Mapeamento Sistemático do Ensino Teórico e Prático de Programação Paralela

Naylor G. Bachiega[1], **Paulo S. L. Souza**[1],
Sarita Bruschi[1], **Simone do R. S. de Souza**[1]

[1]Instituto de Ciências Matemáticas e de Computação (ICMC)
Avenida Trabalhador São-carlense – 400 – São Carlos – Brazil

naylor@usp.br,{pssouza,sarita,srocio}@icmc.usp.br

Abstract. *Providing parallel programming education is an emerging challenge, requires teaching methods to promote the learning process and also an elaborate infrastructure to provide a suitable environment for practical laboratory classes. Also, conventional teaching methods are used to share knowledge in the area, sometimes not being appropriate for this purpose. Effective learning methods, software configuration facility and the necessary laboratory infrastructure are needed. This paper presents a survey of recent research and challenges about parallel programming teaching. The main contribution of this work is to group state of the art in this context, highlighting the teaching practices available in the literature, to determine consolidated initiatives in development.*

Resumo. *Fornecer educação em programação paralela é um desafio emergente, requer métodos de ensino para promover o processo de aprendizagem e também uma infraestrutura para fornecer um ambiente adequado para as aulas de laboratório. Além disso, os métodos de ensino convencionais que são utilizados para compartilhar o conhecimento, às vezes não são adequados para esse propósito. Métodos de aprendizagem eficazes, configuração de software e uma infraestrutura de laboratório são requisitos necessários. Assim, este artigo apresenta uma revisão sistemática sobre o ensino de programação paralela. A principal contribuição deste trabalho é reunir o estado da arte neste contexto, destacando as principais práticas de ensino disponíveis na literatura.*

1. Introdução

O ensino de Computação de Alto Desempenho ou HPC (*High-Performance Computing*), ao contrário de outros tipos de computação, aborda desafios e condições especiais. O ensino abrange grandes quantidades de informações, problemas computacionais e a compreensão das plataformas suportadas, onde os alunos precisam de capacidades e habilidades para operar esses ambientes especiais [Shamsi et al. 2015].

[Zarza et al. 2012] destacam a importância de HPC como uma ferramenta valiosa para a sociedade. HPC fornece o desenvolvimento de uma grande quantidade de aplicativos e serviços. Também abordam que o estudo de arquiteturas paralelas é uma das questões-chave para estudantes acadêmicos de computação, devido ao uso geral de computadores em diversas disciplinas.

A Programação Paralela, uma parte de HPC, tornou-se ativa e intrínseca às tecnologias atualmente disponíveis, principalmente considerando os distintos modelos de programação provenientes dessas diferentes arquiteturas paralelas.

DOI: 10.5753/cbie.wcbie.2017.1089

No entanto, esse tipo de ensino traz muitos desafios. Na abordagem teórica, é necessário verificar se o método de ensino é eficaz para o aluno adquirir competências para um determinado assunto. Na abordagem prática, é importante definir o ambiente de trabalho e a arquitetura básica para a realização dos exercícios práticos de modo a desenvolver as habilidades esperadas.

Nesse sentido, esse trabalho realizou uma revisão sistemática da literatura para reunir os principais trabalhos que englobam as abordagens de ensino para programação paralela.

2. Abordagem Teórica

Para a abordagem teórica, muitas metodologias de ensino visam que o aluno adquira competências específicas para uma determinada disciplina ou assunto. A Tabela 1 da Seção 4 mostra algumas abordagens encontradas nos trabalhos investigados, como programação de padrões, abordagens baseadas em projetos, centradas em modelos, construtivistas, sala de aula invertida, entre outras.

Outras abordagens encontradas na Seção 4, baseiam-se na taxonomia da Bloom para definir níveis de aprendizagem [Bloom et al. 1956]. Essas abordagens geralmente atingem o nível de aplicação desta taxonomia. A seguir, são definidos alguns conceitos encontrados nos trabalhos que fazem parte desse mapeamento sistemático:

- ***Feedback*:** é uma abordagem utilizada para orientar e incentivar os alunos a entender e refletir sobre suas respostas. Assim, pode promover a mudança na forma de pensar, atuar e no processo de ensino e aprendizagem. Além disso, o *feedback* pode contribuir para a motivação dos estudantes com respostas positivas, sugestões e dicas de resposta. [Gonçalves 2017].
- **Sala de aula invertida:** é um modelo de ensino que inverte o modelo pedagógico tradicional. Nesse contexto, antes da aula, os elementos típicos de leitura e da lição de casa são vistos pelos alunos. Desta forma, os alunos observam pequenas instruções de vídeo ou outro material em casa, antes da aula, enquanto o tempo em sala de aula é dedicado para discussões e exercícios [Moore and Dunlop 2016].
- **Sala de aula parcialmente invertida:** tem o mesmo conceito que a sala de aula invertida, mas neste caso, apenas uma pequena quantidade de tempo no final da aula é dedicada para esta abordagem [Grossman et al. 2017].
- **Aprendizagem baseada em projetos:** é uma abordagem centrada no aluno, tornando o processo de ensino e aprendizagem mais dinâmico, no qual desafios e problemas reais são direcionados para a sala de aula, proporcionando um conhecimento mais realista para os estudantes [Zarestky and Bangerth 2014].
- **Programação de padrões:** usa um conjunto de padrões que fornecem regras detalhadas para um determinado problema. Por exemplo, comparar o uso de vários padrões para construir laços em Java, dando regras com base no número de iterações necessárias para o laço, usando um pseudocódigo. No entanto, a determinação de padrões efetivos continua sendo um problema de projeto, exigindo sua remodelação constante, tornando-a intrínseca em um curso de computação [Clancy and Linn 1999].
- **Abordagem analítica:** é um método de aprendizagem construtivista no qual os alunos iniciam os estudos através da compreensão dos componentes fundamentais

ou dos blocos de construção e, em seguida, aplicam o conhecimento adquirido para resolver problemas mais complexos [Ul-Abdin and Svensson 2015].

- **Aprendizagem baseada em problemas:** é uma abordagem que tem o conceito semelhante à aprendizagem baseada em projetos. Utiliza problemas específicos como um desafio para os alunos, sendo ordenados em um crescente grau de complexidade e contemplando assuntos aprendidos. Os alunos identificam problemas relacionados à sua área de trabalho e tentam resolvê-los de acordo com os critérios informados pelo professor [Cuenca and Giménez 2016].
- **Aprendizado centrado em modelos:** é uma abordagem construtivista, na qual os estudantes constroem conhecimento, experimentam o meio ambiente, observam resultados e extraem conclusões. Esta teoria, criada por [Gibbons 2001], define que o propósito da instrução é ajudar os alunos a desenvolver conhecimentos sobre os objetos e eventos em seu ambiente.

Além das abordagens teóricas, os artigos investigados neste trabalho oferecem alternativas e ferramentas para as aulas práticas de laboratório. Essas abordagens práticas também foram relacionadas na Tabela 1 da Seção 4.

3. Abordagem Prática

[Kim et al. 2016] destacam a importância dos laboratórios para o processo de ensino e aprendizagem, na qual a experiência é um componente essencial para a educação em programação paralela, proporcionando aos alunos oportunidades para praticar e aplicar os conceitos aprendidos e observados.

Deve-se notar que computadores paralelos e infraestruturas que contêm supercomputadores são caras e dificilmente são utilizadas pelos diversos cursos de Ciências da Computação. Embora, esses recursos possam estar disponíveis remotamente em uma infraestrutura compartilhada, ainda há os custos de gerenciamento, problemas de acesso, criação de usuários e largura de banda [Li et al. 2008].

De acordo com [Ivica et al. 2009], além dessas despesas, pode-se citar ainda custos de hardware, custos associados à hospedagem, energia e refrigeração para um *cluster* típico. Uma outra configuração desejável, para um laboratório, é alterar um *cluster* para utilização em salas de aula, onde podem ser necessários pacotes diferenciados, além da criação de contas de estudantes e manuais de acesso para professores e usuários.

Outro ponto relevante é desenvolver metodologias práticas que se adaptem na estrutura que faz parte da realidade dos alunos e das condições financeiras de suas escolas. Em algumas instituições de ensino, por exemplo, há apenas um microcomputador para um ou mais estudantes.

[Jiang and Song 2015] também salientam que, em infraestruturas compartilhadas com outros cursos ou disciplinas, existem a coexistência de outros programas que usam partes do espaço de armazenamento e recursos do computador, limitando o número de recursos disponíveis para a execução de programas paralelos.

[Kim et al. 2016] enfatizam que as máquinas virtuais têm sido utilizadas intensamente nas últimas décadas para o ensino de tecnologias da informação. No entanto, de acordo com [Pahl 2015], esse modelo de virtualização requer mais recursos para seu próprio suporte, maior armazenamento de disco e sua inicialização é lenta.

Assim, o trabalho investigado apresenta soluções para o desafio de fornecer uma infraestrutura para estudantes e professores. Algumas das abordagens envolveram o desenvolvimento de software proprietário, *clusters* de alto custo, infraestrutura compartilhada, ambientes virtuais e máquinas virtuais.

4. Mapeamento Sistemático

Nesta seção são apresentados os trabalhos que enfatizam o ensino de programação paralela. O mapeamento sistemático foi realizado com a estratégia PICO, onde uma *string* de busca, nas principais bases de dados, foi definida: *"(teach* or education or learn*) and parallel (programming or computing) and instructional (design or unit)"*.

Foram encontrados 180 artigos, dos quais 24 foram selecionados pela relação com o tema. Os pontos relevantes foram agrupados na Tabela 1, a qual mostra as abordagens utilizadas para o ensino teórico e prático.

[Joiner et al. 2006] demonstram ferramentas para o ensino de computação de alto desempenho através de dois métodos educacionais. O primeiro método usa um CD inicializável que transforma temporariamente a estação de trabalho em um *cluster* para a execução de código paralelo. O segundo método utiliza um protótipo de *cluster*. Para a parte de ensino, eles utilizaram uma implementação prática e coleta de *feedback*.

[Marowka 2008] demonstram um curso introdutório em programação paralela para estudantes do terceiro ano de graduação em Ciências da Computação. Abordam tópicos representativos, teóricos e práticos. O curso está focado em três pontos: como apresentar aos alunos que a computação paralela está em todos os lugares; como treinar os estudantes para que pensem em paralelo; e qual modelo de programação paralela é usado para a prática desse tipo de programação.

[Li et al. 2008] abordam um curso de 16 semanas com o objetivo de apresentar princípios e técnicas de computação paralela e distribuída para estudantes de informática. O conteúdo do curso é dividido em três áreas: programação paralela, algoritmo paralelo e arquitetura paralela de computador. As aulas são divididas em teóricas e práticas, utilizando OpenMP e MPI para o laboratório.

[Ivica et al. 2009] apresentam um sistema de educação de programação paralela consistindo em uma imagem de máquina virtual pré-configurada que, através de *scripts*, cria um *cluster* baseado na *Amazon Elastic Computing Cloud* (EC2). Esse *cluster*, chamado StarHPC, é compartilhado pela classe para executar códigos paralelos.

[Arroyo 2013] propõe um conjunto de módulos para ensinar programação paralela, começando pela computação básica, computação de alto desempenho e temas de programação paralela e distribuída. Além disso, eles propõem uma biblioteca em C++ com um conjunto de padrões paralelos para desenvolvimento de programas. Esta biblioteca foi chamada pdt (*Parallel e Distributed Templates*) e se concentra no ensino de programação através de padrões ou esqueletos paralelos.

[Adams et al. 2013] mostram duas abordagens para ensinar programação paralela. A primeira abordagem é o uso de padrões de programação paralela baseados na experiência dos profissionais. Esses padrões são entregues aos alunos como uma coleção de problemas a serem resolvidos. A segunda abordagem envolve o uso de aplicativos que

Tabela 1. Trabalhos Relacionados com Estratégias de Ensino [a]

			Estratégias de Ensino		Ferramentas para Laboratórios		
Artigo	**T**	**P**	**Abordagem**	**Método**	**Infraestrutura**	**SP**	**Método**
[Joiner et al. 2006]	x	x	Feedback	Exemplos	Máquina Virtual Paralela (PVM)	x	BCCD, MPICH e openMosix
[Marowka 2008]	x	x		Questionários, exemplos e apresentação			Visual Studio e C++
[Li et al. 2008]	x	x		Exemplos			OpenMP e MPI
[Ivica et al. 2009]		x			Máquina Virtual (VM)	x	StarHPC, OpenMP e MPI
[Arroyo 2013]		x	Programação de padrões	Classificação de Bloom		x	pdt, OpenMP, MPI e OpenCL
[Adams et al. 2013]	x		Programação de padrões	Padrões OPL			
[Ferner et al. 2013]		x	Programação de padrões			x	Java e compilador Paraguin
[Branco et al. 2013]	x	x		Apresentação e exemplos			TinyOS
[Shafi et al. 2014]	x	x		Apresentação e exemplos			Java *Threads*, OpenMP, Intel Cilk Plus e MPJ Express
[Burkhart et al. 2014]	x	x	Aprendizagem baseada em projetos			x	Java, Chapel, OpenMP, MPI, Patus e CUDA
[Zarestky and Bangerth 2014]	x		Sala de aula invertida	Pesquisa de artigos e escrita reflexiva			
[Shamsi et al. 2015]	x	x	Aprendizagem baseada em programação	*Feedback*, aprendizagem interativa, discussão de pares e discussões em grupo			MPI, OpenMP, Hadoop e GPGPU
[Dolgopolovas et al. 2015]	x	x	Aprendizado centrado em modelos (Construtivista)	Exemplos			OpenMP e MPI
[Ul-Abdin and Svensson 2015]	x	x	Analítico (Construtivista)	Conferência, laboratório, projeto e seminário	Open Virtual Platform (OVP)	x	aDesigner, GT-Board e C
[Nezu 2015]	x	x		Exercícios			C e OpenMP
[Schlarb et al. 2015]		x			Web	x	Sistema para Avaliação de Código Atribuído (SAUCE)
[Cuenca and Giménez 2016]	x	x	Aprendizado baseado em problemas	Apresentações e exemplos	Sistema simulado		OpenMP e MPI
[Liu 2016]	x	x		Pesquisa, exercícios e apresentações	UMA, *Clusters* Beowulf e GPUs		Microsoft's Visual Studio *Task Parallel* e C#
[Eijkhout 2016]	x	x		Exemplos e oficina intensiva			MPI
[Moore and Dunlop 2016]	x	x	Sala de aula invertida	Palestra, exercício, discussão em classe, ajuda individual, vídeo e questionários	Stampede *Supercomputer*		C e MPI
[Baldwin et al. 2016]		x			Hathi *Cluster*		Scholar, *Thin-line,* Rstudio *Server* e Jupyterhub
[Gardner and B. 2017]		x				x	Biblioteca *Pilot*
[Grossman et al. 2017]	x	x	Sala de aula parcialmente invertida	Questionários, vídeos e tarefas		x	HJlibparallel e Habanero Autograder
[Capel et al. 2017]	x	x	Programação de padrões	Exemplos e exercícios	*Cluster* próprio		

[a] **(T) Teórica**, **(P) Prática, (SP)** ***Software*** **Proprietário**

utilizam técnicas de programação paralela para resolver algum problema real, motivando os alunos a aprender os princípios e práticas desse tipo de programação.

[Ferner et al. 2013] desenvolveram materiais educacionais baseados em duas abordagens para o ensino de programação paralela para estudantes de graduação. A primeira abordagem utiliza um software proprietário, o qual cria uma abstração de alto nível para a programação paralela e distribuída com base na abordagem de programação de padrões. A segunda abordagem utiliza diretrizes do compilador para descrever como um programa deve ser paralelizado.

[Branco et al. 2013] propõem um curso com duas abordagens para estudantes avançados de graduação ou pós-graduação. O curso utiliza livros como materiais expositivos. Na parte prática, no laboratório, os alunos executam códigos preparados e, depois de entendê-los, os estudantes podem alterá-los.

[Shafi et al. 2014] apresentam uma visão geral do curso de graduação em Engenharia de Software. Basicamente, o curso foi dividido em três seções, sendo que a primeira trata de técnicas de programação paralela para sistemas de memória compartilhada. A segunda seção discute ferramentas paralelas usadas para memória distribuída e *cluster*. A última abordagem inclui tópicos avançados como MapReduce e o GPGPU (*General Purpose Computing on Graphics Processing Units*).

[Burkhart et al. 2014] mostram uma abordagem baseada em projetos para resolver problemas específicos. Neste artigo, duas abordagens são definidas, autônoma e integrada. Na abordagem autônoma, os alunos precisam configurar o ambiente de desenvolvimento, gerar os dados e analisar os resultados. Na abordagem integrada, desenvolveu-se uma ferramenta pedagógica que torna o ambiente pré-configurado e integrado. Neste ambiente, as ferramentas necessárias estão disponíveis para resolver problemas e analisar os resultados.

[Zarestky and Bangerth 2014] descrevem o uso da sala de aula invertida no ensino de HPC. Neste tipo de abordagem, o instrutor oferece a lição de casa e após, a atividade da sala de aula é realizada. O curso foi administrado com base em três componentes: formato invertido, pesquisa de artigos (os alunos devem descrever o que viram no trabalho de casa) e a escrita reflexiva (os alunos devem analisar aquilo que aprenderam). Assim, os autores conseguiram concluir que os alunos tiveram um maior envolvimento com o conteúdo do curso, maior motivação e facilidade de comunicação com o instrutor.

[Shamsi et al. 2015] realizam uma abordagem prática com base na programação e no uso de plataformas múltiplas de HPC. A carga teórica do curso é extensa e cobre todos os assuntos planejados para um curso de HPC. Para a metodologia de ensino são utilizadas técnicas de *feedback*, aprendizagem interativa, discussão entre pares e discussões em grupo.

[Dolgopolovas et al. 2015] apresentam uma metodologia baseada em uma abordagem construtivista. Esta abordagem é centrada em modelo e aprendizagem comparativa, que utiliza modelos de programação baseados em simulações estocásticas fornecidas por um sistema de filas multi-fase. Este sistema de enfileiramento foi utilizado pela simplicidade e possibilidades de paralelização, permitindo realizar várias experiências, comparar e investigar a eficácia do método paralelizado. Os métodos de paralelização incluem memória compartilhada, distribuída e paralelização híbrida usando OpenMP e MPI.

[Ul-Abdin and Svensson 2015] fornecem uma abordagem analítica com foco no desempenho e na eficiência energética de sistemas embarcados. O curso foi dividido em duas partes (teórica e prática). A prática inclui um pequeno projeto e seminário. Os métodos de ensino baseiam-se na teoria da aprendizagem construtivista, onde os alunos estão envolvidos na construção do conhecimento e na análise de estudos de caso reais.

[Nezu 2015] apresenta um pequeno trabalho, onde são utilizados materiais contendo exercícios de programação paralela. A ideia básica do material é permitir que os alunos programem com seus computadores, utilizando o sistema operacional Linux.

[Cuenca and Giménez 2016] usam problemas de otimização bi-objetivo para ensinar programação paralela aos alunos do curso de Engenharia de Software. Esta aprendizagem baseada em problemas é focada no trabalho prático desses estudantes e consiste em otimizar o tempo de execução e o consumo de energia em um *cluster* primário heterogêneo. O curso dura um semestre e abrange tópicos como revisão das arquiteturas paralelas, paradigmas e ferramentas de programação paralela.

[Liu 2016] enfatiza o uso de codificação em suas aulas. A necessidade de priorizar a codificação também ajudará o aluno a ganhar experiência na aquisição de conceitos como comunicação de processos, bloqueio, balanceamento de carga, entre outros. *Clusters* foram utilizados para as aulas de laboratório com algumas arquiteturas possíveis, como UMA, GPUs e *clusters* Beowulf. Os alunos precisaram realizar projetos de programação prática, bem como pesquisas e apresentações.

[Eijkhout 2016] propõem uma sequencia de etapas para o ensino de programação paralela com MPI. O artigo descreve os exercícios e oficinas necessários para completar o processo de ensino e aprendizagem, começando com mecanismos que enfatizam a simetria entre processos no modelo de programação SPMD (*Single Program Multiple Data*). Também enfatiza que o OpenMP deve ser ensinado após o MPI, devido ao modelo de paralelismo mais intuitivo, pois será mais difícil para os alunos entenderem o modelo simétrico do MPI, caso eles adquiram conceitos como *thread* mestre e regiões paralelas.

[Moore and Dunlop 2016] utilizam a abordagem pedagógica chamada sala de aula invertida, na qual os elementos típicos de leitura de um curso são invertidos. Para atingir esse objetivo, antes do início da aula, pequenos vídeos e palestras são enviados aos alunos. Após, em sala de aula, o tempo é dedicado para exercícios e discussões sobre os conceitos cobertos no vídeo. Em comparação com a abordagem do ensino tradicional, o artigo conclui que essa nova abordagem motivou os alunos e mostrou melhores resultados. Além disso, eles foram agrupados misturando estudantes mais experientes com menos experientes. A turma também foi composta por alunos de graduação e pós-graduação.

[Baldwin et al. 2016] apresentam uma ferramenta de ensino chamada Scholar para alfabetização computacional. Esta ferramenta é utilizada nas salas de aula para ensinar HPC e realizar experimentos. Para fornecer acesso à computação de alto desempenho, o Scholar tem acesso ao *cluster* "Hathi", um sistema HPC de 576 nós, construído com processadores Intel Xeon E5-2660 de 2,6 GHz. Cada nó possui 64 GB de RAM e está interconectado com Infiniffer FDR de 56Gb.

[Gardner and B. 2017] mostram em seu artigo os relatórios sobre a experiência do ensino paralelo para estudantes de graduação. Eles apresentam uma biblioteca chamada Pilot, que representa uma camada acima do MPI para ajudar os alunos na programação

por passagem de mensagens.

[Grossman et al. 2017] apresentam um curso introdutório de programação paralela, discutindo sua estrutura e abordagens pedagógicas. O curso usa sala de aula parcialmente invertida, sendo esta aplicada nos últimos minutos de aula. Também utilizam questionários, vídeos e trabalhos de casa para aumentar os problemas pedagógicos. Eles usaram as ferramentas HJlib e Habanero Autograder para aulas de laboratório.

[Capel et al. 2017] criaram um modelo de ensino que envolve uma abordagem demonstrativa para a construção de programas paralelos. Os exemplos são selecionados através de fragmentos de código e um conjunto padrão de algoritmos. Estes casos foram disponibilizados aos estudantes através do Moodle, onde os alunos em um ciclo de trabalho, seguem as seguintes etapas: o professor explica um exemplo em uma palestra; os estudantes desenvolvem o programa para resolver um problema específico; e, finalmente, eles analisam o resultado e fazem os exercícios.

[Schlarb et al. 2015] apresentam a criação de um *software* para avaliação e medição automatizada de código fonte de programas paralelos. O sistema, chamado de SAUCE, é um *software* de código aberto e escrito em Python, executado por um navegador da *Web*. Esta ferramenta fornece *feedback* imediato e pode ser usada de forma interativa na sala de aula, além de fornecer suporte para MPI, OpenMP e CUDA.

5. Conclusão

Neste artigo foram apresentados os principais trabalhos encontrados na literatura que utilizam abordagens para o ensino teórico e prático de programação paralela. Apesar das diferenças nas abordagens, todos relatam ganhos no ensino e na aprendizagem.

Algumas das principais lacunas observadas são uma comparação mais profunda do impacto das abordagens propostas para os futuros egressos dos cursos de computação, ausência de iniciativas que procurem antecipar o ensino de programação paralela para os primeiros períodos nas grades curriculares, ausência de estudos que abordem a interdisciplinaridade da programação paralela com outros conteúdos vistos nos cursos como banco de dados, processamento de imagens, sistemas digitais, arquiteturas de computadores, sistemas operacionais e outros.

Outrossim, muitas das soluções descritas aqui são baseadas em *software* proprietário, que corre o risco de descontinuidade de manutenção, alto custo e abandono do projeto. Ressalta-se que a virtualização também foi extensamente utilizada para as aulas de laboratório, neste caso VMs, PVMs e OVPs.

Assim, esse mapeamento trouxe as atuais abordagens, utilizadas pelas comunidades científica e acadêmica, que envolvem o processo de ensino e aprendizagem. Esse levantamento poderá ajudar no direcionamento de novas metodologias e ferramentas que alavanquem e popularizem o ensino de HPC.

Referências

Adams, J., Brown, R., and Shoop, E. (2013). Patterns and exemplars: Compelling strategies for teaching parallel and distributed computing to cs undergraduates. In *IEEE Int.Symp.on Parallel Distributed Processing*, pages 1244–1251.

Arroyo, M. (2013). Teaching parallel and distributed computing to undergraduate computer science students. In *IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, pages 1297–1303.

Baldwin, M. E., Zhu, X., Smith, P. M., Harrell, S. L., Skeel, R., and Maji, A. (2016). Scholar: A campus hpc resource to enable computational literacy. In *2016 Workshop on Education for High-Performance Computing (EduHPC)*, pages 25–31.

Bloom, B. S., Engelhart, M. B., Furst, E. J., Hill, W. H., and Krathwohl, D. R. (1956). *Taxonomy of educational objectives. The classification of educational goals. Handbook 1: Cognitive domain*. Longmans Green, New York.

Branco, A., d. Moura, A. L., Rodriguez, N., and Rossetto, S. (2013). Teaching concurrent and distributed computing – initiatives in rio de janeiro. In *IEEE Int. Symp. on Parallel Distributed Processing, Workshops and Phd Forum*, pages 1318–1323.

Burkhart, H., Guerrera, D., and Maffia, A. (2014). Trusted high-performance computing in the classroom. In *Workshop on Education for High Perf Computing*, pages 27–33.

Capel, M. I., Tomeu, A. J., and Salguero, A. G. (2017). Teaching concurrent and parallel programming by patterns: An interactive ict approach. *J.Par.Dist.Comp.*, 105:42–52.

Clancy, M. J. and Linn, M. C. (1999). Patterns and pedagogy. *SIGCSE Bull.*, 31(1):37–42.

Cuenca, J. and Giménez, D. (2016). A parallel programming course based on an execution time-energy consumption optimization problem. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 996–1003.

Dolgopolovas, V., Dagienė, V., Minkevičius, S., and Sakalauskas, L. (2015). Teaching scientific computing: A model-centered approach to pipeline and parallel programming with c. *Sci. Program.*, 2015:11:11–11:11.

Eijkhout, V. (2016). Teaching mpi from mental models. In *2016 Workshop on Education for High-Performance Computing (EduHPC)*, pages 14–18.

Ferner, C., Wilkinson, B., and Heath, B. (2013). Toward using higher-level abstractions to teach parallel computing. In *2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, pages 1291–1296.

Gardner and B., W. (2017). Should we be teaching parallel programming? In *Proceedings of the 22Nd Western Canadian Conference on Computing Education*, WCCCE '17, pages 3:1–3:7, New York, NY, USA. ACM.

Gibbons, A. S. (2001). Model-centered instruction. *Journal of Structural Learning and Intelligent Systems*, 14(4):511–540.

Gonçalves, R. Q. (2017). Ensino de gerenciamento de projetos de software mediado por ferramentas. Master's thesis, Universidade Federal de Santa Catarina, Florianópolis.

Grossman, M., Aziz, M., Chi, H., Tibrewal, A., Imam, S., and Sarkar, V. (2017). Pedagogy and tools for teaching parallel computing at the sophomore undergraduate level. *J.Par.Dist.Computing*, 105:18 – 30.

Ivica, C., Riley, J. T., and Shubert, C. (2009). Starhpc - teaching parallel programming within elastic compute cloud. In *Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces*, pages 353–356.

Jiang, K. and Song, Q. (2015). A preliminary investigation of container-based virtualization in information technology education. In *Proceedings of the 16th Annual Conf. on Inf. Tech. Education*, SIGITE '15, pages 149–152, New York, NY, USA. ACM.

Joiner, D. A., Gray, P., Murphy, T., and Peck, C. (2006). Teaching parallel computing to science faculty: Best practices and common pitfalls. In *Proceedings of the Eleventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '06, pages 239–246, New York, NY, USA. ACM.

Kim, T.-H., Jiang, K., and Rajput, V. S. (2016). Adoption of container-based virtualization in it education. In *ASEE's 123rd Annual - Conference and Exposition*, New Orleans.

Li, J., Guo, W., and Zheng, H. (2008). An undergraduate parallel and distributed computing course in multi-core era. In *2008 The 9th International Conference for Young Computer Scientists*, pages 2412–2416.

Liu, J. (2016). 20 years of teaching parallel processing to computer science seniors. In *Workshop on Education for High-Performance Computing (EduHPC)*, pages 7–13.

Marowka, A. (2008). Think parallel: Teaching parallel programming today. *IEEE Distributed Systems Online*, 9(8):1–1.

Moore, S. V. and Dunlop, S. R. (2016). A flipped classroom approach to teaching concurrency and parallelism. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 987–995.

Nezu, N. (2015). Teaching parallel programming in 50 minutes. *J. Comput. Sci. Coll.*, 31(2):18–24.

Pahl, C. (2015). Containerization and the paas cloud. *IEEE Cloud Computing*, 2(3):24–31.

Schlarb, M., Hundt, C., and Schmidt, B. (2015). *SAUCE: A Web-Based Automated Assessment Tool for Teaching Parallel Programming*, pages 54–65. Springer International Publishing, Cham.

Shafi, A., Akhtar, A., Javed, A., and Carpenter, B. (2014). Teaching parallel programming using java. In *Proceedings of the Workshop on Education for High-Performance Computing*, EduHPC '14, pages 56–63, Piscataway, NJ, USA. IEEE Press.

Shamsi, J. A., Durrani, N. M., and Kafi, N. (2015). Novelties in teaching high performance computing. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, pages 772–778.

Ul-Abdin, Z. and Svensson, B. (2015). Towards teaching embedded parallel computing: An analytical approach. In *Proceedings of the Workshop on Computer Architecture Education*, WCAE '15, pages 8:1–8:6, New York, NY, USA. ACM.

Zarestky, J. and Bangerth, W. (2014). Teaching high performance computing: Lessons from a flipped classroom, project-based course on finite element methods. In *2014 Workshop on Education for High Performance Computing*, pages 34–41.

Zarza, G., Lugones, D., Franco, D., and Luque, E. (2012). An innovative teaching strategy to understand high-performance systems through performance evaluation. *Procedia Computer Science*, 9:1733 – 1742.