Impacto de Abstrações IPC sobre o Processador MPPA-256

João Vicente Souto[1], Emmanuel Podestá Jr.[1], Pedro H. Penna[2], Márcio Castro[1]

[1] Laboratório de Pesquisa em Sistemas Distribuídos (LaPeSD)
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, Brasil

[2]Université Grenoble Alpes (UGA) – Grenoble, França

joao.vicente.souto@grad.ufsc.br, emmanuel.podesta@posgrad.ufsc.br,
pedro.penna@univ-grenoble-alpes.fr, marcio.castro@ufsc.br

Resumo. *Processadores manycore possuem ambientes de desenvolvimento onerosos e suscetíveis a erros. Desta forma, abstrações surgiram a fim de facilitar o desenvolvimento sobre esses processadores. Este artigo demonstra o impacto de abstrações sobre o processador MPPA-256. Os resultados demonstraram ser viável desenvolver métodos de comunicação entre processos eficientes utilizando abstrações de baixo nível sem perder portabilidade e escalabilidade.*

1. Introdução

Processadores *manycore* foram introduzidos para possibilitar um aumento de desempenho, em contraste com abordagens tradicionais que recorriam ao aumento da frequência dos processadores. Assim, ao quantificar o desempenho não apenas pelo seu poder de processamento (FLOPS), como também em relação com o consumo de energia, processadores *manycore* viabilizaram supercomputadores alcançarem a era *exascale* (10^{18}) [Kogge et al. 2008].

Devido a uma arquitetura particular que: (i) integra até milhares de núcleos de baixa potência em um único chip; (ii) faz uso de uma *Network-on-Chip* (NoC) para comunicação de dados entre núcleos; e (iii) apresenta subsistemas de memória restritos, o desenvolvimento de aplicações paralelas para processadores *manycore* resume-se em uma tarefa desafiadora [Varghese et al. 2014, Castro et al. 2016]. Em função disso, recorre-se a abstrações para amenizar-se a complexidade de desenvolvimento através da manipulação de informações relevantes, sem um detalhamento excessivo da arquitetura ou do domínio da aplicação.

Embora as diferenças arquiteturais mencionadas anteriormente concedam a processadores *manycore* menor sobrecarga, maior escalabilidade e eficiência energética do que arquiteturas tradicionais, elas introduzem desafios no projeto de *software*. Logo, prover um ambiente de desenvolvimento que melhor relacione desempenho, escalabilidade e portabilidade ainda é um problema em aberto [Penna et al. 2018]. Nesse contexto, o presente trabalho apresenta uma avaliação do desempenho dos serviços *Inter-Process Communication* (IPC) implementados utilizando abstrações de alto e baixo nível do processador MPPA-256. Dois *microbenchmarks* foram utilizados com diversas configurações com o objetivo de fazer o comparativo entre as abstrações. Em resumo, os resultados mostram que o uso otimizado dos recursos de *hardware* proporcionam um aumento significativo do desempenho de aplicações no processador *manycore* estudado.

O restante desse trabalho está organizado da seguinte forma. A Seção 2 apresenta os aspectos relevantes de *hardware* e *software* do MPPA-256. A Seção 3 discute as semelhanças e diferenças das implementações e apresenta os *microbenchmarks* utilizados. Os resultados são apresentados na Seção 4 e, as conclusões, na Seção 5.

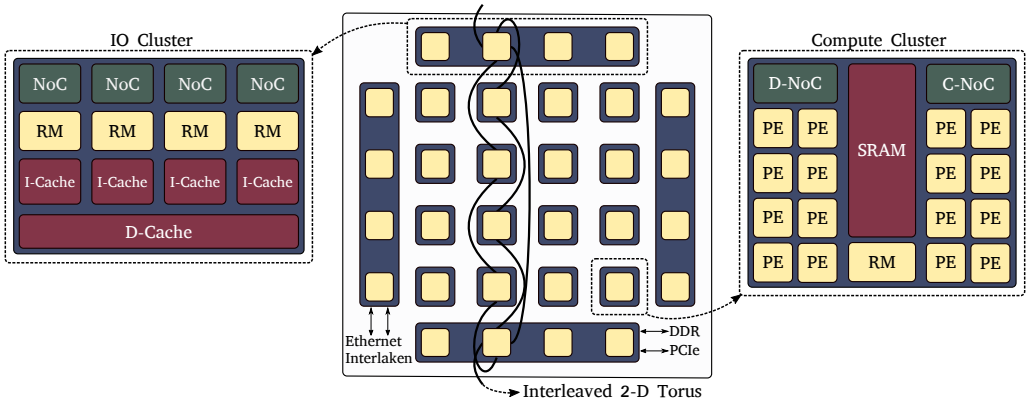


Figura 1. Visão arquitetural simplificada do MPPA-256 [Penna et al. 2018].

2. MPPA-256: Hardware e Software

O MPPA-256 é um processador *manycore* de alto desempenho e baixo consumo de energia desenvolvido pela empresa francesa Kalray. Como mostra a Figura 1, o processador possui 256 núcleos organizados em 16 *Compute Clusters* somados a 4 subsistemas de E/S (*IO Clusters*). Cada *Compute Cluster* apresenta: (i) 16 *Processing Element*s (PEs); (ii) um *Resource Manager* (RM); e (iii) espaço de endereçamento privado. A comunicação é realizada através de duas NoC distintas, *Data NoC* (D-NoC) e *Control NoC* (C-NoC). Com o processador, a Kalray disponibiliza um conjunto de *softwares* e bibliotecas para auxiliar o desenvolvimento de aplicações para o MPPA-256. Dentre esse conjunto e relevantes ao presente trabalho, as bibliotecas *MPPA IPC* e *MPPA NoC* provêem a comunicação entre processos através de interfaces de alto e baixo nível, respectivamente.

A biblioteca *MPPA IPC* disponibiliza uma interface baseada na especificação POSIX IPC. Toda a comunicação através da NoC é realizada mediante operações sobre descritores de arquivos especiais. Especificamente, tais arquivos seguem uma convenção que identifica: (i) a abstração desejada; (ii) os *clusters* envolvidos; e (iii) os *buffers* utilizados. As abstrações disponíveis exercem serviços de sincronização, controle de fluxo e transferência de dados entre processos. Para suportar esta biblioteca, o ambiente de execução deve ser configurado em conjunto com os sistemas de *runtime RTEMS* e *NodeOS*.

Por outro lado, a biblioteca *MPPA NoC* oferece uma interface de baixo nível que permite acesso direto aos recursos de comunicação das NoCs. A *MPPA NoC* foi projetada para manipular recursos e interrupções causando uma sobrecarga mínima de processamento. A biblioteca expõe: (i) 256 *buffers* de recebimento de dados; (ii) 128 *buffers* de recebimento de comandos; (iii) 8 *buffers* de envio de dados; (iv) 8 μ*threads* para envio de dados assíncronos; e (v) 1 *buffer* para envio de comandos. Por se tratar de uma biblioteca de baixo nível, a *MPPA NoC* não gerencia o fluxo nem a coerência de *cache* das operações. Porém, não necessita de um *runtime* auxiliar, apenas do *exokernel*, nativo.

3. Serviços IPC para o MPPA-256

Um sistema operacional, baseado em uma arquitetura *multikernel*, está sendo projetado e desenvolvido para *manycores*, incluindo o MPPA-256. Este sistema, denominado Nanvix[1], busca proporcionar um ambiente de execução que forneça a melhor relação entre desempenho, escalabilidade e portabilidade. Em sua versão atual, o Nanvix provê serviços

[1]Disponível em `https://github.com/nanvix`

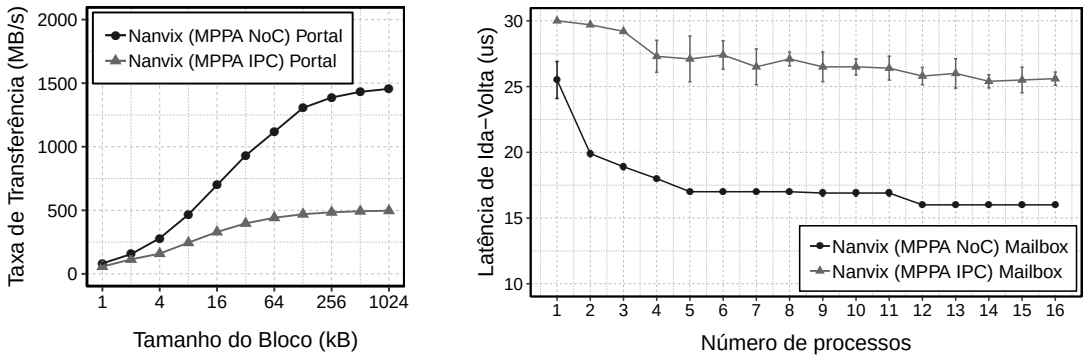
IPC como: (i) *Sync* para a sincronização de processos; (ii) *Mailbox* para fila de mensagens de tamanho fixo; e (iii) *Portal* para transferência de quantidades arbitrária de dados entre processos. Neste ponto, as operações dependentes de arquitetura são omitidas por uma *Camada de Abstração de Hardware* (HAL), facilitando o suporte às diversas arquiteturas. Em virtude disso, o presente artigo estuda e compara duas HALs que operam sobre bibliotecas e sistemas de *runtime* distintos.

Primeiramente, a HAL foi implementada utilizando a *MPPA IPC* para sincronização e comunicação através da NoC. Desta forma, a HAL é responsável apenas por gerir e proteger os descritores dos arquivos especiais. Por se tratar de uma biblioteca de comunicação síncrona, a *MPPA IPC* não utiliza os aceleradores em *hardware* disponíveis no MPPA-256 em suas comunicações, além de eliminar a possibilidade de otimização no uso dos recursos por realizar, implicitamente, o gerenciamento dos mesmos. Além disso, não se possuía conhecimento do seu impacto nas comunicações sobre o MPPA-256, em conjunto com os sistemas de *runtime* dos quais ela depende.

Estes fatos levaram ao desenvolvimento de uma nova HAL implementada com a *MPPA NoC*. Por ser uma biblioteca de baixo nível, toda a responsabilidade de alocação, configuração, proteção e gerenciamento dos recursos descritos na Seção 2 foi transferida para a HAL. Apesar da maior complexidade do código, foi possível desenvolver a mesma interface implementada pela *MPPA IPC*, pois ela necessita exatamente das mesmas informações utilizadas nos descritores dos arquivos especiais. Uma otimização realizada foi a multiplexação dos recursos da NoC reduzindo o uso exclusivo para momentos indispensáveis. Além disso, a utilização das μ*threads* para envio de dados permitiu prover um comportamento síncrono que utiliza-se dos aceleradores em *hardware* dedicado para transferência de dados existentes. Por fim, a dispensa de um sistema de *runtime* pela *MPPA NoC* levou a necessidade de implementar manualmente algumas abstrações, como *mutexes* e barreiras.

4. Resultados Experimentais

Para avaliar o impacto das abstrações serão utilizados dois tipos de *microbenchmarks*[2] que fazem uso intensivo dos serviços do Nanvix. Para garantir 95% de confiança em nossos resultados, foram realizados 10 replicações para cada configuração de cada *kernel*, conduzindo a um erro padrão inferior a 1%.



(a) *Portal* (b) *Mailbox*

Figura 2. Resultados Experimentais.

[2]Disponível em `https://github.com/nanvix/benchmarks`

Inicialmente, para avaliar o impacto no serviço *Portal*, um *kernel* efetuará a transferência de dados de 1 *IO Cluster* para cada um dos 16 *Compute Clusters*. Segundo, realizará o recebimento da mesma quantidade de dados no sentido oposto. Nenhuma computação é efetuada visto que buscamos avaliar o impacto nas comunicações. Este *kernel*, por sua vez, recebe como parâmetro o tamanho do bloco que deve ser manipulado, variando entre 1 KB e 1024 KB. Os resultados da Figura 2(a) mostram que, apesar do aumento da complexidade e tamanho do código, existe uma vantagem em otimizar o uso do *hardware* do MPPA-256. A taxa de transferência de dados aumentou em média 2.52×, não apresentando desvio padrão significativo (10^{-6} MB/s), por causa do uso das μ*threads* expostas pela *MPPA NoC* em relação ao envio síncrono realizado pela *MPPA IPC*. Da mesma forma, esse aspecto se apresentou nos resultados da taxa de recebimento de dados, aumentando em média 5.87×, com desvio padrão de 1,12 MB/s.

Por último, para comparar a latência na comunicação através da *Mailbox*, outro *kernel* efetuará a troca de uma mensagem entre 1 *IO Cluster* e todos os *Compute Clusters* envolvidos. Ao contrário do *kernel* anterior, o número de *Compute Clusters* incluídos na troca da mensagem varia de 1 a 16. A latência da *Mailbox*, Figura 2(b), apresentou uma diminuição em média de 35%, com desvio padrão de 0,37 μs, ao utilizar a *MPPA NoC* em relação a *MPPA IPC*. Neste caso, por se tratar de uma mensagem de tamanho pequeno, 120 bytes, foi utilizado o envio síncrono em ambas as HALs. Desta forma, o ganho de desempenho ocorreu devido ao menor impacto da *MPPA NoC* ao processador e na diferença da proteção dos recursos. Principalmente, pela mudança de *mutexes* na *MPPA IPC*, o que torna o processo de sincronização mais oneroso, por *spinlocks* da *MPPA NoC*, acarretando em uma diminuição na espera pela utilização de recursos.

5. Conclusão

Neste trabalho, foi comparado o desempenho e possibilidades no uso de diferentes tipos de bibliotecas disponíveis para comunicação entre processos para o MPPA-256. Os resultados mostraram uma grande vantagem no uso de bibliotecas de baixo nível ao otimizarem e melhor explorarem os recursos do processador. Como trabalhos futuros, pretende-se otimizar a comunicação utilizando a biblioteca *MPPA NoC* e realizar um comparativo com a biblioteca assíncrona de alto nível *MPPA Async* disponibilizada pela Kalray.

Referências

Castro, M., Francesquini, E., Dupros, F., Aochi, H., Navaux, P. O., and Méhaut, J.-F. (2016). Seismic wave propagation simulations on low-power and performance-centric manycores. *Parallel Computing*, 54:108–120.

Kogge, P., Borkar, S., Campbell, D., Carlson, W., Dally, W., Denneau, M., Franzon, P., Harrod, W., Hiller, J., Keckler, S., Klein, D., and Lucas, R. (2008). Exascale computing study: Technology challenges in achieving exascale systems. *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Techinal Representative*, 15.

Penna, P. H., Souza, M., Junior, E. P., do Nascimento, B. M., Castro, M., Broquedis, F., Freitas, H., and Méhaut, J.-F. (2018). An Operating System Service for Remote Memory Accesses in Low-Power NoC-Based Manycores. (October).

Varghese, A., Edwards, B., Mitra, G., and Rendell, A. P. (2014). Programming the Adapteva Epiphany 64-core network-on-chip coprocessor. In *International Parallel Distributed Processing Symposium Workshops (IPDPSW)*, pages 984–992, Phoenix, USA. IEEE Computer Society.