

A performance evaluation using an MPI library over the communication infrastructure of Nanvix

João Fellipe Uller

`joao.f.uller@grad.ufsc.br`

Universidade Federal de Santa Catarina (UFSC) – Brazil
Departamento de Informática e Estatística (INE)

INE410129-41000025DO/ME (20201) - Computação Paralela
Prof. Dr. Márcio Bastos Castro

November 10, 2020



Introduction

- Context
- Lightweight Manycores
- MPI
- Goal

LWMPI

- Overview
- Implementation

Evaluation

- Methodology
- Experimental Platform
- Results

Conclusions

Introduction

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

- Computational power vs energy consumption
- Lightweight manycores emerged to address high performance and energy efficiency demands ([FRANCESQUINI et al., 2015](#))

Characteristics

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

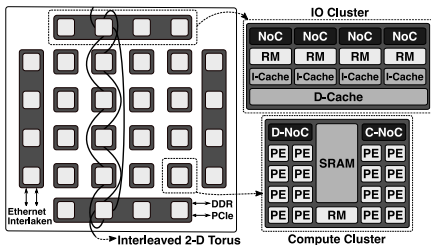
Methodology

Experimental Platform

Results

Conclusions

- Hundreds or thousands of low-power cores on a single chip
- Heterogeneous environment
- Distributed memory system with small local memories
- Communication through message passing on a rich Network-on-Chip (NoC)



MPPA-256 Architecture

Software Development Challenges

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

- Data fetching
- Data tiling
- Asynchronous communications ([HASCOËT et al., 2017](#))
- Hand-operated routing

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

Approaches to address programmability in lightweight manycores:

■ Operating Systems

- **Pros:** Bridges hardware intricacies and programmability gaps, providing **portability**
- **Cons:** Provided interface may be complex, retarding software development

■ Baremetal Runtime Systems

- **Pros:** Expose rich and performance-oriented interfaces narrowed to the underlying architecture
- **Cons:** Mostly vendor-specific, resulting in non-portable software

Duality: fast development process OR better software portability?

Message Passing Interface

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

- A **portable** message passing standard
- Maintained and defined by the **MPI-Forum**¹
- Widespread between **industry** and **academia**
- **De facto standard for message passing in HPC!**

¹MPI-Forum website: <https://www.mpi-forum.org>

Goal

Introduction

- Context
- Lightweight Manycores
- MPI
- Goal**

LWMPI

- Overview
- Implementation

Evaluation

- Methodology
- Experimental Platform
- Results

Conclusions

Provide a **light MPI-compliant library**, designed to cope with **architectural intricacies** of **lightweight manycores**, that is **portable** across multiple architectures and **easily extensible**.

Introduction

- Context
- Lightweight Manycores
- MPI
- Goal

LWMPI

- Overview
- Implementation

Evaluation

- Methodology
- Experimental Platform
- Results

Conclusions

LWMPI

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

LWMPI: Lightweight Message Passing Interface

- Compatible with 3.1 MPI specification (2015)
- Designed **from scratch** to be **light**
- Copes with **architectural intricacies** of lightweight manycores
- Implemented on top of a **POSIX-compliant distributed OS** (Nanvix²) to enable **portability** across different lightweight manycore architectures (PENNA et al., 2019)

²<http://www.github.com/nanvix>

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

LWMPI currently supports the following MPI features:

- *Runtime Management* (MPI_Init / MPI_Finalize)
- *Communicators*
- *Communication groups*
- *Error handlers*
- Standard *datatypes* for the C language
- Point-to-point communication (MPI_Send and MPI_Recv) in **synchronous mode**

Architecture

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

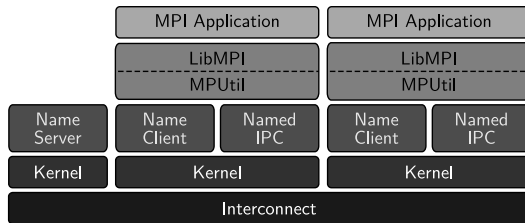
Evaluation

Methodology

Experimental Platform

Results

Conclusions



LWMPI Architecture

- **LibMPI**: top-level library that implements the MPI functions in a OS-independent way
- **MPUtil**: interface between LibMPI and the OS-level IPC system
 - Includes OS-dependent code

Nanvix Support to LWMPI

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

■ IPC Abstractions (SOUTO et al., 2020)

- Mailbox
- Portal
- Sync

■ Runtime Services

- Name Server
- Name Client
- Named IPC

Communication protocol for synchronous communications

Introduction

Context
Lightweight Manycores
MPI
Goal

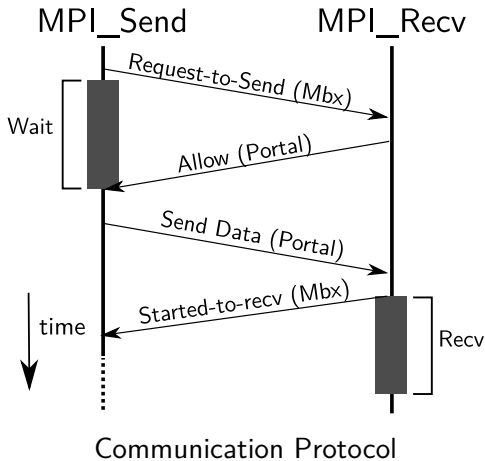
LWMPI

Overview
Implementation

Evaluation

Methodology
Experimental Platform
Results

Conclusions



Introduction

- Context
- Lightweight Manycores
- MPI
- Goal

LWMPI

- Overview
- Implementation

Evaluation

- Methodology
- Experimental Platform
- Results

Conclusions

Evaluation

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

■ Applications: CAP Bench Suite³ (SOUZA et al., 2017)

- Exercise different *parallel patterns*, *task types*, *comm. intensities* and *task loads*

App	Boundary	Parallel Pattern	Comm. Intensity
FN	CPU-bound	MapReduce	Light
GF	CPU/IO Bound	Stencil	Average
KM	IO-bound	Map	Heavy

■ Performance evaluation

- **Target architecture:** Kalray MPPA-256 (DINECHIN et al., 2013)
- **Baseline:** vendor-specific runtime (Kalray Runtime)
- **Performance metric:** speedup

³Publicly available at: <https://github.com/nanvix/benchmarks>

Kalray MPPA-256

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

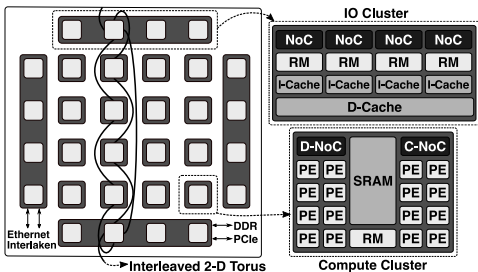
Methodology

Experimental Platform

Results

Conclusions

- 288 cores (256 GP cores + 32 firmware cores)
- 16 Compute Clusters (CCs)
- 4 I/O Clusters (IOs)
- 2 Network-on-Chips (C-NoC + D-NoC)



Experimental Scenarios

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

- CAP Bench apps have a single leader that coordinates the execution, with worker processes varying from 1 to 15 (max.)
- **Problems sizes:**
 - **FN:** numbers ranging from 1000001 to 1000129
 - **GF:** 512 × 512 image and 7 × 7 mask
 - **KM:** 30720 points and 64 centroids
- **Experimental design**
 - 30 trials for each configuration
 - Maximum CV < 1%

Introduction

Context
Lightweight Manycores
MPI
Goal

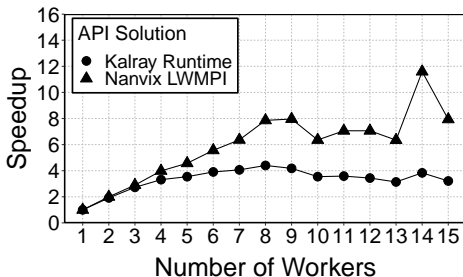
LWMPI

Overview
Implementation

Evaluation

Methodology
Experimental Platform
Results

Conclusions



- Communication has little interference, since the kernel is CPU-bound
- Load imbalance with more than 8 workers
- LWMPI shows better scalability
- Easy adaptation of the kernel without significant overheads

Introduction

Context
Lightweight Manycores
MPI
Goal

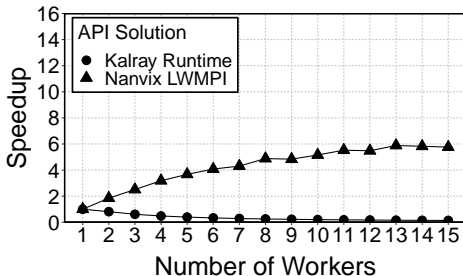
LWMPI

Overview
Implementation

Evaluation

Methodology
Experimental Platform
Results

Conclusions



- Small problem sizes resulted in insufficient workloads for the Kalray runtime
- Seems to be attenuated as the parallelism increases in LWMPI, proving better scalability also in these situations
- **Possibility of improvement:** asynchronous communications

Introduction

Context
Lightweight Manycores
MPI
Goal

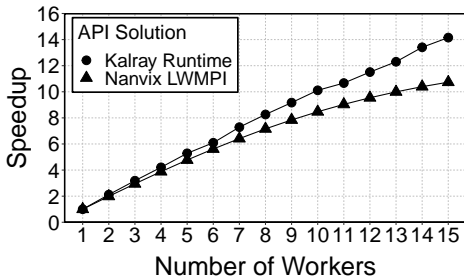
LWMPI

Overview
Implementation

Evaluation

Methodology
Experimental Platform
Results

Conclusions



- LWMPI and Kalray Runtime achieved similar speedups
- LWMPI showed slightly worse scalability due to coarse grain data transfers
- **Possibility of improvement:** a mechanism that dynamically chooses which OS-level comm. abstraction fits better the data transfer granularity

Introduction

- Context
- Lightweight Manycores
- MPI
- Goal

LWMPI

- Overview
- Implementation

Evaluation

- Methodology
- Experimental Platform
- Results

Conclusions

Conclusions

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

- **Our solution provides better programmability for lightweight manycores, because:**
 - It is based on an industry-standard interface (MPI)
 - It leverages a POSIX-compliant OS
 - It is portable across different lightweight manycore architectures
- **Experimental results**
 - LWMPI presents similar scalability for parallel and distributed problems, when compared to the vendor-specific runtime library (Kalray Runtime)

References I

Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions



DINECHIN, B. D. de et al. A Clustered Manycore Processor Architecture for Embedded and Accelerated Applications. In: *IEEE High Performance Extreme Computing Conf.* Waltham, USA: IEEE, 2013. p. 1–6. ISBN 978-1-4799-1365-7.



FRANCESQUINI, E. et al. On the Energy Efficiency and Performance of Irregular Application Executions on Multicore, NUMA and Manycore Platforms. *Journal of Parallel and Distributed Computing (JPDC)*, Elsevier - Academic Press, Orlando, v. 76, n. C, 2015. ISSN 0743-7315.



HASCOËT, J. et al. Asynchronous One-Sided Communications and Synchronizations for a Clustered Manycore Processor. In: *Symp. on Embedded Systems for Real-Time Multimedia*. Seoul: ACM Press, 2017. p. 51–60. ISBN 9781450351171.



PENNA, P. et al. On the Performance and Isolation of Asymmetric Microkernel Design for Lightweight Manycores. In: *Brazilian Symp. on Computing Systems Engineering*. Natal, Brazil: [s.n.], 2019. p. 1–8. ISSN 2324-7894.



SOUTO, J. V. et al. Mecanismos de comunicação entre clusters para lightweight manycores no nanvix os. In: *Escola Regional de Alto Desempenho da Região Sul*. Porto Alegre, RS, Brasil: SBC, 2020. (ERAD/RS '20), p. 1–4. ISSN 2595-4164.

References II

Introduction

- Context
- Lightweight Manycores
- MPI
- Goal

LWMPI

- Overview
- Implementation


Evaluation

- Methodology
- Experimental Platform
- Results

Conclusions



SOUZA, M. et al. Cap bench: A benchmark suite for performance and energy evaluation of low-power many-core processors. *Concurrency and Computation: Practice and Experience (CCPE)*, Wiley Online Library, v. 29, n. 4, p. 1–18, february 2017. ISSN 1532-0626.



Thank you!
Questions?

João Fellipe Uller

`joao.f.uller@grad.ufsc.br`

Universidade Federal de Santa Catarina (UFSC) – Brazil
Departamento de Informática e Estatística (INE)

INE410129-41000025DO/ME (20201) - Computação Paralela
Prof. Dr. Márcio Bastos Castro

November 10, 2020



Introduction

Context

Lightweight Manycores

MPI

Goal

LWMPI

Overview

Implementation

Evaluation

Methodology

Experimental Platform

Results

Conclusions

- An Open-Source, POSIX compliant, distributed OS that targets lightweight manycores (<https://github.com/nanvix/>)
- Designed in a distributed fashion **multikernel**
 - Multiple instances of an assymetric *microkernel*

