

João Vicente Souto

**Desenvolvimento de um *Exokernel* para o
Processador *Manycore* MPPA-256**

Florianópolis

2018

João Vicente Souto

Desenvolvimento de um *Exokernel* para o Processador *Manycore* MPPA-256

Proposta de monografia submetida ao Programa de Graduação em Ciência da Computação para a obtenção do Grau de Bacharel.

Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Ciência da Computação

Orientador: Márcio Bastos Castro
Coorientador: Pedro Henrique Penna

Florianópolis

2018


FOLHA DE APROVAÇÃO DE PROPOSTA DE TCC

Acadêmico(s)	João Vicente Souto
Título do trabalho (subtítulo)	Desenvolvimento de um <i>Exokernel</i> para o Processador <i>Manycore</i> MPPA-256
Curso	Ciência da Computação /INE/UFSC
Área de Concentração	Sistemas de Computação

Instruções para preenchimento pelo ORIENTADOR DO TRABALHO:

- - Para cada critério avaliado, assinale um X na coluna SIM apenas se considerado aprovado. Caso contrário, indique as alterações necessárias na coluna Observação.

Critérios	Aprovado				Observação
	Sim	Parcial	Não	Não se aplica	
1. O trabalho é adequado para um TCC no CCO/SIN (relevância / abrangência)?	✓				
2. O título do trabalho é adequado?	✓				
3. O tema de pesquisa está claramente descrito?	✓				
4. O problema/hipóteses de pesquisa do trabalho está claramente identificado?	✓				
5. A relevância da pesquisa é justificada?	✓				
6. Os objetivos descrevem completa e claramente o que se pretende alcançar neste trabalho?	✓				
7. É definido o método a ser adotado no trabalho? O método condiz com os objetivos e é adequado para um TCC?	✓				
8. Foi definido um cronograma coerente com o método definido (indicando todas as atividades) e com as datas das entregas (p.ex. Projeto I, II, Defesa)?	✓				
9. Foram identificados custos relativos à execução deste trabalho (se houver)? Haverá financiamento para estes custos?	✓				
10. Foram identificados todos os envolvidos neste trabalho?	✓				
11. As formas de comunicação foram definidas (ex: horários para orientação)?	✓				
12. Riscos potenciais que podem causar desvios do plano foram identificados?	✓				
13. Caso o TCC envolva a produção de um software ou outro tipo de produto e seja desenvolvido também como uma atividade realizada numa empresa ou laboratório, consta da proposta uma declaração (Anexo 3) de ciência e concordância com a entrega do código fonte e/ou documentação produzidos?				✓	

Avaliação	<input checked="" type="checkbox"/> Aprovado <input type="checkbox"/> Não Aprovado		
Professor Responsável	Márcio Bastos Castro	12/11/18	

Resumo

As decisões de projeto no desenvolvimento de qualquer aplicação influenciam diretamente seu desempenho e consumo de energia. Sistemas operacionais não são diferentes, incorporando um conjunto de decisões de projeto que, em muitos casos, são herdadas e permanecem inalteradas mesmo que o *hardware* e o *software* tenham evoluído com o passar dos anos. Pelo fato dos sistemas operacionais formarem a base de quase todas as pilhas de *software*, suas inadequações possuem um impacto generalizado nos sistemas atuais (HUNT; LARUS, 2007).

O objetivo deste trabalho é propor um *kernel* minimalista, denominado *exokernel*, que abstraia a alocação, manipulação e segurança dos recursos de comunicação do processador *manycore* Kalray MPPA-256. A motivação para o seu desenvolvimento está relacionado às abstrações de alto nível para comunicação entre processos denominadas *Inter-Process Communication* (IPC) e disponibilizadas para o processador MPPA-256, onde, apesar de facilitar o desenvolvimento de aplicações paralelas e distribuídas, elas dependem de uma pilha de *software* excessiva. Como alternativa, é possível recorrer as bibliotecas de baixo nível que são utilizadas por tais abstrações, porém, características como alocação, sincronização, multiplexação e segurança dos recursos necessários, aumentam a complexidade do desenvolvimento e diminuem a portabilidade das aplicações. Neste ponto, o *exokernel* fornecerá as mesmas abstrações IPC disponíveis com uma comunicação entre processos eficiente, agregado a uma quantidade menor de camadas de *software* intermediárias e conservando a facilidade e portabilidade existentes.

Diversos experimentos serão efetuados com as abstrações disponibilizadas pela Kalray e implementadas pelo *exokernel* proposto, com o propósito de comparar as duas soluções. A análise ocorrerá sobre métricas relevantes para o projeto, buscando avaliar, principalmente, o aumento do desempenho e eficiência energética do processador MPPA-256.

Palavras-chaves: *manycores*. MPPA-256. *exokernel*. *lightweight*. *comunicação*. *IPC*.

Sumário

1	INTRODUÇÃO	7
2	OBJETIVOS	9
3	FUNDAMENTAÇÃO TEÓRICA	11
3.1	Processador <i>manycore</i> MPPA-256	11
3.2	Exokernel	12
4	MÉTODO DE PESQUISA	15
5	PLANEJAMENTO	17
5.1	Cronograma	17
5.2	Recursos Humanos	17
5.3	Custos	18
5.4	Comunicação	18
5.5	Riscos	20
	REFERÊNCIAS	21

1 Introdução

Nas últimas décadas, a medida que os limites de melhoria para um único núcleo de processamento começaram a ser alcançados, o desenvolvimento de arquiteturas paralelas, possuindo processadores com centenas e até milhares de núcleos, supriu a crescente necessidade de poder de processamento das plataformas de *High-Performance Computing* (HPC). No entanto, o aumento da capacidade computacional foi acompanhada pelo aumento no consumo de energia, tornando-se a maior desvantagem para plataformas HPC.

Em uma era em que os computadores alcançaram o Petaflop, um relatório feito pelo Departamento de Defesa do Governo dos Estados Unidos (DARPA/IPTO) (KOGGE et al., 2008), mostra o aumento da preocupação com o consumo de energia à medida que se torna mais importante ter uma relação equilibrada entre poder de processamento e eficiência energética a fim de atingirmos o *Exascale*. Em virtude disso, pesquisas científicas e industriais voltaram sua atenção para a aplicabilidade de arquiteturas *manycore* de baixo consumo em problemas que demandam alto desempenho (CASTRO et al., 2014; CASTRO et al., 2016).

Essa nova classe de arquiteturas *manycore* de baixo consumo energético apresentam diversas características distintas das arquiteturas *multicore* e *manycore* existentes, tais como: (i) centenas ou até mesmo milhares de núcleos de processamento simplificados em um único *chip*; (ii) comunicação entre núcleos baseada em troca de mensagens usando uma *Network on Chip* (NoC); e (iii) possuem subsistemas de memória restritiva. Embora a relação entre poder de processamento e eficiência energética seja vantajosa, tais características dificultam o desenvolvimento de aplicações paralelas e distribuídas para essas arquiteturas (SOUZA et al., 2016; CASTRO et al., 2016; PENNA et al., 2018).

Além dos aspectos de *hardware*, aspectos de *software* também influenciam no desenvolvimento e, principalmente, no desempenho de aplicações para qualquer arquitetura. Decisões de projeto, nível de abstração utilizado e sistema operacional escolhido, por exemplo, podem causar uma sobrecarga indesejada na execução de uma determinada aplicação (APPEL; LI, 1991; CAO; FELTEN; LI, 1994; HARTY; CHERITON, 1992; KRUEGER et al., 1993; STONEBRAKER, 1981; THEKKATH; LEVY, 1994; HUNT; LARUS, 2007). Além disso, mesmo que a aplicação utilize adequadamente todos os recursos disponíveis, ela deixará de explorar possíveis otimizações do seu domínio por não possuir acesso direto ao recursos do *hardware*.

Como alternativa, muitas vezes existe a opção de utilizar bibliotecas de baixo nível que permitem a elaboração personalizada do ambiente de execução, otimizando ao máximo o seu desempenho. Contudo, ao optar por se aproximar de uma plataforma específica, a portabilidade e complexidade do desenvolvimento se tornam um fator crítico ao projeto.

Neste contexto, o problema de gerenciamento e proteção dos recursos de *hardware* associado a um nível de abstração que não comprometa o desempenho de aplicações para processadores *manycore* de baixo consumo energético permanece em discussão. Uma alternativa viável é o desenvolvimento de um *exokernel* que concretize, através do uso de bibliotecas de baixo nível, uma camada de abstração de *hardware* consistente e minimalista denominada *Hardware Abstraction Layer* (HAL).

2 Objetivos

Atualmente a Kalray disponibiliza dois ambientes de execução que proporcionam facilidades no desenvolvimento de aplicações paralelas e distribuídas para o processador MPPA-256. Porém, tais ambientes inferem uma sobrecarga no desempenho das aplicações e forçam o desenvolvedor a lidar com as decisões de projeto definidas pelos projetistas da Kalray.

O presente trabalho tem como objetivo propor um ambiente de execução alternativo que equilibre flexibilidade, funcionalidade, desempenho e eficiência energética para o processador MPPA-256. Desta forma, ao utilizar camadas mais próximas do *hardware* e exportar e proteger os recursos físicos através de uma camada de abstração, o *exokernel* proposto proporcionará maior liberdade ao desenvolver em explorar otimizações para o domínio de aplicações de HPC e, ao mesmo tempo, não aumentará a complexidade do projeto por não trabalhar diretamente com um *hardware* específico.

O *exokernel* deverá, dada a utilização das primitivas IPC para comunicação e sincronização entre processos em *clusters* distintos, realizar a alocação, proteção e multiplexação dos recursos necessários, além da configuração e envio/recebimento de dados. Ao prover a comunicação, o *exokernel* deverá implementar uma política de troca de mensagens assíncronas abaixo de sua interface, a fim de providenciar uma comunicação eficiente e não provocar perda de desempenho ou eficiência energética em relação aos ambientes de execução existentes. Para isso, serão utilizadas bibliotecas de baixo nível do processador MPPA-256.

Restrições:

- Executar aplicações no processador;
- Utilizar as bibliotecas de baixo nível do processador;
- Memória limitada no *chip*, dividida entre o sistema operacional e a aplicação;
- Respeitar as datas de entregas.

Premissas:

- Processador disponível;
- Documentação disponível;
- Acesso ao processador de forma remota;

- Computador disponível;
- Disponibilidade de água, luz e energia;
- Acesso à Internet.

Marcos:

- Entrega da proposta: 12-11-2018;
- Entrega do resumo em TCC I: junho de 19;
- Primeira entrega da monografia em TCCII: outubro de 19;
- Apresentação da monografia: novembro de 19;
- Segunda entrega da monografia em TCCII (versão final): dezembro de 19.

Crerios de aceite:

- Aprovaço pela banca;
- Aprovaço do orientador;
- Conformidade com as normas definidas pela instituiço;
- Prazos cumpridos;
- Entregas cumpridas.

3 Fundamentação Teórica

Esta seção apresenta uma fundamentação teórica básica sobre o principal objeto de estudo (Seção 3.1) e conceito necessário (Seção 3.2) para a viabilidade deste projeto.

3.1 Processador *manycore* MPPA-256

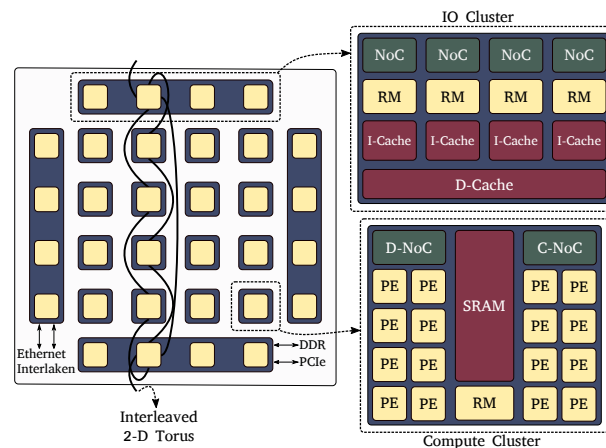
O processador MPPA-256 é um exemplo de arquitetura *manycore* de baixo consumo energético (CASTRO et al., 2013) desenvolvido pela empresa francesa Kalray. Como mostra a Figura 1, o MPPA-256 é um processador que integra 256 núcleos de propósito geral, denominados *Processing Elements* (PEs), e 32 núcleos dedicados ao sistema operacional, denominados *Resource Managers* (RMs), criados sobre uma tecnologia CMOS de 28nm. Esses núcleos são distribuídos através de 16 *clusters* de computação (*Compute Clusters*) com 16 PEs e 1 RM cada, somados a 4 *quad-cores* para os subsistemas de E/S (*IO Clusters*). Cada *cluster* possui um espaço de endereçamento privado, enquanto as comunicações ocorrem através de uma NoC dedicada a dados, denominada *Data NoC* (D-NoC), e outra dedicada a controles, denominada *Control NoC* (C-NoC).

As camadas de *software* disponibilizadas para o MPPA-256, como pode ser visto na Figura 2, possuem (i) suporte a programação paralela e distribuída, como *POSIX*, *OpenMP* etc; (ii) primitivas para sincronização e comunicação entre processos (IPC), tais como, *sync*, *portal* e *mailbox*; (iii) sistemas operacionais para cada tipo de *cluster*; e (iv) acesso aos recursos do *hardware* através de bibliotecas de baixo nível.

Nos *IO Clusters*, é possível executar uma variação do sistema operacional Linux, nomeada *RTEMS*, possibilitando a comunicação da máquina *host* com o dispositivo, além da comunicação com os *Compute Clusters*. Sobre outra perspectiva, nos *Compute Clusters*, o sistema operacional *NodeOS*, projetado para adequar-se aos 2-MB de memória local, é executado no RM com o objetivo de auxiliar e gerenciar a comunicação dos PEs com os demais elementos do processador. Em alternativa a esses ambientes de execução, existe a possibilidade de utilizar o MPPA-256 apenas com o *hypervisor* e o auxílio de bibliotecas de baixo nível.

A comunicação entre processos em *clusters* distintos ocorre de maneira explícita, principalmente através de três primitivas IPC. Entre elas estão, (i) *Sync*: permite a sincronização de $N:M$ processos através da C-NoC; (ii) *Mailbox*: envio de mensagens pequenas através da D-NoC; e (iii) *Portal*: envio de grandes quantidade de dados através da D-NoC. De forma semelhante aos sistemas operacionais, o desenvolvedor tem a liberdade de lidar com bibliotecas de baixo nível que são utilizadas por essas primitivas.

Figura 1 – Visão geral do MPPA-256.



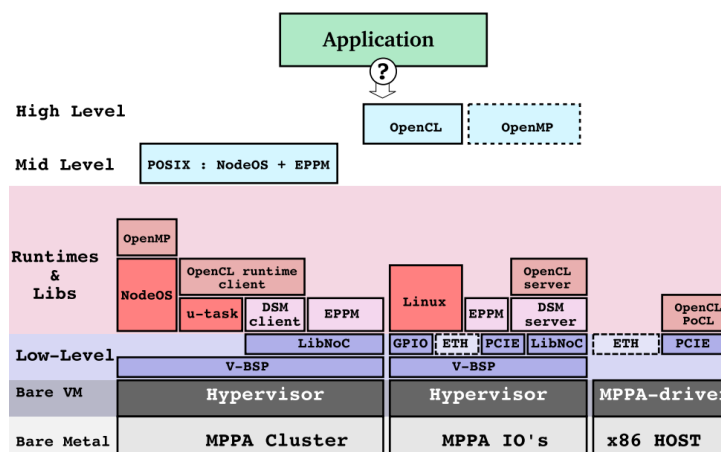
Fonte: (Penna et al. (2018))

3.2 Exokernel

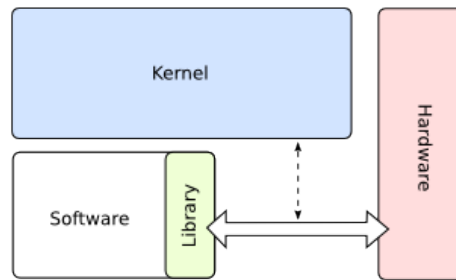
O *Exokernel* consiste em uma arquitetura desenvolvida pelo *Massachusetts Institute of Technology* (MIT) que exporta com segurança todos os recursos de *hardware* através de uma interface de baixo nível. Seu objetivo é proporcionar o gerenciamento de recursos físicos no nível da aplicação em oposição a limitação do desempenho, da flexibilidade e da funcionalidade impostas por sistemas operacionais tradicionais.

Ao contrário da abordagem comum escolhida por sistemas operacionais monolíticos, expondo os recursos de *hardware* através de abstrações de alto nível, o *Exokernel* transfere a responsabilidade de implementá-las para o desenvolver da aplicação, como pode ser visto na Figura 3. Neste ponto, o *Exokernel* também se opõe a ideia dos *microkernels*, que, ao contrário de buscar fornecer as mesmas abstrações de sistemas monolíticos com uma quantidade menor de camadas intermediárias, propõe apenas separar a função de proteção dos

Figura 2 – Ambiente de execução do processador MPPA-256.



Fonte: Documentação do processador MPPA-256

Figura 3 – Arquitetura *Exokernel*.

Fonte: (OSDev Wiki, 2018)

recursos do *hardware* da função de gerenciamento. Desta forma, ao não impor abstrações ao desenvolvedor, permite a personalização do ambiente de execução em prol do domínio da aplicação.

Por fim, a arquitetura de *software* definida pelo *Exokernel* provou ser uma alternativa viável às arquiteturas de sistemas operacionais tradicionais demonstrando que, (i) um número limitado de primitivas simplificadas permite implementá-las eficientemente; (ii) a proteção dos recursos pode ser realizada de forma eficaz com implementações de baixo nível; (iii) abstrações tradicionais de sistemas operacionais podem ser desenvolvidas em cima dessas primitivas de forma eficiente; e (iv) permite que aplicações desenvolvam abstrações de propósito geral através de pequenas modificações (ENGLER; KAASHOEK; JR., 1995).

4 Método de Pesquisa

O andamento deste projeto ocorrerá em paralelo à pesquisa de doutorado do coorientador deste projeto, Pedro Henrique Penna, e utilizará como base o protótipo do sistema operacional distribuído, denominado *multikernel*, implementando no repositório do sistema operacional Nanvix (PENNA, 2017; PENNA et al., 2017) disponibilizado em <<https://github.com/nanvix/multikernel>>.

O desenvolvimento do trabalho será feito na linguagem C e dividido em cinco módulos, os quais são, (i) *NoC Driver*: utilização da biblioteca de baixo nível da NoC para controle, manipulação e multiplexação dos *buffers* de comunicação; (ii) *Core Driver*: identificação e configuração dos elementos ativos do sistemas (*IO Clusters* e *Compute Clusters*); (iii) *Sync*: abstração para sincronização entre processos localizados em *clusters* distintos; (iv) *Mailbox*: abstração para troca de mensagens pequenas, geralmente associadas a operações de controle; e (v) *Portal*: abstração dedicada a troca de dados intensa. Os módulos *sync* e *mailbox* são dependentes dos dois primeiros módulos, assim como o *portal* é dependente dos demais.

Os módulos e testes desenvolvidos utilizarão as bibliotecas e documentação disponibilizados pela Kalray e serão executados no processador MPPA-256. A plataforma que contém o MPPA-256 está localizada em Grenoble (França) e o acesso remoto é proporcionado por uma parceria entre o *Distributed Systems Research Laboratory* (LaPeSD) e o *Laboratoire d'Informatique de Grenoble* (LIG). O orientando terá acesso ao laboratório LaPeSD, localizado na Universidade Federal de Santa Catarina (UFSC), no qual possuirá uma máquina disponível para o desenvolvimento do trabalho. Além disso, o andamento deste trabalho ocorrerá associado a bolsa de iniciação científica do orientando, contribuindo com a sua formação e na qualidade final do trabalho desenvolvido.

5 Planejamento

Este capítulo apresenta planejamento do trabalho, o qual é composto pelo cronograma de atividades, os recursos humanos necessários e os seus custos. Além disso, serão descritas as atividades de comunicação necessárias para a execução do projeto, assim como os possíveis riscos associados às atividades.

5.1 Cronograma

O planejamento deste trabalho está organizado da seguinte maneira. O desenvolvimento da solução abrange a especificação do trabalho apresentado no Capítulo 4. O primeiro resumo da monografia, relacionado ao relatório do Projeto I, desenvolverá a estrutura básica da monografia e conterá os resultados obtidos até o momento. As conclusões encerrarão o desenvolvimento da solução, levando em consideração o retorno obtido do relatório do Projeto I e iniciarão a síntese dos resultados. O rascunho relacionado ao Projeto II envolverá as atividades de elaboração da monografia e o preparo da apresentação para a defesa. A defesa apresentará o trabalho desenvolvido e os resultados alcançados. Por fim, serão realizados os ajustes e correções solicitados pelos membros da banca e a versão final da monografia será enviada ao acervo da UFSC.

5.2 Recursos Humanos

Papel	Nome
Orientador	Márcio Bastos Castro
Coorientador	Pedro Henrique Penna
Coordenador	Renato Cislighi
Membro da Banca I	A definir
Membro da Banca II	A definir
Autor	João Vicente Souto

	2018			2019											
Etapas/Meses	10	11	12	01	02	03	04	05	06	07	08	09	10	11	12
Desenvolv. da Solução															
Relatório Projeto I															
Conclusões															
Rascunho Projeto II															
Defesa															
Ajustes e Envio Final															

Tabela 1 – Cronograma de atividades do projeto.

5.3 Custos

Estimativas para Recursos Humanos					
Nome	Data Início	Data Fim	Hora/Mês	Valor/Hora	Custo Total
Autor	30/09/2018	15/12/2019	96	R\$ 5,00	R\$ 7200,00
Orientador	30/09/2018	15/12/2019	8	R\$ 65,00	R\$ 7800,00
Coorientador	30/09/2018	15/12/2019	8	R\$ 65,00	R\$ 7800,00
Coordenador	30/06/2019	15/12/2019	1	R\$ 137,00	R\$ 959,00
Banca I	30/06/2019	15/12/2019	1	R\$ 135,00	R\$ 945,00
Banca II	30/06/2019	15/12/2019	1	R\$ 135,00	R\$ 945,00
Subtotal estimativas para recursos humanos					R\$ 25.649,00

Estimativas para Recursos Não Humanos					
Descrição	Data Início	Data Fim	Quantidade	Valor Unitário	Custo Total
CDs	12/19	12/19	2	R\$ 3,00	R\$ 6,00
Impressão	12/19	12/19	3	R\$ 15,00	R\$ 45,00
Subtotal estimativas para recursos não humanos					R\$ 51,00

5.4 Comunicação

O que precisa ser comunicado	Entrega da Proposta
Emissor	João Vicente Souto
Receptor	Renato Cislighi
Comunicação	Será entregue a proposta completa do TCC
Forma de comunicação	Sistema de TCC
Frequência ou Quando	Única vez

O que precisa ser comunicado	Entrega do Relatório em TCC I
Emissor	João Vicente Souto
Receptor	Renato Cislighi
Comunicação	Será entregue a primeira parte da monografia
Forma de comunicação	Sistema de TCC
Frequência ou Quando	Única vez

O que precisa ser comunicado	Entrega da monografia completa em TCC II
Emissor	João Vicente Souto
Receptor	Renato Cislacchi
Comunicação	Será entregue a monografia completa
Forma de comunicação	Sistema de TCC
Frequência ou Quando	Única vez

O que precisa ser comunicado	Defesa do TCC
Emissor	João Vicente Souto
Receptor	Márcio Bastos Castro, Membro da Banca I, Membro da Banca II
Comunicação	Será feita a defesa do TCC aos membros da banca.
Forma de comunicação	Pessoalmente
Frequência ou Quando	Única vez

O que precisa ser comunicado	Reuniões com o Orientador
Emissor	João Vicente Souto
Receptor	Márcio Bastos Castro
Comunicação	Reuniões com o orientador para atualizações e dúvidas
Forma de comunicação	Pessoalmente
Frequência ou Quando	Quinzenalmente

O que precisa ser comunicado	Reuniões com o Coorientador
Emissor	João Vicente Souto
Receptor	Pedro Henrique Penna
Comunicação	Reuniões com o coorientador para atualizações e dúvidas
Forma de comunicação	Videoconferência
Frequência ou Quando	Semanalmente

O que precisa ser comunicado	Enviar a monografia com ajustes após a defesa
Emissor	João Vicente Souto
Receptor	Renato Cislacchi
Comunicação	Entrega da monografia com os ajustes necessários
Forma de comunicação	Sistema de TCC
Frequência ou Quando	Única vez

5.5 Riscos

Nome	Probabilidade	Impacto	Exposição	Estratégia	Ações de Prevenção
Sobrecarga da graduação	Média	Alto	Alto	Mitigar	Cumprir o cronograma proposto.
O acesso à máquina está offline	Média	Médio	Médio	Mitigar	-
Outra pessoa está utilizando a máquina por um período longo	Média	Médio	Média	Aceitar	-
Orientador está de viagem	Média	Baixo	Médio	Mitigar	-
Orientador fica doente	Média	Baixo	Baixa	Mitigar	-
Greve de ônibus	Média	Baixo	Baixa	Mitigar	Manter a manutenção do carro em dia.
O aluno ficou doente	Média	Médio	Média	Aceitar	Fazer exercícios, comer bem e ir ao médico ocasionalmente.
Computador não está mais funcionando	Baixa	Médio	Média	Mitigar	Tomar cuidado ao manusear o computador e desligá-lo adequadamente.

Referências

- APPEL, A. W.; LI, K. Virtual memory primitives for user programs. In: *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY, USA: ACM, 1991. (ASPLOS IV), p. 96–107. ISBN 0-89791-380-9. Disponível em: <<http://doi.acm.org/10.1145/106972.106984>>.
- CAO, P.; FELTEN, E. W.; LI, K. Implementation and performance of application-controlled file caching. In: *Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 1994. (OSDI '94). Disponível em: <<http://dl.acm.org/citation.cfm?id=1267638.1267651>>.
- CASTRO, M. et al. Energy efficient seismic wave propagation simulation on a low-power manycore processor. In: *International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. Paris, France: IEEE Computer Society, 2014. p. 57–64.
- CASTRO, M. et al. Seismic wave propagation simulations on low-power and performance-centric manycores. *Parallel Computing*, v. 54, p. 108–120, 2016. ISSN 01678191. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167819116000417>>.
- CASTRO, M. et al. Analysis of computing and energy performance of multicore, NUMA, and manycore platforms for an irregular application. In: *Workshop on Irregular Applications: Architectures & Algorithms (IA³)*. Denver, EUA: ACM, 2013. p. 5:1–5:8.
- ENGLER, D.; KAASHOEK, F.; JR., J. O. Exokernel: An operating system architecture for application-level resource management. In: *SOSP '95 Proceedings of the 15th ACM Symposium on Operating Systems Principles*. ACM, 1995. (SOSP '95), p. 251–266. ISBN 0-89791-715-4. Disponível em: <<https://dl.acm.org/citation.cfm?doid=224057.224076>>.
- HARTY, K.; CHERITON, D. R. Application-controlled physical memory using external page-cache management. In: *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY, USA: ACM, 1992. (ASPLOS V), p. 187–197. ISBN 0-89791-534-8. Disponível em: <<http://doi.acm.org/10.1145/143365.143511>>.
- HUNT, G. C.; LARUS, J. R. Singularity: Rethinking the software stack. v. 41, n. 2, p. 37–49, 2007. ISSN 0163-5980. Disponível em: <<http://doi.acm.org/10.1145/1243418.1243424>>.
- KOGGE, P. et al. Exascale computing study: Technology challenges in achieving exascale systems. *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Techinal Representative*, v. 15, 01 2008.
- KRUEGER, K. et al. Tools for the development of application-specific virtual memory management. In: *Proceedings of the Eighth Annual Conference on Object-oriented Programming Systems, Languages, and Applications*. New York, NY, USA: ACM, 1993. (OOPSLA '93), p. 48–64. ISBN 0-89791-587-9. Disponível em: <<http://doi.acm.org/10.1145/165854.165867>>.
- OSDev Wiki. *Exokernel Description*. 2018. Accessed = 12-11-2018. Disponível em: <<https://wiki.osdev.org/Exokernel>>.

PENNA, P. H. *The Nanvix Operating System*. Belo Horizonte, 2017. 20 p. Disponível em: <<https://hal.archives-ouvertes.fr/hal-01495741>>.

PENNA, P. H. et al. Using the Nanvix Operating System in Undergraduate Operating System Courses. In: *2017 VII Brazilian Symposium on Computing Systems Engineering*. Curitiba, Brazil: IEEE, 2017. (SBESC '17), p. 193–198. ISBN 978-1-5386-3590-2. Disponível em: <<http://ieeexplore.ieee.org/document/8116579/>>.

PENNA, P. H. et al. An os service for transparent remote memory accesses in noc-based lightweight manycores. Poster. 2018.

SOUZA, M. A. et al. CAP Bench: A Benchmark Suite for Performance and Energy Evaluation of Low-power Many-core Processors. *Concurrency and Computation: Practice and Experience*, 2016. ISSN 1532-0634.

STONEBRAKER, M. Operating system support for database management. *Commun. ACM*, ACM, New York, NY, USA, v. 24, n. 7, p. 412–418, jul. 1981. ISSN 0001-0782. Disponível em: <<http://doi.acm.org/10.1145/358699.358703>>.

THEKKATH, C.; LEVY, H. Hardware and software support for efficient exception handling. v. 29, p. 110–119, 01 1994.