernos são programas responsáveis por gerenciar os componentes físicos do computador (*hardware*) ([Tanenbaum 2010], [Silberschatz 2013]). Isto significa que para ter acesso ao processador e memória, por exemplo, as demais aplicações devem, obrigatoriamente, requisitá-los para o SO.

Para gerenciar e coordenar as tarefas básicas desempenhadas pelo SO, o sistema dispõe de quatro grandes funcionalidades ([Silberschatz 2013], [Tanenbaum 2010], [Deitel 2005], [Maziero 2018]). Uma destas funcionalidades é a gerência de processos. Ela é responsável por fornecer, às aplicações, acesso ao processador. Ainda, ela coordena para que o uso deste recurso não seja monopolizado por alguma aplicação.

Outra funcionalidade é a gerência de memória. Ela tem por objetivo primário alocar e endereçar os programas na memória física do computador. O sistema de arquivos, por sua vez, tem o objetivo de gerenciar os dados criados pelas aplicações dos usuários. Já a gerência de entrada e saída (E/S) tem como objetivo permitir que periféricos (Ex.: teclado e mouse) possam trocar informações com o computador de maneira segura e eficiente.

Cada funcionalidade contribui para o correto funcionamento do sistema. Todas atuam de maneira integrada. Compreender, porém, esta complexa integração é uma tarefa difícil e requer alto grau de abstração [Maziero 2002]. Sendo assim, muitas vezes

os sistemas operacionais são desmembrados e suas funcionalidades são estudadas de maneira isolada. Neste sentido, a próxima seção descreve os principais aspectos da gerência de processos e suas funcionalidades.

2.2. Gerência de Processos

A gerência de processos controla o acesso e utilização do processador. Isto é necessário porque geralmente existem mais processos a serem executados do que processadores disponíveis [Tanenbaum 2010]. A gerência de processos baseia-se em um conjunto de conceitos que são implementados pelos sistemas operacionais [Maziero 2018]. Entre estes conceitos, os principais são:

Processo: É um estado dinâmico de um programa. Isto é, um editor de textos é apenas um programa. Ao iniciar sua execução ele se torna um processo no SO.

Contexto e Troca de Contexto: São as informações de controle sobre o estado atual de um processo, entre elas: identificador do processo e conteúdo de variáveis. A troca de contexto é o ato de armazenar e recuperar as informações do contexto de um processo no momento em que ele deixa e retorna para o processador, respectivamente.

Escalonamento de processos: Este conceito diz respeito ao método aplicado para tornar um processo elegível para o processador. Isto é, como as filas de processos, aptos a utilizar o processador, serão criadas e os processos nelas organizados.

Threads: O termo “thread” ou “threading” se refere a menor sequência de instruções programadas que podem ser gerenciadas independentemente pelo sistema operacional [Lamport 1979]. Tanenbaum (2010) define uma thread como um fluxo de execução que pode ser associado a outros fluxos (threads) para formar um processo macro. O conceito de thread está vinculado ao conceito de paralelismo, descrito a seguir.

2.3. Paralelismo

O conceito de paralelismo, também conhecido como processamento paralelo (ou programação paralela), refere-se a capacidade computacional de executar dois ou mais processos simultaneamente ([Silberschatz 2013], [Deitel 2005], [Maziero 2018]). Este recurso, quando utilizado, permite que programas sejam executados (e concluídos) com menor tempo de duração.

A implementação do paralelismo está diretamente vinculada ao conceito de threads. Uma thread representa uma parte da execução de um todo. Isto significa que, para alguns tipos de tarefas, elas podem ser divididas em pequenos fluxos de execução ([Silberschatz 2013], [Deitel 2005], [Maziero 2018]).

A fim de ilustrar este conceito, considera-se um processo que retorna o resultado de uma função matemática: x = (2 - b) + (3 * a). Neste exemplo, as partes em evidência podem ser executadas, cada qual, por uma thread. Isto implica em duas possibilidades: a) executar mais processos no mesmo intervalo de tempo, ou b) reduzir o tempo necessário para completar a tarefa. A Figura 1 ilustra o conceito descrito. É possível observar na figura que o processo **P** foi dividido em duas threads (**P' e P''**).

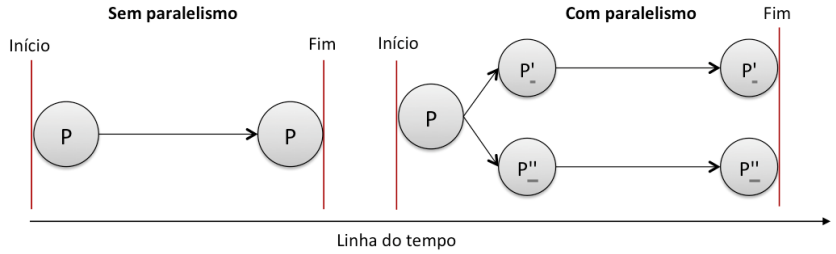

Figura 1. Processos com paralelismo *versus* processos sem paralelismo

Para criar threads e obter o paralelismo, um programa deve fazer uso das chamadas de sistema. Uma chamada de sistema é um recurso de *software* oferecido pelo SO para que as aplicações solicitem acesso aos dispositivos de *hardware* ([Silberschatz 2013], [Tanenbaum 2010]). Geralmente são acessadas por meio de bibliotecas do sistema. Sendo assim, ao criar threads, um programa pode requisitar o uso de todos os processadores (ou núcleos) disponíveis no computador. A chamada de sistema, no entanto, é acionada pelo desenvolvedor do programa.

Os processadores modernos oferecem tecnologias de suporte a programação paralela. Entre as tecnologias disponíveis, destaca-se a Hyper-Threading da Intel®, descrita a seguir.

2.4. Hyper-Threading

Em 2002 a Intel® lançou comercialmente, com o Pentium® 4, a tecnologia Hyper-Threading. Segundo a fabricante, esta tecnologia usa os recursos do processador de maneira mais eficiente [Intel 2017]. Ainda segundo a Intel® (2017), a tecnologia atua no agendamento dos processos permitindo que várias partes do mesmo programa sejam executadas em núcleos diferentes simultaneamente, diminuindo assim o tempo total de execução.

Em geral, processadores que possuem a tecnologia Hyper-Threading podem melhorar o desempenho de aplicações permitindo o uso do paralelismo. Estudos anteriores mostram que esta tecnologia melhora o desempenho de aplicações em até 30% [LENG et. al. 2002]. Geralmente, processadores com este recurso são melhor aproveitados quando o programa executado tem sua estrutura desenvolvida de forma a aproveitar essa tecnologia [BULPIN e PRATT 2004].

É notório, portanto, que a combinação de múltiplos processadores com múltiplos processos pode produzir programas que melhoram o desempenho e maximizam o uso do *hardware*. Para isto, é necessário que os desenvolvedores das aplicações saibam utilizar os recursos combinados (processos e processadores) para prover paralelismo às aplicações. Ainda que seu uso pareça trivial, ele deve ser tratado pelos programadores. Isto é, o paralelismo não é obtido de forma automática apenas por possuir mais processadores no computador [Deitel 2005]. Neste sentido, a próxima seção descreve abordagens alternativas como estratégias de ensino de sistemas operacionais a fim de melhorar o aprendizado dos alunos e futuros desenvolvedores.

2.5. Ensino de Sistemas Operacionais

Conforme apresentado na subseção 2.4, fica evidente a relevância acerca do estudo sobre sistemas operacionais. Assim sendo, os estudantes dos cursos de informática

devem ser estimulados a conhecer e dominar os conceitos relacionados ao sistema operacional, em especial a gerência de processos.

A Sociedade Brasileira de Computação (SBC) publica, periodicamente, um guia que é utilizado como referência para a composição das ementas curriculares dos cursos de graduação [SBC 2017]. Neste guia, a disciplina de SO é sempre sugerida como componente curricular, indiferentemente do curso ou área de especialidade.

Ainda que seja uma disciplina importante, o ensino de sistemas operacionais nos cursos de graduação é uma tarefa complexa [Maziero 2002]. Isto porque a abordagem aplicada muitas vezes se reduz a exposição teórica do conteúdo. Neste sentido, alguns trabalhos propõem soluções alternativas visando melhorar o aprendizado.

Um trabalho que atua neste contexto é o proposto por Nascimento e Nascimento (2014). O objetivo deste trabalho é fornecer aos estudantes um simulador das funcionalidades da gerência de memória, em especial da memória virtual. Neste simulador o aluno pode visualizar o funcionamento dos algoritmos de substituição de páginas bem como criar o seu próprio.

Na mesma linha, o trabalho proposto por Romero et. al. (2012), envolve a criação de um sistema web que simula o comportamento de memória cache. Neste trabalho, os autores descrevem o desenvolvimento da solução apresentando-a como ferramenta de estudo que aborda especificamente a hierarquia de memória.

O trabalho de Ribeiro et. al. (2014) difere-se dos anteriores no sentido de abranger também os conceitos da gerência de processos. O trabalho propõe o desenvolvimento de dois simuladores. Um para a gerência de memória e outro para a gerência de processos. Assim, o aluno pode compreender e visualizar graficamente o comportamento dos algoritmos que coordenam as tarefas tanto de memória quanto de processos.

Ampliando a abrangência, em Schimitz et. al. (2006) é apresentada uma abordagem alternativa aos métodos tradicionais de ensino que propõe o desenvolvimento de um simulador completo que abrange as quatro grandes funcionalidades de um SO. Isto é, neste simulador o aluno pode estudar sobre gerência de processos, memória, entrada e saída e sistemas de arquivos.

Da mesma maneira, Caetano et. al. (2011) propõe a integração de quatro simuladores distintos cada qual atendendo uma das quatro grandes funcionalidades. Neste trabalho o objetivo não foi desenvolver os simuladores, mas sim, aplicá-los em aula para mensurar sua eficiência no processo de ensino/aprendizagem.

Diante do exposto, pode-se notar que há um esforço plurilateral em relação ao processo de qualificar o ensino de sistemas operacionais. Neste sentido, é proposto uma abordagem alternativa como instrumento de estudos sobre SO, em especial gerência de processos. Para isto, este trabalho consiste em avaliar o desempenho de programas que foram desenvolvidos para utilizar o conceito de paralelismo por meio de threads.

3. Metodologia

Conforme descrito na subseção 2.5, o estudo dos sistemas operacionais é de suma importância para o desenvolvimento acadêmico e profissional de estudantes dos cursos de informática em geral. Assim sendo, é importante que o aluno possa validar os conhecimentos teóricos obtidos em aula com experimentos práticos. Diante disto, este

trabalho descreve uma atividade prática proposta para a disciplina de sistemas operacionais. A disciplina foi ministrada no curso superior em Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal do Paraná (IFPR), Campus Paranaguá.

Nesta atividade os alunos foram organizados em grupos. Foi sugerido para que os grupos buscassem desenvolver soluções nas quais pudessem ser aplicados os conceitos estudados em SO. Um destes grupos sugeriu a implementação e execução de um programa escrito em C para que este utilizasse paralelismo por meio da tecnologia Hyper-Threading. O objetivo do experimento era codificar, executar, registrar os tempos e, por meio de uma análise estatística, comprovar a eficácia de programas que utilizam processamento paralelo.

Para tal, o trabalho foi divido em duas etapas. A primeira diz respeito ao desenvolvimento de um programa capaz de aplicar e validar, os conceitos de paralelismo. A segunda tem como objetivo: realizar experimentos a partir do programa desenvolvido e; analisar os resultados a fim de comprovar a importância do ensino de sistemas operacionais para a formação profissional do aluno. As duas etapas são descritas nas próximas seções.

4. Desenvolvimento do Programa

A primeira etapa deste trabalho é o desenvolvimento de um programa capaz de aplicar os conceitos de paralelismo e multiprogramação. Para isto, foi proposto a implementação de um programa capaz de executar uma ou mais threads, acionando assim, um ou mais núcleos de processamento.

O programa foi escrito utilizando a linguagem de programação C juntamente com a biblioteca **pthreads.h**. Esta biblioteca permite ao programa realizar chamadas de sistema para acionar um ou mais núcleos de processamento disponíveis no *hardware*. A tarefa do programa consiste em efetuar cálculos de matriz transposta com tamanho de 4096 x 4096. O tamanho de 4096 corresponde a quantidade de endereços (páginas) que o processador pode manipular sem efetuar transferências entre memórias [Maziero 2018]. Isto significa que ao utilizar este tamanho de matriz, o processador obrigatoriamente utilizará toda a memória disponível para realizar o cálculo. Assim, ele necessitará efetuar troca de dados com a memória principal tornando o programa (propositalmente) lento. Esta técnica foi aplicada para que fosse possível registrar as medições de tempo.

Após a implementação do programa, o mesmo foi submetido aos experimentos para medir o desempenho (em tempo de execução). Esta medição pode fornecer informações para ratificar os conceitos vistos em aula.

5. Experimentos

Para a obtenção dos resultados foram realizados 4 (quatro) experimentos com 10 (dez) repetições (testes) para cada. Para cada experimento foi definido uma quantidade específica de threads (núcleos ativos), que variou de acordo com os seguintes valores: 1, 2, 4 e 8. Os testes foram realizados em um computador com processador Intel® Core i7 (6700) 3.40 GHz; Memória cache de 8MB; 8GB de memória RAM e sistema operacional Ubuntu LTS 16.04 (x64). Para cada rodada de testes foram registradas a

quantidade de threads ativas e o tempo de execução (em milissegundos). Os resultados dos testes são apresentados na Tabela 1.

Tabela 1. Resultados absolutos dos experimentos

Teste	Tempo de execução em milissegundos (ms)			
	1 Thread	**2 Threads**	**4 Threads**	**8 Threads**
1	300096	46745	10881	4700
2	293522	47263	10204	4710
3	135125	77574	9955	4459
4	288125	47487	9010	4693
5	198784	34176	13676	4725
6	279959	33939	8752	4449
7	240228	44197	14866	4454
8	273544	39653	12605	4417
9	134911	62775	13466	5065
10	251946	47728	17927	4721

6. Análise dos Resultados

A fim de sistematizar a análise dos resultados foram aplicadas técnicas Estatísticas. Com isso, têm-se diversas ferramentas que podem ser empregadas para a análise de resultados, visando os mais diferentes objetivos. Um deles trata da Análise de Variância (ANOVA), que tem como objetivo testar se as médias de duas ou mais populações são iguais [Barbetta 2010]. É um procedimento usado para comparar a distribuição de três ou mais grupos em amostras independentes.

No presente trabalho, a ANOVA foi empregada para validar a existência de diferença entre as médias obtidas em conjuntos de resultados [Barbetta 2010]. Neste trabalho, o que se espera é responder a seguinte pergunta: existe diferença real (significativa) entre os testes realizados com diferentes threads? A Tabela 2 apresenta o emprego da ANOVA.

Dentre os resultados retornados pelo cálculo da ANOVA, além do cálculo das médias e das variâncias por número de threads (a), tem-se a análise de variância (b). A ANOVA é composta dos valores **SQ, gl, MQ, F, valor-P e F crítico**, sendo que para este trabalho apenas os valores **F** e **F crítico** tem relevância. Isto porque, na prática, estes dois valores são comparados.

Tabela 2. Teste da Anova (Análise de Variância)

(a) Anova: fator único

Número de Threads	*Contagem*	*Soma*	*Média*	*Variância*
1 Thread	10	2396240	239624	3,93E+09
2 Threads	10	481537	48153,7	1,74E+08
4 Threads	10	121342	12134,2	8515195
8 Threads	10	46393	4639,3	39920,23

(b) ANOVA

Fonte da variação	*SQ*	*gl*	*MQ*	***F***	*valor-P*	***F crítico***
Entre grupos	3,67E+11	3	1,22E+11	**118,9556**	9,69E-19	**2,866266**
Dentro dos grupos	3,7E+10	36	1,03E+09			
Total	4,04E+11	39				

Como pode ser observado na Tabela 2, os valores destacados mostram o valor calculado para "F" e o valor "F crítico". Logo, como "F" > "F crítico", conclui-se, pela teoria de ANOVA, que estatisticamente, existe diferença entre as quatro sequências de testes apresentadas acima. As demais colunas (SQ, gl, MQ e valor-P) são valores base para o cálculo de F e F crítico.

Os resultados obtidos podem ser visualizados e comparados através de um diagrama em caixas. Neste tipo de gráfico, podem ser visualizados os valores máximos e mínimos, além do primeiro e terceiro quartil (valores que dividem a distribuição em quatro partes iguais). Observa-se a figura a seguir.

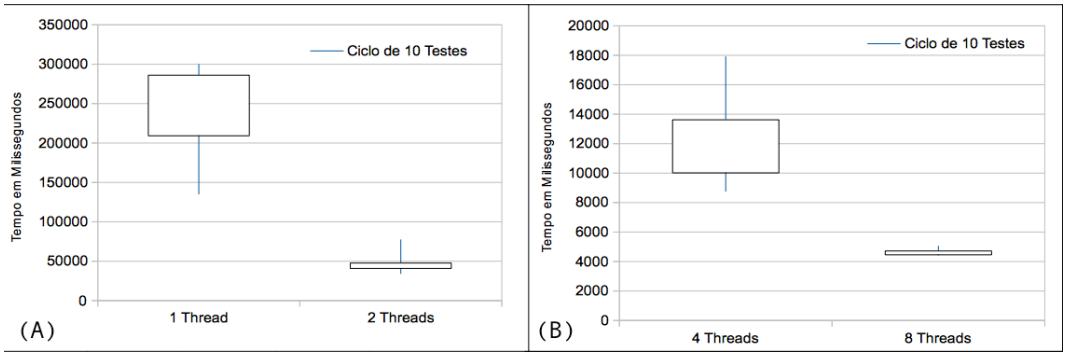

Figura 2. Tempo de execução de 10 testes para (A) 1 e 2 threads e (B) 4 e 8 threads

De acordo com a Figura 2(A), pode-se observar que, 50% das anotações de tempo para 1 thread estão entre 200000ms e 300000ms. Já para 2 threads, 50% das observações de tempo estão entre 4000ms e 5000ms. Os tempos obtidos para 2 threads, tendem a ser mais próximos de um valor típico, pois seu desvio interquartílico é menor. Fora do intervalo entre os quartis, observa-se uma cauda mais longa para baixo para 1 thread, o que significa, que existem alguns tempos abaixo do primeiro quartil. Para 2 threads, a cauda mais longa está para cima, logo, existem tempos acima do terceiro quartil.

Para a Figura 2(B) com 4 threads e 8 threads, observa-se que 50% das anotações de tempo para 4 threads estão entre 10000ms e 14000ms, enquanto que para 8 threads, os tempos estão entre 4000ms e 5000ms. Os valores obtidos para 8 threads tendem a ser mais próximos de um valor típico, pois seu desvio interquartílico é o menor de todos os observados. Para o 4 threads, observa-se uma cauda mais longa para cima, o que significa que existem alguns valores que estão acima do terceiro quartil. Um comportamento parecido pode ser observado para 8 threads.

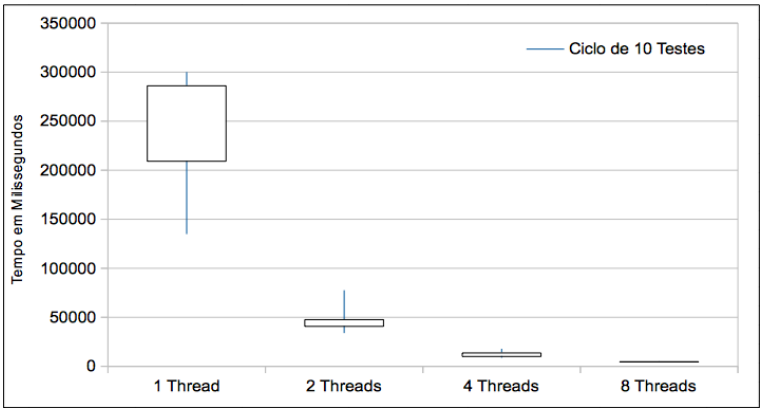

Figura 3. Tempo de execução de 10 testes para 1, 2, 4 e 8 threads

Da Figura 3, observa-se que o menor tempo computacional obtido é o de 8 threads, mostrando os resultados diretos do ganho de desempenho utilizando o processamento paralelo com a tecnologia Hyper-Threading.

É possível notar, portanto, que a análise dos resultados comprova a eficácia de um programa desenvolvido para utilizar o paralelismo. Desta forma, é factível afirmar que o desempenho de um programa está diretamente relacionado ao conhecimento que o programador possui sobre os recursos disponíveis de sistemas operacionais. Isto é, ao tomar conhecimento sobre a tecnologia Hyper-Threading e, codificar um programa para usar esta tecnologia, o programador está aplicando o conhecimento obtido sobre SO para melhorar o desempenho do programa. Isto ficou evidente ao constatar que, durante o desenvolvimento deste trabalho, os alunos puderam comprovar na prática os efeitos positivos de programas escritos para otimizar os recursos de hardware.

7. Considerações Finais

Após a análise dos resultados obtidos, é possível constatar que programas de computador podem ter seu desempenho afetado de acordo com o *hardware* no qual é executado. Para contornar esse desafio, o desenvolvedor deve ser capaz de codificar rotinas que se adéquem ao equipamento e sistema operacional utilizado. Para tal, é necessário ao programador conhecimentos relacionados a aspectos internos do sistema operacional e o entendimento sobre como manipular adequadamente seus recursos.

Como mostrado nos experimentos, sempre que uma aplicação é desenvolvida para utilizar o paralelismo ela obtém melhor tempo de resposta. Isto só é possível graças ao conhecimento que o programador possui sobre o *hardware* e o sistema operacional que o opera. Este conhecimento por sua vez é obtido, na grande maioria das vezes, nos cursos de formação deste profissional.

Fica claro, portanto, que o ensino de sistemas operacionais pode influenciar no desempenho dos programas por meio da atuação do profissional. Isto é, o conhecimento adquirido nos cursos de graduação pode ser utilizado para o desenvolvimento de aplicações mais eficientes.

8. Referências

Ap Romero S. Frata Peres F. Habib El Khouri J. Ferramenta Educacional Web para Simulação de Memória Cache. XX Workshop sobre Educação em Computação, 2012.

Barbetta, P. A.; Reis, M. M.; Bornia, A. C. Estatística para cursos de Engenharia e Informática. 3ª edição. Editora Atlas, São Paulo, 2010.

Bulpin, J. R.; Pratt, I. A.: Multiprogramming Performance of the Pentium 4 with Hyper-Threading. In: the Third Annual Workshop on Duplicating, Deconstructing and Debunking (WDDD2004), pp 53-62, Munique, Alemanha (Junho, 2004)

Caetano De Souza C., Rodrigues M. T., Nóbrega De Sousa G. R., Davi T De Sousa N. Silva E. Ribeiro De A. R. Um Ambiente Integrado de Simulação para Auxiliar o Processo de Ensino/Aprendizagem da Disciplina de Sistemas Operacionais. XVII Workshop de Informática na Escola, 2011. ISSN: 2176-4301

Deitel, H. M. Sistemas Operacionais: terceira edição. H. M. Deitel, P. J. Deitel, D. R. Choffnes, Pearson/Prentice Hall, 2005.

Intel® Hyper-Threading Technology, disponível em: <https://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html> Acessado em: Outubro de 2017.

Lamport, L.: "How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs", Volume C-28. p.690, 1979.

Leng, T.; Ali, R.; Hsieh, J.; Mashayekhi, V.; Rooholamini, R.: An Empirical Study of Hyper-Threading in High Performance Computing Clusters: Proceedings of LCI International Conference on Linux Clusters. The HPC revolution. 2002

Maziero, Carlos A,. Reflexões sobre o ensino prático de sistemas operacionais. Workshop de Ensino de Informática da SBC, 2002.

Maziero, Carlos A. Sistemas Operacionais. E-book. disponível em: <http://wiki.inf.ufpr.br/maziero/doku.php?id=so:livro_de_sistemas_operacionais>. Acessado em: Agosto de 2018.

Nascimento Fonseca F do. Maristela S. Nascimento F. Uma Ferramenta de Simulação do Processo de Substituição de Páginas em Gerência de Memória Virtual. XXXIV Congresso da Sociedade Brasileira de Computação, 2014.

Ribeiro T. Lucas Bernardes R. Lobo E. Simulador de Rotinas do Sistema Operacional para Auxílio as Aulas Teóricas. SBSI - Simpósio Brasileiro de Sistemas de Informação, 2014.

SBC, Curriculos de Referência. Disponível em: <http://www.sbc.org.br/documentos-da-sbc/send/131-curriculos-de-referencia/1165-referenciais-de-formacao-para-cursos-de-graduacao-em-computacao-outubro-2017>. Acessado em: Agosto de 2018.

Schimitz de Carvalho D. da Rocha Balthazar G. Rodrigo Dias C. Antônio Pereira Araújo M. Henrique Rezende Monteiro P. Simulador para a Prática de Sistemas Operacionais. XIV Workshop sobre Educação em Computação, 2006.

Silberschatz, A. Fundamentos de Sistemas Operacionais: princípios básicos 1a ed., LTC, 2013.

Tanenbaum, A. S. Sistemas Operacionais modernos. 3a ed., Pearson/Prentice Hall, 2010.