Opinião de Brasileiros Sobre a Produtividade no Desenvolvimento de Aplicações Paralelas

Gabriella Andrade[1], Dalvan Griebler[1], Rodrigo Santos[2], Luiz Gustavo Fernandes[1]

[1]Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

[2]Departmento de Informática Aplicada, Universidade Federal do Estado do
Rio de Janeiro (UNIRIO), Rio de Janeiro – RJ – Brasil

gabriella.andrade@edu.pucrs.br,

{dalvan.griebler,luiz.fernandes}@pucrs.br,rps@uniriotec.br

Resumo. *A partir da popularização das arquiteturas paralelas, surgiram várias interfaces de programação a fim de facilitar a exploração de tais arquiteturas e aumentar a produtividade dos desenvolvedores. Entretanto, desenvolver aplicações paralelas ainda é uma tarefa complexa para desenvolvedores com pouca experiência. Neste trabalho, realizamos uma pesquisa para descobrir a opinião de desenvolvedores de aplicações paralelas sobre os fatores que impedem a produtividade. Nossos resultados mostraram que a experiência dos desenvolvedores é uma das principais razões para a baixa produtividade. Além disso, os resultados indicaram formas para contornar este problema, como melhorar e incentivar o ensino de programação paralela em cursos de graduação.*

1. Introdução

Nos últimos anos, as arquiteturas paralelas se tornaram populares devido às limitações da indústria de silício em projetar novas Unidades Centrais de Processamento (UCPs) utilizando técnicas tradicionais [Pacheco 2011]. Para permitir a exploração do poder de processamento de tais arquiteturas, surgiram diferentes Interfaces de Programação Paralela (IPPs). As IPPs fornecem abstrações para que os desenvolvedores não precisem lidar com otimizações de hardware de baixo nível (por exemplo, localidade de memória) e aumentar a produtividade. Entretanto, o desenvolvimento de aplicações paralelas ainda é uma atividade para desenvolvedores experientes, pois o processo de desenvolvimento envolve muitos fatores, tais como identificar regiões paralelizáveis, identificar seções críticas, sincronização de dados e threads, explorar a concorrência, balanceamento de carga etc. [McCool et al. 2012]. Logo, esta contínua sendo uma tarefa desafiadora para desenvolvedores iniciantes devido à sua falta de conhecimento para realizar otimizações a fim de alcançar alto desempenho e desenvolver aplicações paralelas produtivamente.

Conforme a ISO 9241-11, a produtividade (ou eficiência) é um fator essencial que, juntamente com a eficácia e a satisfação do usuário, são indicadores de usabilidade. A produtividade mede os recursos utilizados em relação aos resultados alcançados, incluindo o esforço humano empregado no desenvolvimento de software, os custos de um projeto e os materiais utilizados [ISO 9241-11 2018]. A partir de indicadores de produtividade, é possível propor melhorias para projetar novas IPPs e refinar as existentes. Na programação paralela, a produtividade é comumente avaliada pelo esforço de desenvolvimento. Alguns pesquisadores conduziram experimentos controlados com estudantes para

avaliar a produtividade da codificação [Szafron e Schaeffer 1996, Griebler et al. 2014, Andrade et al. 2022]. Entretanto, planejar, executar e analisar os resultados de um experimento demanda tempo. Além disso, encontrar uma amostra representativa de participantes experientes no domínio da programação paralela é desafiador. Logo, o uso de métricas de codificação bem estabelecidas na área de Engenharia de Software (e.g., Linhas de Código (LOC), Halstead e Constructive Cost Model - COCOMO) pode facilitar a avaliação da produtividade [Adornes et al. 2015, Griebler et al. 2018, Miller et al. 2018, Rodriguez-Canal et al. 2021, Martínez et al. 2022, Peccerillo e Bartolini 2022]. Porém, nosso estudo anterior [Andrade et al. 2021] mostrou limitações, uma vez que estas métricas de codificação não foram projetados para avaliar aplicações paralelas.

Neste contexto, o objetivo deste trabalho é obter indicadores de produtividade em programação paralela a partir de uma pesquisa de opinião com desenvolvedores brasileiros. Neste estudo, realizamos um esforço para identificar os fatores que impactam o desenvolvimento de aplicações paralelas. Além disso, também pretendemos identificar possíveis formas de contornar esses problemas a fim de aumentar a produtividade dos desenvolvedores. Este estudo possui as seguintes contribuições científicas: *(i) caracterização do perfil dos desenvolvedores de aplicações paralelas brasileiros; e (ii) análise qualitativa das percepções dos desenvolvedores brasileiros sobre a produtividade na programação paralela.*

Este artigo está organizado da seguinte forma: a Seção 2 apresenta uma contextualização e os trabalhos relacionados, a Seção 3 apresenta a metodologia utilizada, a Seção 4 apresenta os resultados alcançados e a Seção 5 conclui este trabalho.

2. Contextualização e Trabalhos Relacionados

Pesquisas de opinião são normalmente realizadas para extrair informações de/sobre pessoas, procurando descrever, comparar ou explicar seus conhecimentos, atitudes e comportamentos [Wohlin et al. 2012]. A partir de uma pesquisa de opinião, é possível quantificar estatisticamente a opinião de uma determinada população [Glasow 2005]. A realização de uma pesquisa envolve três etapas principais: planejamento, execução e análise dos resultados. Na etapa de planejamento, realiza-se a definição dos objetivos, participantes, instrumentos de coleta e métricas para a análise dos resultados. A partir de uma revisão da literatura, é possível determinar os objetivos da pesquisa e a amostra dos participantes. O instrumento de coleta pode ser, por exemplo, um questionário online contendo perguntas abertas (discursiva) e/ou fechadas (múltipla escolha) com base nos objetivos do estudo. Além disso, devem ser definidas as métricas para a análise dos resultados, tais como teste de hipóteses ou estatística descritiva. Finalmente, o pesquisador pode conduzir a pesquisa de opinião e analisar os dados obtidos para relatar os resultados e as ameaças à validade [Glasow 2005, Wohlin et al. 2012].

Na área de programação paralela, foram realizadas diferentes pesquisas para determinar a opinião da comunidade sobre um assunto específico da área. [Meade et al. 2013] realizaram um estudo exploratório e uma pesquisa de opinião para investigar as ferramentas e práticas utilizadas pelos especialistas para realizar a decomposição de dados ao realizar a paralelização das aplicações. Em [Maiterth et al. 2018], foi realizada uma pesquisa de opinião para descobrir as técnicas de gerenciamento de recursos de energia e potência utilizadas em nove centros de computação de alto desempenho (*High Performance Computer* - HPC) nos Estados Unidos, Europa e Ásia. [Schlagkamp e Renker 2015] inves-

tigaram a satisfação dos usuários de cluster em relação ao tempo de espera necessário para executar aplicações paralelas. Em [Schlagkamp e Renker 2015], foi realizada uma pesquisa de opinião para identificar o uso do MPI no desenvolvimento de aplicações no projeto Exascale Computing dos Estados Unidos. [Hori et al. 2021] conduziram uma pesquisa de opinião online visando analisar a adoção do MPI no desenvolvimento paralelo de aplicações. Além disso, [Amaral et al. 2020] conduziram uma pesquisa de opinião com especialistas para avaliar os resultados obtidos em um mapeamento sistemático da literatura sobre modelagem e simulação de alto desempenho para aplicações de big data.

Alguns estudos buscam avaliar o ensino da programação paralela a partir de pesquisas de opinião. Em [López e Baydal 2018], o objetivo era avaliar as estratégias de ensino propostas para um curso de computação em cluster para estudantes de graduação ou graduados em Ciência da Computação ou Engenharia da Computação. Em [Memeti e Pllana 2018], uma ferramenta para ensinar e ajudar programadores iniciantes a evitar erros de programação (*Parallel Programming Assistant* - PAPA) foi avaliada por uma pesquisa de opinião com estudantes da Universidade de Linnaeus. Em [Ferdinandy et al. 2019], as reações de pesquisadores com pouca ou nenhuma formação em Tecnologia da Informação e Comunicação foram avaliadas durante seu primeiro contato com programação paralela. Além disso, [Plale et al. 2021] realizaram uma pesquisa de opinião na conferência *Student Cluster Competition* (SCC) a fim de coletar informações sobre iniciativas para permitir a replicabilidade dos estudos na área de HPC.

Neste estudo, realizamos uma pesquisa online para descobrir a opinião dos desenvolvedores brasileiros sobre a produtividade em programação paralela. Ao contrário dos trabalhos anteriores, nosso objetivo é determinar os fatores que impactam a produtividade. Para isso, avaliamos as opiniões dos participantes a partir de uma análise textual usando procedimentos de codificação inspirados na Grounded Theory (GT) [Corbin e Strauss 1990] e estatística descritiva.

3. Metodologia de Pesquisa

Neste estudo, o objetivo é descobrir os fatores que impedem o desenvolvimento de aplicações paralelas produtivamente, a partir da opinião de 50 desenvolvedores brasileiros. Para este fim, nossa pesquisa consistiu em três etapas: planejamento, execução e análise dos resultados. Cada uma das etapas é detalhada a seguir.

3.1. Planejamento

A primeira etapa consistiu em planejar a pesquisa. Para definir o instrumento de pesquisa utilizamos a abordagem GQM (*Goal/Question/Metrics*) [Caldiera e Rombach 1994], que é um sistema de medicação para planejar quais métricas serão utilizadas para interpretar os dados com base em um conjunto de questões e objetivos específicos. GQM é uma hierarquia com três níveis, conforme a Figura 1: objetivos das medições (G), questões (Q) e métricas de avaliação (M). Com base nessa abordagem, foram definidos três objetivos: G1) Identificar os fatores que impactam nao esforço de desenvolvimento de aplicações paralelas; G2) Traçar o perfil dos desenvolvedores de aplicações paralelas; e G3) Classificar as aplicações paralelas desenvolvidas. A Tabela 1 apresenta as 17 questões derivadas a partir dos objetivos, das quais seis são fechadas (Q1 à Q6), sete são de múltipla escolha (Q7 a Q13) e quatro são dissertativas. A partir do tipo de questão, definimos as métricas de análise utilizadas: estatística descritiva para questões fechadas e de múltipla escolha e procedimentos de codificação para questões abertas (Q14 a Q17).

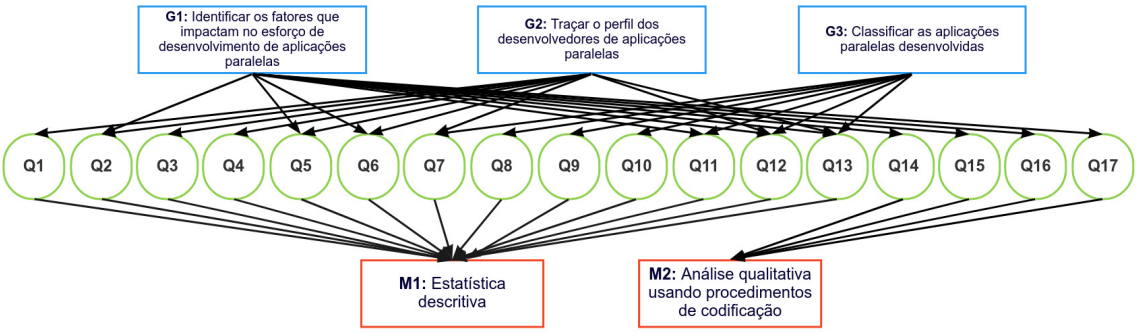


Figura 1. Hierárquia da abordagem GQM.

Tabela 1. Questionário aplicado aos participantes da pesquisa de opinião.

ID	Questão
Q1.	Qual é a sua nacionalidade?
Q2.	Qual é a sua formação acadêmica? (Ensino Médio, Curso Técnico, Graduação, Especialização, Mestrado, ou Doutorado)
Q3.	Qual é a sua filiação (empresa ou instituição de ensino)?
Q4.	Em que país sua afiliação está localizada?
Q5.	Qual é a sua experiência com programação paralela (em anos)? (De 0 a 1, de 1 a 2, de 2 a 5, de 5 a 10, ou mais de 10 anos)
Q6.	Que tipo de desenvolvedor você se considera? (Iniciante, Intermediário, Avançado, ou Outro)
Q7.	Que tipo de aplicações você desenvolve? (Científica, Comercial, ou Outro)
Q8.	Qual é o domínio destas aplicações?
Q9.	Você pode especificar os nomes das aplicações ou descrever o que as aplicações fazem?
Q10.	Como você desenvolve as aplicações?
Q11.	Quais são as linguagens de programação nas quais as aplicações são desenvolvidas?
Q12.	Quais IPPs ou ferramentas você usa para paralelizar as aplicações?
Q13.	Em que arquiteturas você implementa o paralelismo em suas aplicações?
Q14.	Na sua opinião, o que afeta a produtividade do desenvolvimento de aplicações paralelas?
Q15.	Na sua opinião, o que pode ser feito para tornar a programação paralela mais produtiva?
Q16.	Na sua opinião, quais IPPs você recomendaria para um programador iniciante desenvolver aplicações paralelas de forma mais produtiva? Por favor, justifique sua escolha.
Q17.	Na sua opinião, quais IPPs você acha que um desenvolvedor precisa ter mais experiência em programação paralela para ser capaz de paralisar uma aplicação de forma mais produtiva? Por favor, justifique sua escolha.

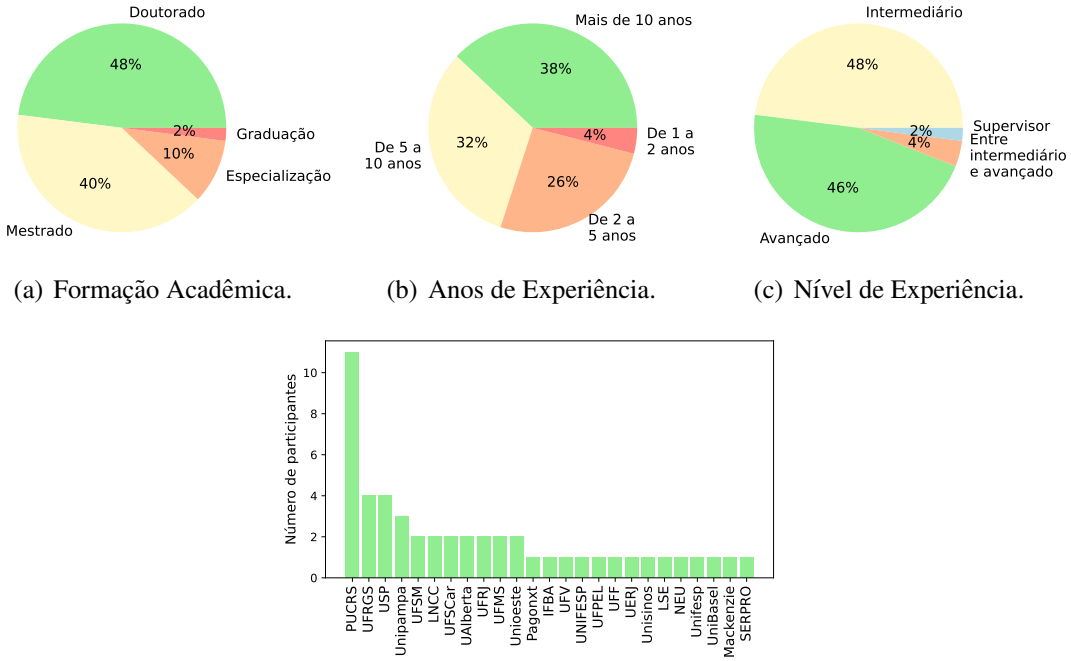
3.2. Execução

Os participantes foram identificados a partir de sua participação em conferências na área de HPC, como Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD), *International Symposium on Computer Architecture and High Performance Computing* (SBAC-PAD) e *Euromicro Workshop on Parallel, Distributed and Network-based Processing* (PDP). O perfil do Google Scholar[1] também foi avaliado. Após a seleção dos participantes, enviamos esta pesquisa por e-mail para 124 pessoas. A pesquisa esteve disponível de 28 de janeiro de 2022 a 8 de julho de 2022. Durante este período, enviamos alguns lembretes aos participantes. Após este prazo, foram obtidas 50 respostas (40,33%).

3.3. Análise

Neste estudo, realizamos uma análise quantitativa e qualitativa dos dados (Figura 1). Utilizamos técnicas de estatística descritiva, como a percentagem, para resumir e descrever os dados obtidos em perguntas fechadas e de múltipla escolha. Realizamos uma análise qualitativa dos dados baseada em codificação (aberta e axial), que consiste em um dos procedimentos iniciais de GT. Na codificação aberta, as respostas dos participantes foram analisadas detalhadamente para criar códigos relacionados à trechos específicos das respostas. Em seguida, foi realizada a codificação axial na qual as categorias foram identificadas e relacionadas a suas subcategorias, criando relações. A análise qualitativa foi realizada com a ferramenta ATLAS.ti[2]. A partir desta análise foram geradas representações

[1]Disponível em: `https://scholar.google.com.br`
[2]Disponível em: `https://atlasti.com/`



(a) Formação Acadêmica. (b) Anos de Experiência. (c) Nível de Experiência.

(d) Afiliação dos Participantes.

Figura 2. Perfil dos Participantes.

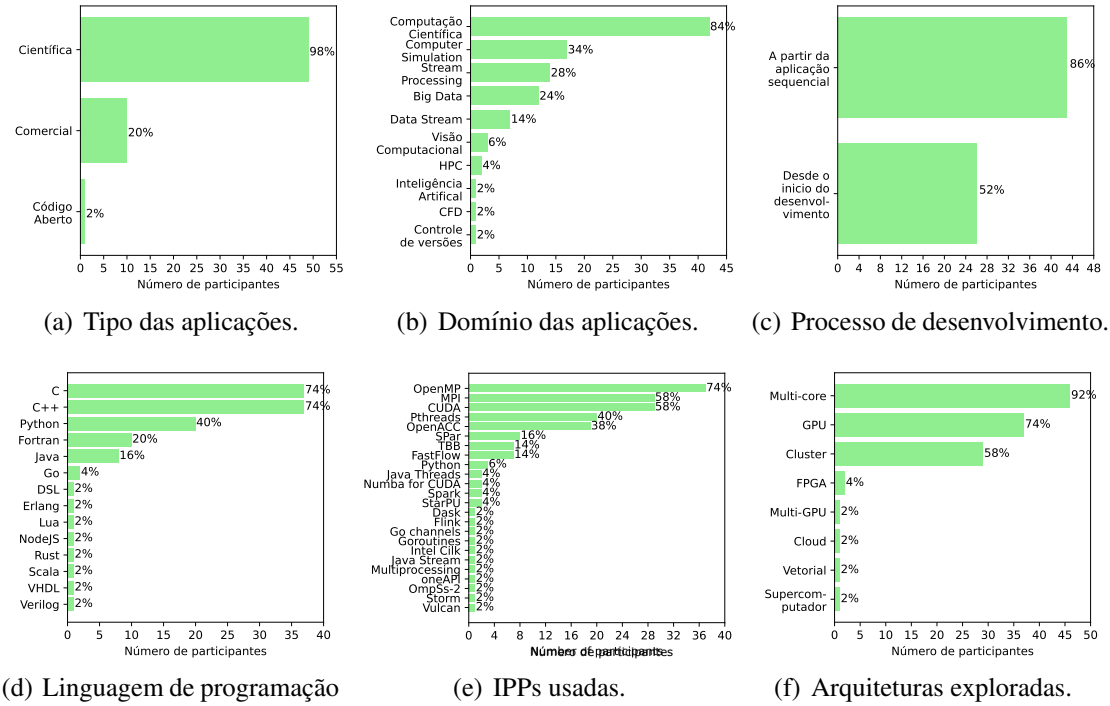
gráficas, nas quais as linhas vermelhas representam as relações entre códigos e categorias, e as linhas pretas com uma etiqueta descrevem o tipo de relação entre os códigos.

4. Resultados

4.1. Análise Quantitativa

Esta seção apresenta uma análise quantitativa das respostas dos participantes. A Figura 2 apresenta o perfil dos 50 participantes desta pesquisa, os quais são todos brasileiros. A Figura 2(a) mostra que a maioria dos participantes possui doutorado e mestrado (24 e 20, respectivamente). Apenas um participante possui especialização e dois participantes possuem graduação. A Figura 2(d) mostra que as respostas vieram de diferentes universidades no Brasil, principalmente da região sul, como a Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), a Universidade Federal do Rio Grande do Sul (UFRGS) e a Universidade Federal da Pampa (Unipampa). Cinco dos participantes brasileiros estudam e/ou trabalham em universidades internacionais. Além disso, dois atuam na indústria.

A Figura 2(b) mostra que a maioria dos participantes são desenvolvedores com muitos anos de experiência em programação paralela: 13 têm entre dois e cinco anos de experiência, 16 têm entre cinco e dez anos e 19 têm mais de dez anos de experiência. Além disso, apenas dois participantes possuem poucos anos de prática no desenvolvimento de aplicações paralelas, os quais atuaram nessa área em um período de um a dois anos. Entretanto, apenas os anos de prática não podem provar a perícia do desenvolvedor. Logo, pedimos aos participantes que classificassem seu nível de conhecimento em programação paralela entre iniciante, intermediário e avançado, como mostra a Figura 2(c). A maioria deles se considera de nível avançado e 23 acreditam ser desenvolvedores intermediários. Apenas dois acreditam que seu nível de conhecimento está entre o



(a) Tipo das aplicações. (b) Domínio das aplicações. (c) Processo de desenvolvimento.

(d) Linguagem de programação (e) IPPs usadas. (f) Arquiteturas exploradas.

Figura 3. Características das aplicações.

intermediário e o avançado. Além disso, um dos participantes se considera um supervisor, pois apenas supervisiona o desenvolvimento das aplicações e não as desenvolve.

A Figura 3(a) mostra o tipo de aplicações desenvolvidas pelos participantes deste estudo. Todos os participantes desenvolvem aplicações científicas, oito desenvolvem aplicações comerciais e um desenvolve aplicações de código aberto. A Figura 3(b) mostra o domínio destas aplicações, sendo que a maioria aplicações de computação científica (37 participantes). A simulação computacional é o segundo domínio mais explorado pelos participantes (15). Muitos participantes exploraram domínios de processamento de dados, tais como big data (26,32%), processamento de stream (15,79%) e data stream (10,53%). Outros quatro domínios são menos explorados: visão computacional, HPC, inteligência artificial, dinâmica dos fluidos computacionais (CFD) e controle de versão.

Com relação à exploração do paralelismo, a maioria dos participantes (32) desenvolve aplicações paralelas a partir de aplicações sequenciais. Por outro lado, 60,53% deles implementam o paralelismo desde o início do processo de desenvolvimento, como visto na Figura 3(c). Além disso, a Figura 3(d) mostra que estas aplicações são majoritariamente desenvolvidas com C e C++ (81,58%). Python é outra linguagem de programação muito utilizada pelos desenvolvedores (47,37%), seguida por Fortran (21,05%) e Java (13,16%). Erlang, Lua, Go, NodejS, Rust, VHDL e Verilog são as linguagens de programação menos utilizadas pelos respondentes.

A Figura 3(e) mostra as IPPs utilizadas pelos participantes para explorar o paralelismo. O OpenMP é uma das mais populares e, consequentemente, é a mais utilizada (34 participantes). MPI, outra IPP popular, é a segunda mais usada pelos participantes, seguida de CUDA (29 participantes cada). Pthreads e OpenACC também são ampla-

mente utilizadas (20 e 19 participantes). OpenMP, MPI, CUDA, Pthreads, e OpenACC são IPPs consolidadas, o que justifica estes resultados. Consequentemente, as arquiteturas mais exploradas pelos participantes ao implementar o paralelismo em suas aplicações são multi-core, GPU e cluster (Figura 3(f)).

4.2. Análise Qualitativa

Esta seção apresenta uma análise qualitativa das opiniões dos participantes sobre a produtividade. A Figura 4 mostra os fatores que impedem a produtividade conforme o ponto de vista dos respondentes (Q14). Os participantes relataram diferentes razões, como as relacionadas aos IPPs e seus modelos de programação e arquitetura. Uma das principais razões apresentadas pelos respondentes foi a dificuldade em compreender a aplicação a ser paralelizada. Além disso, é necessário entender o comportamento da aplicação para identificar possíveis dependências de dados, regiões críticas etc. Muitas vezes, é necessário reestruturar o código para permitir a paralelização. As citações abaixo destacam que este é um dos principais fatores que dificultam a paralelização de uma aplicação:

> "*A necessidade de reestruturar uma grande base de códigos sequenciais que pode não estar preparada para o paralelismo*." [Participante 23]
>
> "*Outro fator que afeta a produtividade, em minha opinião, é a reestruturação do código sequencial. Comumente, uma versão otimizada visando GPUs é irreconhecível, ou seja, você vê o código e ele é completamente diferente do código serial*." [Participante 46]

O nível de experiência do desenvolvedor foi também um dos principais fatores identificados. Os participantes destacam ser um desafio realizar otimizações para obter alto desempenho sem conhecer as IPPs utilizadas e a arquitetura alvo. Além disso, a falta de documentação adequada a respeito das IPPs e de cursos que abordem programação paralela durante a graduação dificulta o aprendizado dos estudantes.

A Figura 5 mostra as possíveis soluções apontadas pelos participantes para melhorar a produtividade dos desenvolvedores (Q15), tais como propor IPPs mais expressivas e concisas, melhorar a documentação das IPPs e melhorar o ensino de conceitos de programação paralela em cursos de graduação. Além disso, abstrair as particularidades das arquiteturas e realizar padronizações entre as arquiteturas e as IPPs são soluções apontadas pelos respondentes para oferecer portabilidade às aplicações.

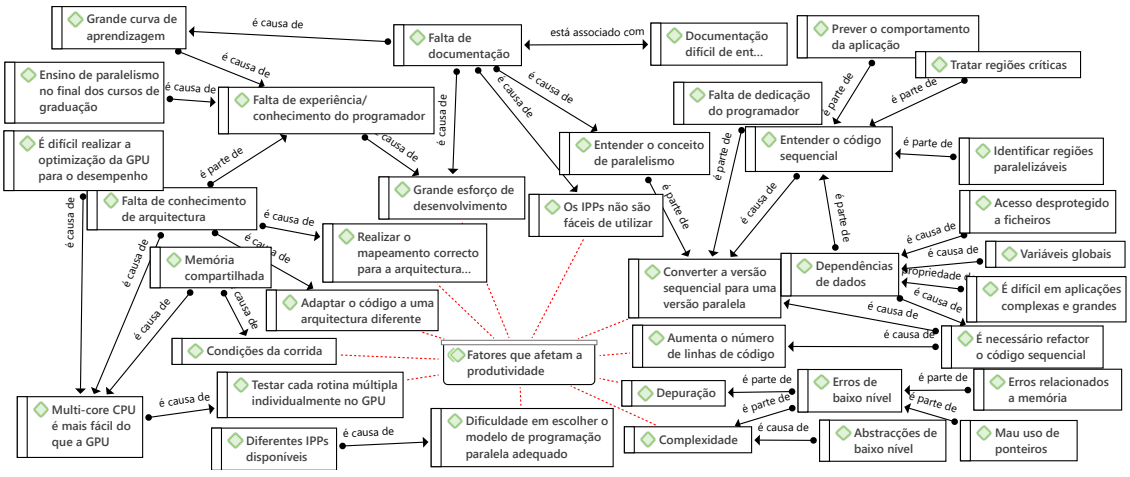


Figura 4. Fatores que impedem os programadores de desenvolver aplicações paralelas de forma produtiva.

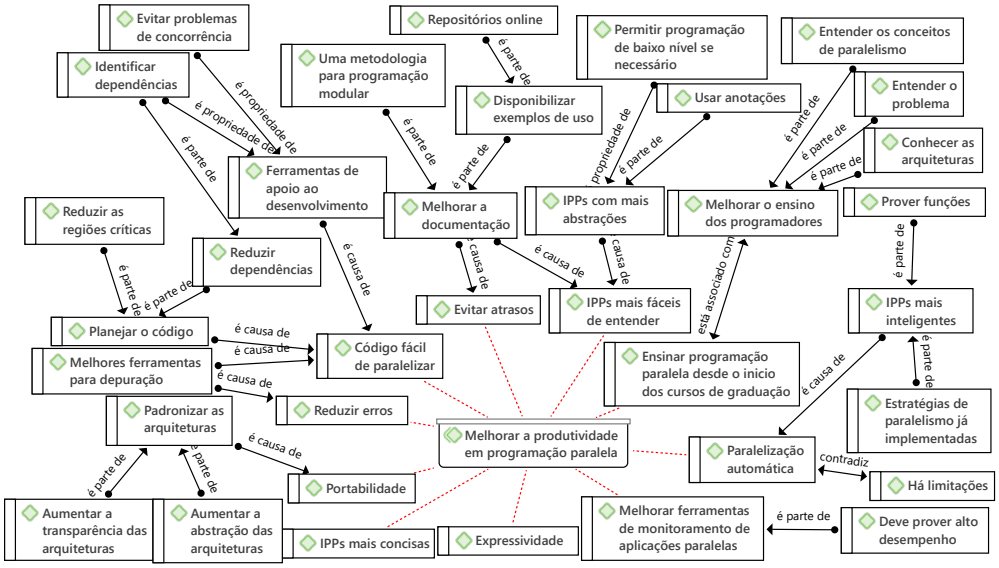


Figura 5. Possíveis soluções para aumentar a produtividade.

Os participantes ressaltam que prover IPPs inteligentes que paralelizam automaticamente o código seria uma solução perfeita. No entanto, há limitações ao projeto de tais IPPs. Uma solução proposta para este problema é a construção de ferramentas para apoiar o desenvolvimento, depuração e teste de aplicações paralelas. Se tais ferramentas não afetarem ainda mais a produtividade dos desenvolvedores, elas poderiam evitar erros de programação e facilitar a paralelização das aplicações. As citações abaixo indicam estes aspectos:

> "*Ferramentas que leem o código e visualizam as dependências ajudariam muito.*" [Participante 1]
>
> "*Ferramentas que auxiliam na depuração de bibliotecas e estruturas que cuidam de forma inteligente de muitas das responsabilidades do programador (como comunicação, cópias de dados, programação).*" [Participante 4]
>
> "*Tornando o IPP mais abstraído e inteligente*." [Participante 49]

Perguntamos aos participantes qual IPP eles recomendariam para iniciantes conseguirem desenvolver aplicações paralelas produtivamente (Q16). A Figura 6 mostra que a maioria dos participantes ($\approx$ 73%) sugere que os iniciantes comecem a desenvolver aplicações paralelas usando o OpenMP. A Figura 7 apresenta as principais razões pelas quais os participantes sugerem OpenMP, como a facilidade de compreensão e uso. É mais fácil para os desenvolvedores iniciantes desenvolverem aplicações paralelas para ambientes multi-core. Além disso, o uso de pragmas é mais simples para iniciantes, pois não requer tantas mudanças no código. As citações abaixo destacam estes aspectos:

> "*Eu recomendo C+OpenMP, já que o uso de pragmas é considerado fácil para iniciantes*." [Participante 12]
>
> "*OpenMP porque esta ferramenta oferece recursos que permitem implementar um código paralelo baseado no código serial (paralelismo incremental). Portanto, isto torna a programação mais fácil do que outras ferramentas que aplicam uma abordagem baseada na conversão de tudo ou nada de um programa inteiro.*" [Participante 31]
>
> "*Abordagens baseadas em anotações, tais como pragmas OpenMP e SPar, tendem a ser mais fáceis de usar*." [Participante 49]

A Figura 6 também mostra as IPPs indicadas pelos respondentes para que os especialistas desenvolvam aplicações paralelas de forma produtiva (Q17). Eles indicaram que os desenvolvedores experientes deveriam explorar arquiteturas diferentes das tradicionais multi-core, tais como GPU e cluster. As duas IPPs mais indicadas foram CUDA e MPI, pois exigem mais experiência dos desenvolvedores. Além disso, um respondente indicou usar MPI combinado com CUDA para obter mais desempenho.

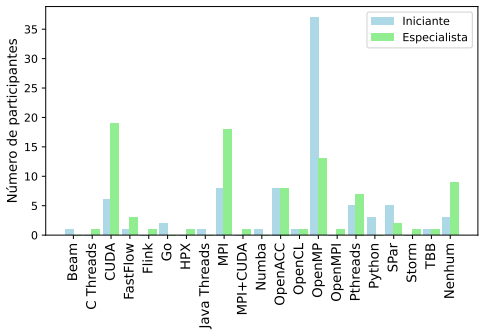


Figura 6. IPPs recomendadas para desenvolver aplicações paralelas.

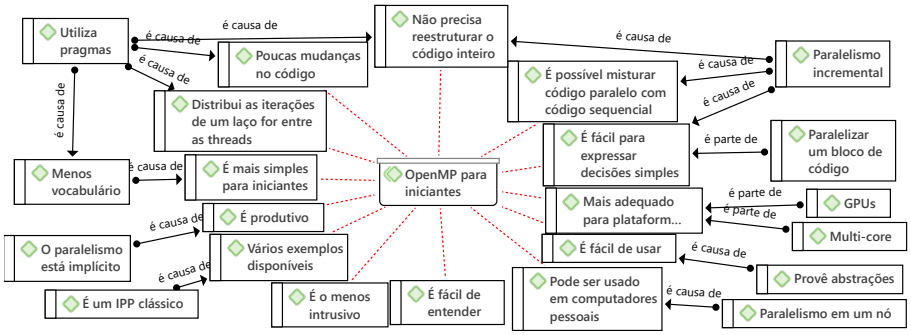
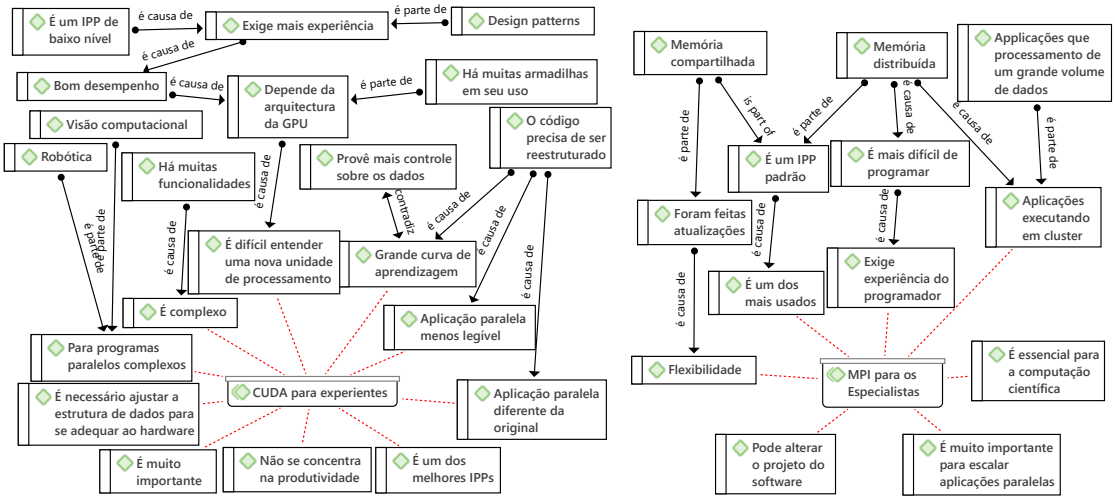


Figura 7. Razões para recomendar o OpenMP para iniciantes desenvolverem aplicações paralelas de forma mais produtiva.



(a) CUDA. (b) MPI.

Figura 8. Razões para recomendar CUDA e MPI para que os especialistas desenvolvam aplicações paralelas de forma mais produtiva.

A Figura 8 apresenta as principais razões apontadas pelos respondentes para indicar CUDA e MPI. Com CUDA, é possível desenvolver aplicações com bom desempenho, embora exija alguma experiência no desenvolvimento de aplicações paralelas e conhecimento de GPU. MPI é uma IPP padrão para memória distribuída. A execução em clusters pode trazer vantagens para aplicações de processamento intensivo de dados, embora exija mais experiência do desenvolvedor. Além disso, os participantes destacam a flexibilidade do MPI, que permite agora a exploração de arquiteturas de memória compartilhada.

Alguns participantes não sugeriram o uso de nenhuma IPP específica para iniciantes (3 participantes) ou especialistas (9 participantes). Para estes participantes, a IPP usada para a paralelização depende dos objetivos do desenvolvedor e da plataforma alvo. Alguns deles (2 participantes) não se sentiram capazes de responder a estas perguntas. Além disso, um participante julga que os iniciantes não deveriam lidar com aplicações paralelas, ao contrário da opinião da maioria dos outros participantes.

5. Conclusão

Neste estudo, realizamos uma pesquisa de opinião com brasileiros para identificar os fatores que impedem a produtividade em programação paralela. Identificamos algumas soluções possíveis para melhorar a produtividade com base nas opiniões dos participantes. Nossos resultados mostraram que a falta de experiência é uma das principais razões para o aumento do esforço de desenvolvimento. Estes resultados mostraram uma lacuna no ensino de programação paralela nas universidades brasileiras, onde os conceitos relacionados às IPPs são cobertos apenas no final dos cursos de graduação. Além disso, a falta de documentação das IPPs dificulta o processo de aprendizagem dos alunos.

Nossos resultados mostram que a programação para GPU é mais complicada do que a programação para multi-core. O nível de experiência do desenvolvedor em relação à arquitetura e ao modelo de programação utilizado também impacta negativamente o esforço de desenvolvimento. Entender como o código sequencial funciona também é essencial para alcançar um bom desempenho na versão paralela. Além disso, quando os desenvolvedores compreendem o comportamento da aplicação sequencial, eles podem evitar muitos erros de programação. Logo, concluímos que o nível de experiência do desenvolvedor é o principal fator que impacta na produtividade. Ou seja, quanto maior for o seu conhecimento a respeito das arquiteturas paralelas, IPPs, habilidades de programação, entre outros, maior será a produtividade de desenvolvimento.

De acordo com a maioria dos participantes, OpenMP é uma das IPPs mais fáceis para programação paralela e CUDA e MPI são as IPPs mais difíceis. Em trabalhos futuros, pretendemos realizar um experimento como o realizado em nosso trabalho anterior [Andrade et al. 2022] a fim de validar esses resultados.

Este estudo fornece descobertas a respeito da produtividade na programação paralela. No entanto, algumas ameaças à validade permanecem. O método de coleta pode ameaçar a validade interna do estudo, uma vez que os participantes podem relatar dados incorretos. O perfil dos participantes pode ser considerado uma ameaça à validade externa, pois avaliamos apenas a opinião de desenvolvedores brasileiros. Logo, o estudo não pode ser generalizado para o nível internacional. A amostra de participantes pode ser considerada uma ameaça à validade de conclusão, uma vez que a maioria dos participantes é do meio acadêmico. Para contornar essas limitações, em trabalhos futuros,

pretendemos realizar esta pesquisa internacionalmente. Além disso, pretendemos incluir mais participantes que atuam no desenvolvimento de aplicações paralelas na indústria.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 e o projeto FAPERGS 10/2020-ARD SPAR4.0 (Nº 21/2551-0000725-7). O terceiro autor agradece a UNIRIO e FAPERJ (Proc. 211.583/2019) pelo apoio parcial para este trabalho.

Referências

Adornes, D., Griebler, D., Ledur, C., e Fernandes, L. G. (2015). A Unified MapReduce Domain-Specific Language for Distributed and Shared Memory Architectures. In *The 27th Int. Conf. on Software Engineering & Knowledge Engineering*, pages 619–624, Pittsburgh, USA. Knowledge Systems Institute Graduate School.

Amaral, V., Norberto, B., Goulão, M., Aldinucci, M., Benkner, S., Bracciali, A., Carreira, P., Celms, E., Correia, L., Grelck, C., et al. (2020). Programming languages for data-intensive hpc applications: A systematic mapping study. *Parallel Computing*, 91:1–17.

Andrade, G., Griebler, D., Santos, R., Danelutto, M., e Fernandes, L. G. (2021). Assessing Coding Metrics for Parallel Programming of Stream Processing Programs on Multi-cores. In *47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 291–295, Pavia, Italy. IEEE.

Andrade, G., Griebler, D., Santos, R., e Fernandes, L. G. (2022). A parallel programming assessment for stream processing applications on multi-core systems. *Computer Standards & Interfaces*, pages 1–27.

Caldiera, V. R. B. G. e Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of software engineering*, pages 528–532.

Corbin, J. M. e Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative sociology*, 13(1):3–21.

Ferdinandy, B., Guerrero-Higueras, Á. M., Verderber, É., Miklósi, Á., e Matellán, V. (2019). Analysis of users' first contact with high-performance computing: first approach with ethology researchers. In *Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality*, pages 554–557.

Glasow, P. A. (2005). Fundamentals of survey research methodology. *Mitre*.

Griebler, D., Adornes, D., e Fernandes, L. G. (2014). Performance and Usability Evaluation of a Pattern-Oriented Parallel Programming Interface for Multi-Core Architectures. In *The 26th International Conference on Software Engineering & Knowledge Engineering*, pages 25–30, Vancouver, Canada. KSI.

Griebler, D., Hoffmann, R. B., Danelutto, M., e Fernandes, L. G. (2018). High-Level and Productive Stream Parallelism for Dedup, Ferret, and Bzip2. *International Journal of Parallel Programming*, 47(1):253–271.

Hori, A., Jeannot, E., Bosilca, G., Ogura, T., Gerofi, B., Yin, J., e Ishikawa, Y. (2021). An international survey on mpi users. *Parallel Computing*, 108:1–13.

ISO 9241-11 (2018). Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts.

López, P. e Baydal, E. (2018). Teaching high-performance service in a cluster computing course. *Journal of Parallel and Distributed Computing*, 117:138–147.

Maiterth, M., Koenig, G., Pedretti, K., Jana, S., Bates, N., Borghesi, A., Montoya, D., Bartolini, A., e Puzovic, M. (2018). Energy and power aware job scheduling and resource management: Global survey—initial analysis. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops*, pages 685–693. IEEE.

Martínez, M. A., Fraguela, B. B., e Cabaleiro, J. C. (2022). A highly optimized skeleton for unbalanced and deep divide-and-conquer algorithms on multi-core clusters. *The Journal of Supercomputing*, pages 10434–10454.

McCool, M., Reinders, J., e Robison, A. (2012). *Structured parallel programming: patterns for efficient computation*. Morgan Kaufmann.

Meade, A., Deeptimahanti, D. K., Johnston, M., Buckley, J., e Collins, J. (2013). Data decomposition for code parallelization in practice: what do the experts need? In *10th International Conference on High Performance Computing and Communications*, pages 754–761. IEEE.

Memeti, S. e Pllana, S. (2018). PAPA: a parallel programming assistant powered by ibm watson cognitive computing technology. *Journal of computational science*, 26:275–284.

Miller, J., Wienke, S., Schlottke-Lakemper, M., Meinke, M., e Müller, M. S. (2018). Applicability of the software cost model COCOMO II to HPC projects. *International Journal of Computational Science and Engineering*, 17(3):283–296.

Pacheco, P. S. (2011). *Introduction to Parallel Programming*. Morgan Kaufmann, Burlington, MA, USA.

Peccerillo, B. e Bartolini, S. (2022). Flexible task-dag management in phast library: Data-parallel tasks and orchestration support for heterogeneous systems. *Concurrency and Computation: Practice and Experience*, 34(2):1–20.

Plale, B. A., Malik, T., e Pouchard, L. C. (2021). Reproducibility practice in high-performance computing: Community survey results. *Computing in Science & Engineering*, 23(5):55–60.

Rodriguez-Canal, G., Torres, Y., Andújar, F. J., e Gonzalez-Escribano, A. (2021). Efficient heterogeneous programming with fpgas using the controller model. *The Journal of Supercomputing*, 77(12):13995–14010.

Schlagkamp, S. e Renker, J. (2015). Acceptance of waiting times in high performance computing. In *International Conference on Human-Computer Interaction*, pages 709–714. Springer.

Szafron, D. e Schaeffer, J. (1996). An experiment to measure the usability of parallel programming systems. *Concurrency: Practice and Experience*, 8(2):147–166.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., e Wesslén, A. (2012). *Experimentation in software engineering*. Springer.