Port do Sistema Operacional Nanvix para Arquitetura RISC-V Plataforma PULP

Lucas S. Oliveira, Thiago H. Nogueira, Henrique C. Freitas *

[1]Departamento de Ciência da Computação – Pontifícia Universidade Católica de Minas Gerais (PUC Minas) Belo Horizonte – MG – Brazil

{lucas.oliveira.1201561, thiago.nogueira}@sga.pucminas.br, cota@pucminas.br

Abstract. *Operating Systems are developed in order to have the best optimization and operation for different processor architectures. Their design is a careful process, as several emerging technologies are constantly being disseminated. In this work, we present the development of the port of the Nanvix operating system to a RISC-V architecture, which will be emulated by a tool called PULP SDK. To verify the effectiveness of the port, automated tests were performed, from kernel to user level.*

Resumo. *Sistemas operacionais são desenvolvidos com o intuito de possuir a melhor otimização e funcionamento para diversas arquiteturas de processadores. A implementação é um processo delicado, já que diversas tecnologias diferentes emergem constantemente. Neste trabalho, é apresentado o desenvolvimento do port do sistema operacional Nanvix para uma arquitetura RISC-V, que vai ser emulada a partir de uma ferramenta denominada PULP SDK. Para verificar o funcionamento do port, testes automatizados foram realizados, desde o nível de kernel ao de usuário.*

1. Introdução

O desenvolvimento de sistemas operacionais (SOs) é influenciado por uma série de fatores, entre eles, o contexto onde será utilizado. Nesse sentido, sistemas embarcados demandam SOs eficientes tanto do ponto de vista de desempenho quanto de energia. A plataforma *Parallel Ultra Low Power* (PULP) [PULP Team 2022] foi desenvolvida com base na arquitetura de conjunto de instruções RISC-V, e possui um número ajustável de processadores de acordo com o que for configurado. O *port* de um sistema operacional para esta plataforma é fundamental para gerência de recursos.

O SO Nanvix foi desenvolvido com foco em processadores *lightweight manycore*. Em linhas gerais, é um SO que consegue ser executado em plataformas com poucos recursos. Há o *port* para arquiteturas MPPA-256, OpenRISC, RISC-V, x86, OpTiMSoC e ARM64. No entanto, existem características que são específicas para cada arquitetura e plataforma. Em especial, a plataforma PULP, que embora seja baseada em RISC-V não é compatível com nenhuma das arquiteturas previamente citadas. Isso faz com que o principal problema abordado no artigo seja a portabilidade do Nanvix para o PULP.

Um software de código aberto chamado PULP-SDK [PULP Platform 2020] foi construído com o intuito de escolher as melhores ferramentas para emulação e

*Lucas e Thiago são alunos de iniciação científica.

virtualização de processadores RISC-V. Bruschi et al. [Bruschi et al. 2021] apresentaram uma proposta de software chamada GVSoC, que é totalmente modificável, rápido e preciso na hora de virtualizar esse processador. Por conta disso, foi incluído no PULP e foi utilizado neste artigo.

Nesse sentido, o objetivo principal deste artigo é apresentar a abordagem de *port* para o PULP com discussões de etapas e verificação via ambiente de integração contínua chamada Jenkins [Server 2022]. Este é um trabalho de pesquisa que contribui para um projeto desenvolvido na Universidade Federal do Rio Grande do Norte, que objetiva projetar um microssatélite para monitorar a floresta Amazônica [Projeto Cevero 2021]. Com isso, o uso eficiente dos recursos, sendo o baixo consumo energético e alta performance, são essenciais.

Este trabalho está organizado da seguinte forma. A Seção 2 discute os trabalhos relacionados. A Seção 3 é um *background* sobre o sistema operacional e o processador utilizado na pesquisa. A Seção 4 apresenta a abordagem do estudo utilizada para o *port* do Nanvix. A Seção 5 refere-se a uma discussão sobre o *port*. E por fim, a Seção 6 com as conclusões seguidas de trabalhos futuros.

2. Trabalhos Correlatos

Esta seção apresenta alguns artigos voltados para o *port* de sistemas operacionais, incluindo o próprio Nanvix para outras arquiteturas.

O artigo de Drozdov et al. [Drozdov et al. 2015] apresenta um modelo de arquitetura e princípios utilizados na construção do Milandr-OS. Desenvolvido para clusters de multi-processadores com comunicação ponta-a-ponta usando sinais digitais. A criação deste se deu no momento em que programadores tinham problemas constantes em reescrever algoritmos entre máquinas diferentes, por conta da diferença de arquitetura entre elas. Esse sistema provê algumas interfaces padronizadas que independente do hardware que for executado, os algoritmos previamente escritos poderão ser reutilizados.

Conforme proposto por Legaspi et al. [Legaspi et al. 2015], um sistema Linux foi portado para um computador embarcado para que pudesse ser usado aplicações que se utilizassem de todos os sensores disponíveis na placa. O sistema era originalmente proposto para alguns modelos de processador ARM, mas não era totalmente funcional com os sensores que estavam presentes nesse FPGA e o artigo mencionado conseguiu que todos os sensores da placa funcionassem.

Diversos trabalhos no Nanvix já foram feitos para que outras arquiteturas fossem suportadas. Pedro Penna apresentou o uso já funcional de um port do Nanvix para outra arquitetura chamada MPPA [Penna et al. 2021]. Foi usado um processador many-core de 288 núcleos que mostrou um ganho significativo de desempenho quando usado um modelo IKC Facility (proposto no artigo em referência), quando comparado com o modelo mailbox (modelo até então vigente).

Pedro Penna apresenta um modelo de construção do sistema em sua tese de doutorado [Penna 2021] com várias proposições de modelos de software construídos para atender com a maior eficiência ao modelo *lightweight manycores*, em que um processador possui núcleos leves (eficientes em energia) e com especialidades diferentes. O sistema operacional pode se aproveitar disso, dividindo o trabalho para os núcleos que

consigam executar aquela instrução com maior performance e menor gasto energético.

O Nanvix está sendo desenvolvido para suportar programas com múltiplas *threads* em ambientes com múltiplos núcleos [Souza Filho 2022]. Reinaldo Souza Filho implementou uma biblioteca de paralelismo OpenMP dentro do sistema. Com isso, tornou-se possível o uso de múltiplos núcelos por uma aplicação paralela visando ganho de desempenho para arquitetura RISC-V.

A soma dos esforços de vários contribuidores objetiva usar o Nanvix dentro de um microssatélite baseado na plataforma PULP, arquitetura RISC-V, que será usado para fotografar a floresta Amazônica [Projeto Cevero 2021]. Em relação aos correlatos discutidos nesta seção, a principal contribuição deste artigo está no *port* do SO Nanvix para a plataforma PULP.

3. Background

3.1. Nanvix

Nanvix é um sistema operacional com origem em uma abordagem educacional [Penna et al. 2017] e que se baseia na estrutura do Unix System V. Embora o *design* seja simples, o Nanvix evoluiu para uma versão distribuída com compatibilidade com a arquitetura *lightweight manycore* MPPA-256.

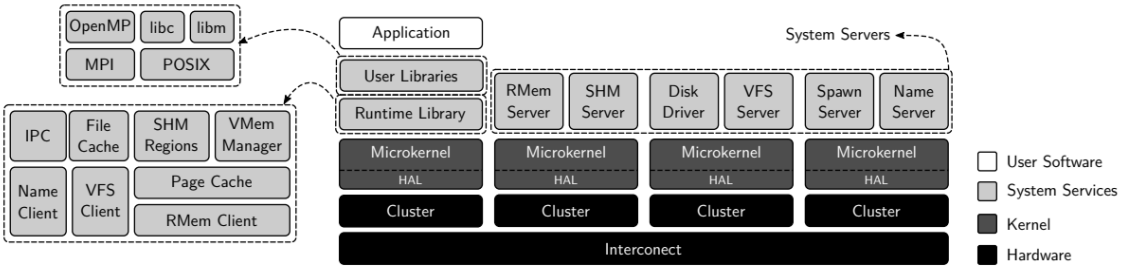

Figura 1. Arquitetura do Nanvix [Penna 2021]

A Figura 1 mostra a organização da arquitetura do Nanvix. Sua estrutura é baseada em um layout de quatro camadas: (i) Nível de Hardware; (ii) Kernel; (iii) Serviços do Sistema; (iv) Software para Usuário.

Na camada (iii), a HAL é responsável por possibilitar a portabilidade do Nanvix para diversos processadores *lightweight manycores*. Enquanto o *Microkernel* oferece recursos para o sistema com a utilização de apenas um cluster, além de executar em *privileged mode* e possuir um nível de abstração mínimo do SO. É possível obter mais informações sobre o Nanvix nos estudos realizados por Pedro Penna [Penna 2021].

3.2. RISC-V

O RISC-V [Organization 2022] é um conjunto de instruções (*Instruction Set Architecture* - ISA) baseado em princípios de RISC (*Reduced Instruction Set Computing*). Diferentes de outros designs de ISA, RISC-V é de código aberto, o que possibilita o uso dessa arquitetura pela comunidade livre de *royalties*.

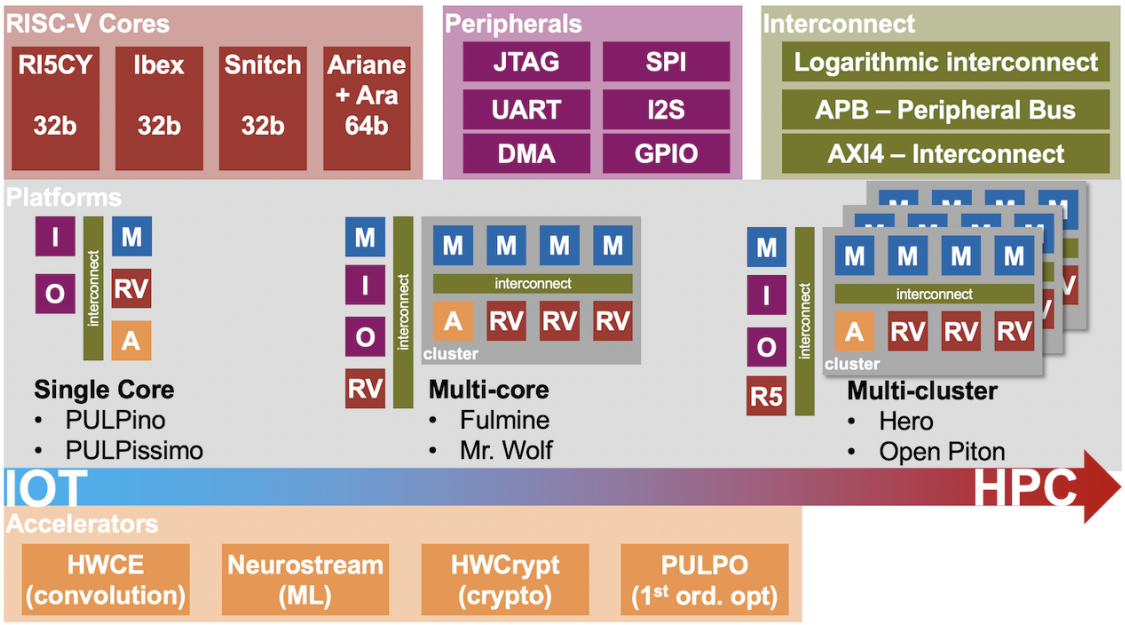

Figura 2. Pipeline do RISC-V
[PULP Team 2022]

Projetos como o PULP [PULP Team 2022] e também algumas empresas, estão se interessando mais pelo RISC-V. É o caso da Nvidia [RISC-V Organization 2016] que está utilizando esses chips dentro das placas de vídeos da nova linha Ampere, e a Western Digital [RISC-V Organization 2022], em seus dispositivos de armazenamento. Por conta de sua versatilidade em resolver uma grande gama de problemas diferentes, a adoção dessa ISA tem crescido consideravelmente nos últimos anos [RISC-V Organization 2021a]. A projeção é que até 2025, essa arquitetura já movimente em torno de um bilhão de dólares [RISC-V Organization 2021b] e com isso irá apenas crescer e se tornar mais simples e acessível.

4. Abordagem do estudo

Nessa seção, serão abordados os passos para reproduzir o processo de suportar uma nova arquitetura dentro do NanvixOS. Começando pela explicação das ferramentas usadas e uma forma de replicar os procedimentos do suporte dessa arquitetura nesse sistema operacional.

4.1. Ferramentas

4.1.1. QEMU

O qemu [QEMU 2022] é um emulador e um virtualizador de arquiteturas genérico e de código aberto. Esse programa possui a capacidade de executar em *userspace*, ou seja, executar instruções de uma arquitetura de processador em outro processador sem a necessidade de uso de virtualização.

A escolha dessa ferramenta foi essencial no contexto do Nanvix, pois o RISC-V ainda não é um hardware disseminado para que todos possam testar o sistema em hard-

ware real. Com poucas pessoas possuindo um RISC-V em mãos, virtualizá-lo permite que todos possam desenvolver para o Nanvix seja de qual arquitetura possuírem.

4.1.2. PULP SDK

O PULP SDK [PULP Platform 2020] é uma ferramenta que pretende ser mais completa e específica para o RISC-V do que o qemu. Conseguindo emular múltiplas *harts* (núcleos de um RISC-V) com bastante fidelidade a um hardware real.

Dentro dele, existe uma ferramenta chamada GVSoC [Bruschi et al. 2021], que foi projetada especificamente para emular instruções de RISC-V, tanto quanto seu comportamento. Essa ferramenta está em desenvolvimento e funciona em uma versão específica do GCC (GNU *Compiler Collection* [Projeto GNU 2022]).

4.2. Port do Nanvix para RISC-V PULP

O problema que existia no sistema era o uso de uma versão do GCC mais atual que implementa pseudo instruções reduzidas que não são suportadas pelo PULP SDK.

Para começar, será necessário algum sistema operacional compatível para o teste. Seja alguma distro baseada no Debian, como o Ubuntu, ou um Mac, este que precisará de algumas adaptações para funcionar corretamente (não há suporte no momento para o Windows).

Para compilar o sistema é necessário usar o gcc com a flag TARGET definida para "qemu-riscv", isso configura o makefile para compilar corretamente a HAL (*Hardware Abstraction Layer*) e a barelib. É necessário ter um *cross-compiler* para RISC-V instalado corretamente no computador e os pacotes do qemu para emulação de um processador RISC-V. Toda essa informação está disponível na documentação oficial do Nanvix [NanvixOS 2022].

Com todas as ferramentas em mãos, o trabalho é fazer um *backport* da versão do compilador gcc que estava sendo usado no projeto (versão 1.10 para 1.9.1). As versões diferem na forma de escrever as pseudo instruções. Para utilizar essa ferramenta é necessário realizar o *cross-compiler* da sua plataforma para RISC-V e definir a variável de ambiente "PULP_RISCV_GCC_TOOLCHAIN" com o caminho para esse programa. Com isso, será possível executar o PULP e testar as aplicações que já estão disponíveis dentro do repositório dele para verificar se a emulação está funcionando corretamente. Sempre é importante ressaltar que é necessário verificar o funcionamento da emulação de múltiplas *harts* ou se está rodando todo o código de forma sequencial. Toda essa informação está disponível na documentação oficial do PULP SDK [PULP Platform 2020].

Para que fosse possível fazer esse *backport*, foi necessário encontrar a documentação do RISC-V em uma versão atual e em uma versão mais antiga, antes de suportar versões reduzidas de suas instruções. Dentro dos arquivos do Nanvix, três deles possuem código de máquina dentro do código em C (usando a keyword __asm__ do gcc) e que precisariam ser desatualizados. Encontrando todas as referências às instruções reduzidas dentro do repositório, é necessário alterá-las para uma instrução completa e, com

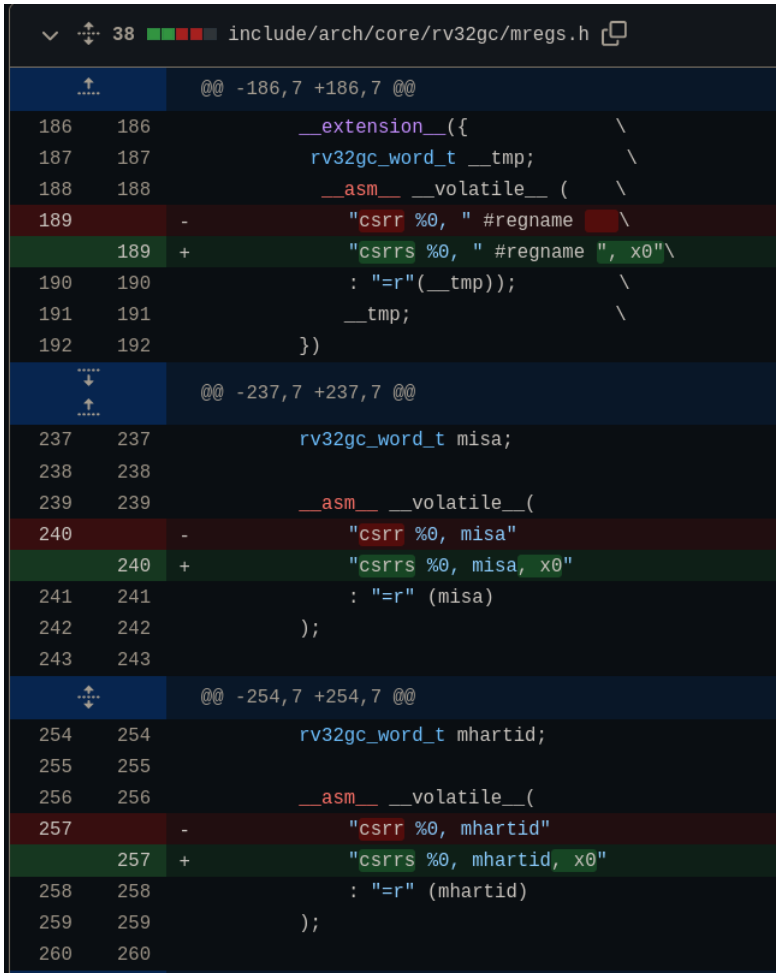

Figura 3. Commit com as intruções modificadas [Oliveira and Nogueira 2021]

isso, o sistema já estará suportando o PULP SDK. É possível testar seu funcionamento compilando o Nanvix usando o PULP.

5. Discussão

Após todas as etapas realizadas, foi necessário verificar o funcionamento das modificações realizadas no código. Para isso foi realizada uma bateria de testes, que pode ser organizada em duas etapas: (i) A primeira consiste em testes que são realizados manualmente por outros contribuidores, ao enviar o *commit* para o repositório da organização; (ii) A segunda consiste na realização de testes na plataforma Jenkins, que faz uma compilação do projeto desde o nível de *kernel* ao nível de usuário.

Os testes foram realizados inicialmente via Jenkins, mas com contribuição fundamental do contribuidor João Souto da Universidade Federal de Santa Catarina (UFSC), que realizou testes manualmente em sua própria máquina para comprovar que as linhas modificadas estavam funcionando conforme o esperado. Com essa aprovação, as linhas modificadas foram inseridas no código oficial do sistema disponível em [Oliveira and Nogueira 2021], e o *backport* foi realizado.

6. Conclusão

Este trabalho teve como foco apresentar um passo a passo de como realizar um *port* funcional para uma arquitetura (RISC-V) no sistema operacional NanvixOS. As instruções modificadas podem ser usadas como uma forma de aprendizado e de base para novos pesquisadores. O *port* foi adicionado ao sistema e está disponível para uso no repositório do GitHub.

Pensando em trabalhos futuros, poderia-se usar a ferramenta de *benchmark* que a organização do Nanvix possui para verificar a eficiência do *port* em diferentes arquiteturas e, com isso, pensar em otimizações para o código do sistema. Além disso, com base nos testes, seria possível verificar em qual cenário determinada arquitetura se destaca. Além disso, este *port* ajudará a viabilizar a execução real do Nanvix na plataforma PULP no projeto CEVERO da UFRN.

7. Agradecimentos

Os autores agradecem ao Fundo de Incentivo à Pesquisa (FIP) da PUC Minas, e aos pesquisadores Pedro Penna (Microsoft Research), João Souto (UFSC) e Reinaldo Souza Filho (UFRN) pelo suporte ao longo da pesquisa.

Referências

Bruschi, N., Haugou, G., Tagliavini, G., Conti, F., Benini, L., and Rossi, D. (2021). Gvsoc: A highly configurable, fast and accurate full-platform simulator for risc-v based iot processors. In *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pages 409–416.

Drozdov, A., Fonin, Y. N., and Kanaev, V. V. (2015). Real-time operating system for dsp clusters. *2015 International Conference on Engineering and Telecommunication (EnT)*, pages 50–53.

Legaspi, P. A. D., Khan, K. K., Santiago, K. D., Sayson, D. F., Aquino, H. O., Densing, C. V. J., Hizon, J. R. E., and Alarcon, L. P. (2015). Porting an operating system on an arm-based sensor platform. In *TENCON 2015 - 2015 IEEE Region 10 Conference*, pages 1–3.

NanvixOS (2022). Educational spinoff of nanvix. `https://github.com/nanvix/nanvix`.

Oliveira, L. S. and Nogueira, T. H. (2021). Commit contendo o *downgrade* das intruções do nanvix. `https://github.com/nanvix/hal/commit/edde1ac203d7613f81afa42aa8613c73270e7add`.

Organization, R.-V. (2022). Risc-v organization site. `https://riscv.org/`.

Penna, P. H. (2021). *Nanvix : A Distributed Operating System for Lightweight Manycore Processors*. Theses, Université Grenoble Alpes [2020-....] ; Pontifícia universidade católica de Minas Gerais (Brasil).

Penna, P. H., Souto, J. V., Uller, J. F., Castro, M., Freitas, H., and Méhaut, J.-F. (2021). Inter-kernel communication facility of a distributed operating system for noc-based lightweight manycores. *Journal of Parallel and Distributed Computing*, 154:1–15.

Penna, P. H. d. M. M., Castro, M. B., Freitas, H. C. d., Méhaut, J.-F., and Caram, J. (2017). Using the nanvix operating system in undergraduate operating system courses. In *2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 193–198.

Projeto Cevero (2021). Chip multi-processor for very energy-efficient aerospace missions. https://github.com/cevero.

Projeto GNU (2022). Gcc, the gnu compiler collection. https://gcc.gnu.org/.

PULP Platform (2020). https://github.com/pulp-platform/pulp-sdk.

PULP Team (2022). PULP Plataform organization. https://pulp-platform.org/.

QEMU (2022). Qemu: generic machine emulator. https://www.qemu.org/.

RISC-V Organization (2016). Nvidia risc-v story. https://riscv.org/wp-content/uploads/2016/07/Tue1100_Nvidia_RISCV_Story_V2.pdf.

RISC-V Organization (2021a). Risc-v celebrates incredible year of growth and progress, ratifying multiple technical specifications, launching new education programs, and accelerating broad industry adoption. https://riscv.org/announcements/2021/12/risc-v-celebrates-incredible-year-of-growth-and-progress_ratifying-multiple-technical-specifications-launching-new_education-programs-and-accelerating-broad-industry-adoption/.

RISC-V Organization (2021b). Risc-v growth and successes in technology and industry : embedded world 2021. https://riscv.org/blog/2021/03/risc-v-growth-and-successes-in-technology-and-industry-em_bedded-world-2021/.

RISC-V Organization (2022). Liberando o poder dos dados através das iniciativas risc-v. https://www.westerndigital.com/pt-br/solutions/risc-v.

Server, J. (2022). Automation Server. https://www.jenkins.io/.

Souza Filho, R. A. d. (2022). Implementação de openmp para o sistema operacional nanvix em risc-v. Master's thesis, Universidade Federal do Rio Grande do Norte, Rio Grande do Norte.