An Inter-Cluster Communication Facility for Lightweight Manycore Processors in the Nanvix OS

João Vicente Souto

joao.vicente.souto@grad.ufsc.br

Graduação em Ciência da Computação
Depto. de Informática e Estatísitca
Universidade Federal de Santa Catarina - Florianópolis

Orientador: Prof. Márcio Bastos Castro, Dr.
Coorientador: Pedro Henrique Penna, Me.

LAPESD
INE UFSC

September 27, 2020

UFSC

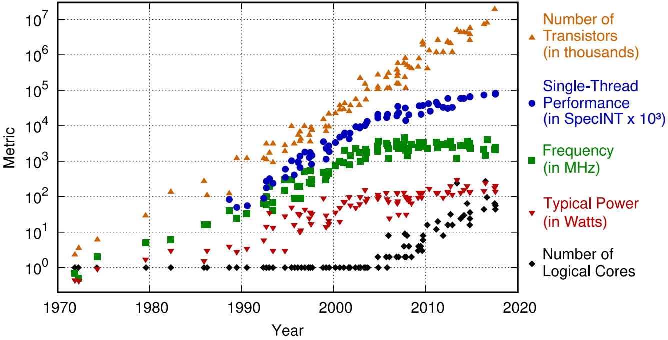
Presentation Outline

LAPESD

LAPESD

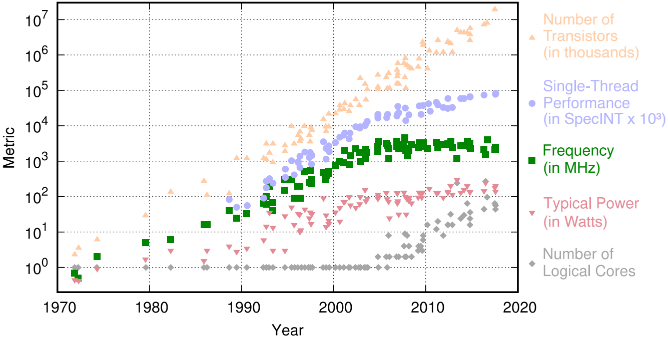
Introduction

Historical Evolution and Trend of Processors



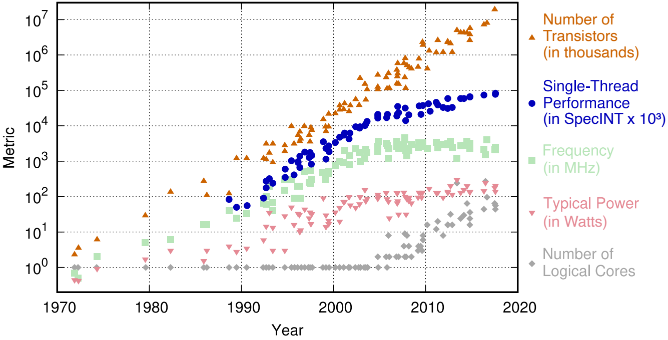
Increase in **processor performance**

Historical Evolution and Trend of Processors

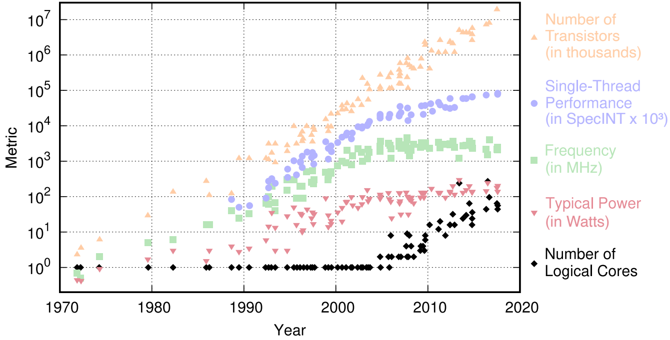


Limited by **frequency barrier**

Historical Evolution and Trend of Processors

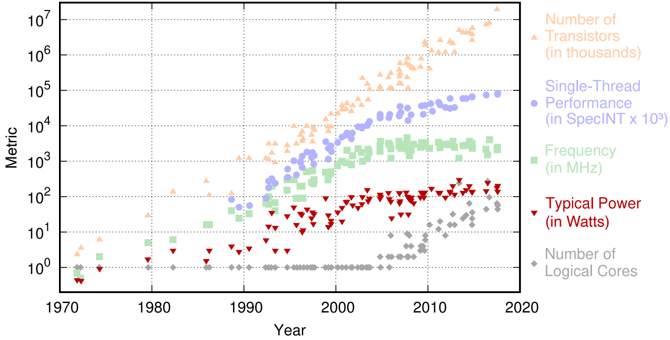


Softened by **technological advancement**



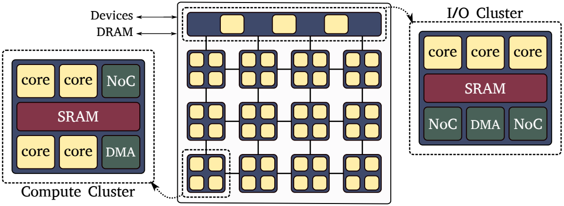
Emergence of **multicores and manycores**

Historical Evolution and Trend of Processors



Advent of **lightweight manycores**

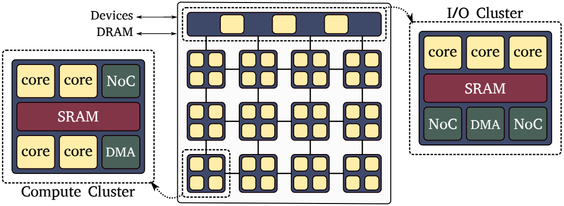
Lightweight Manycores Particularities



Overview of a Manycore

- **Hundreds of Lightweight Cores**
  - Expose Massive thread-level parallelism
  - Feature low-power consumption
  - Target MIMD workloads
- Distributed Memory Architecture
- On-Chip Heterogeneity

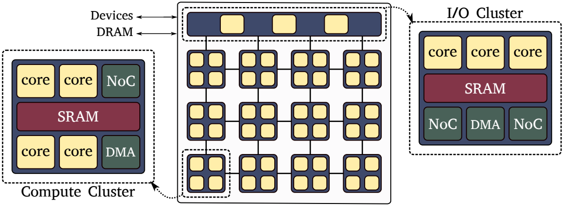
Lightweight Manycores Particularities



Overview of a Manycore

- Hundreds of Lightweight Cores
- **Distributed Memory Architecture**
  - Grants scalability
  - Relies on a Network-on-Chip (NoC)
  - Has constrained memory systems
- On-Chip Heterogeneity

Lightweight Manycores Particularities



Overview of a Manycore

- Hundreds of Lightweight Cores
- Distributed Memory Architecture
- **On-Chip Heterogeneity**
  - Features different components

Goals

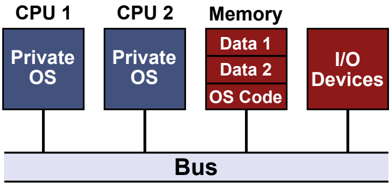
Context and Goals

- **Work context**
  - Challenges arising from **runtimes** and **OSs**
  - OSs for **next-generation** for lightweight manycores
  - **Nanvix OS**
- **Goals**
  - Definition and proposal of an **Inter-Cluster Communication Interface** for lightweight manycores
  - Implementation in the **Nanvix HAL** on Kalray MPPA-256 Lightweight Manycore Processor
  - Integration with the **Nanvix Microkernel**
  - Performance evaluation using synthetic micro-benchmarks with **collective communication routines**

Background

Models of Operating Systems (OS) for Multicores

- **Replicated OS**
- Master-Slave OS
- Symmetric OS

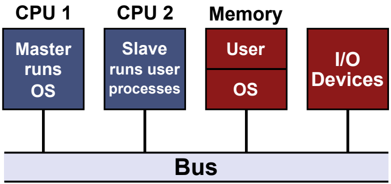


Each core has a **copy of the OS**

Less concurrency but needs **a lot of memory**

Models of Operating Systems (OS) for Multicores

- Replicated OS
- **Master-Slave OS**
- Symmetric OS

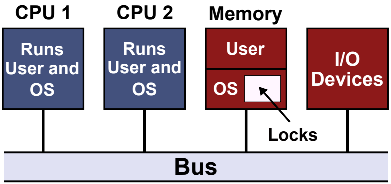


Asymmetric execution where only one core handles the OS

Better scalability and **less inter-core interference** but inserts a possible **bottleneck**

Models of Operating Systems (OS) for Multicores

- Replicated OS
- Master-Slave OS
- **Symmetric OS**



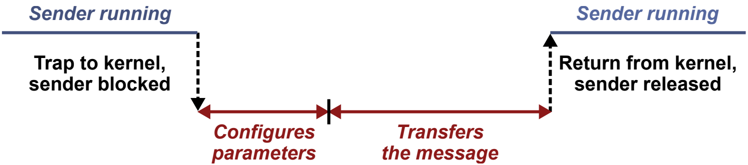
OS shared by all cores

Efficient with few cores and **cache consistency** but has **inter-core interference**

- **Computer network concepts**
- Network interfaces
- Performance impacts
  - Number of intermediate copies
  - Direct Memory Access (DMA)

User-Level Communication for Message-Passsing Model

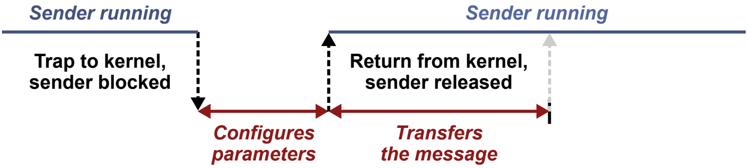
- Send/receive primitives
  - **Synchronous calls**
  - Asynchronous calls



Requester waits for task to be completed

User-Level Communication for Message-Passsing Model

- Send/receive primitives
  - Synchronous calls
  - **Asynchronous calls**

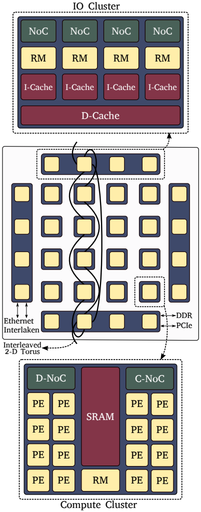


Requester is released to continue to run in parallel

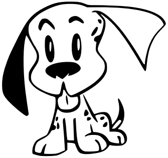
Kalray MPPA-256

A Lightweight Manycore Processor

- **288 processing cores**
  - 16 Compute Cluster (CC)
  - 4 I/O Cluster (IO)
- **Data NoC (D-NoC)**
  - 256 RX slots
  - 8 TX channels
  - 8 μthreads for async TX
- **Control NoC (C-NoC)**
  - 128 RX slots
  - 4 TX channels

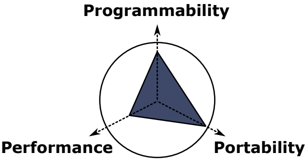


- **Open source** and **collaborative** project
  - Home-grown instructional OS
  - 4 Professors (Brazil and France)
  - 1 PhD, 1 MSc and 3 BSc Students
  - 9 past contributors
  - UGA, PUC Minas, UFSC, and Grenoble INP



Bingo, our mascot

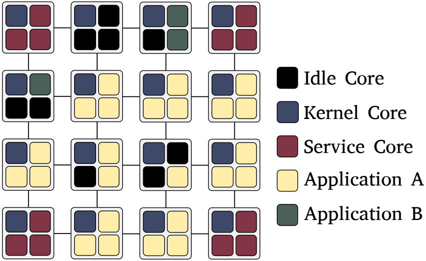
- **Long-Term Goal**
  - General purpose OS
  - POSIX-Compliant



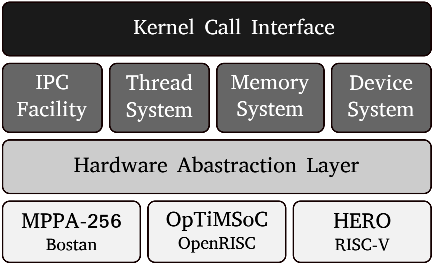
Nanvix Multikernel

Multikernel OS Structure

- **OS services are system processes** and run in isolation
- User processes **request services via message-passing**



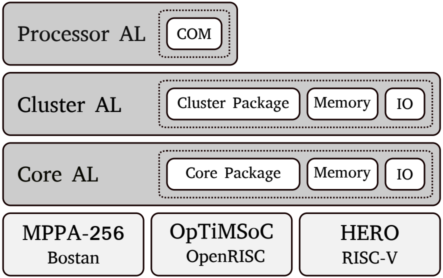
- Follows a **Master-Slave OS** model
- Provides **bare-bones of system abstractions**
- Rich system call interface (Kernel Call)



Nanvix Hardware Abstraction Layer (HAL)

Kernel portability

- **Generic and flexible** hardware abstraction layer
- **Standard view** of these emerging processors



Development

Inter-Cluster Communication Interface

LAPESD

- **Three communication abstractions**
  - Sync
  - Mailbox
  - Portal
- More precise
- Easy-to-use
- Scalable
- Easily portable

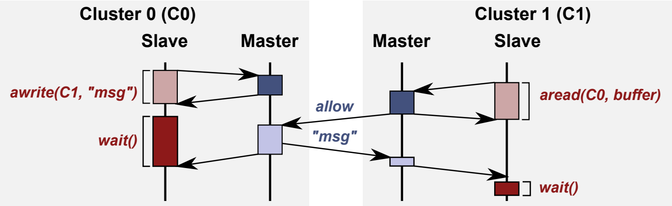
MPPA-256 Hardware Features

- **Hardware independence** in resource identification
  - NoC interfaces (Physical to Logical)
  - NoC resources (Partitioned by abstraction)
- Low-level communication depends on **two features**
  - Interrupt System
  - DMA
- **Work limitations**
  - Poor documentation and examples
  - Reverse engineering
  - Hardware bugs
  - **No asynchronous sends**

LAPESD

General Concepts of Comm. Abstrations

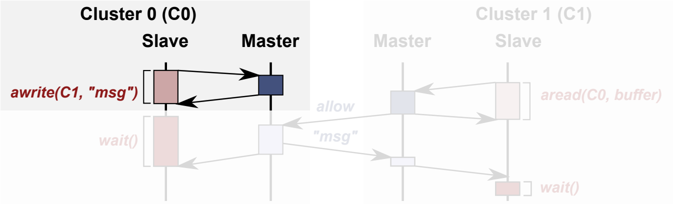
- **Naming Convention** (suffixes)
  - Sender: `open/close/awrite/signal/wait`.
  - Receiver: `create/unlink/aread/wait`
- Interfaces export **only asynchronous calls**
- Master core **cannot be blocked**



Lazy transfer behavior

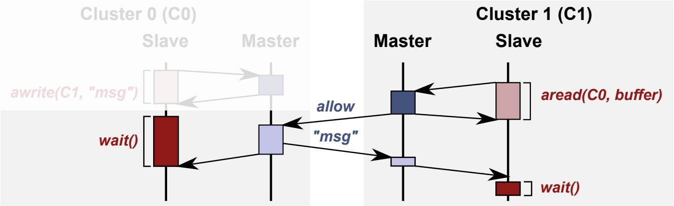
General Concepts of Comm. Abstrations

- **Naming Convention** (suffixes)
  - Sender: `open`/`close`/`awrite`/`signal`/`wait`.
  - Receiver: `create`/`unlink`/`aread`/`wait`
- Interfaces export **only asynchronous calls**
- Master core **cannot be blocked**



Master without sending permission **keeps the parameters**

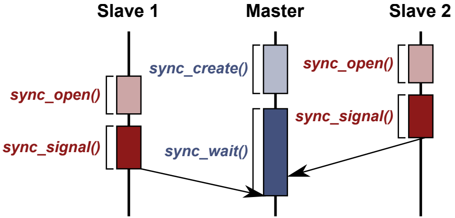
- **Naming Convention** (suffixes)
  - Sender: `open`/`close`/`awrite`/`signal`/`wait`.
  - Receiver: `create`/`unlink`/`aread`/`wait`
- Interfaces export **only asynchronous calls**
- Master core **cannot be blocked**



Master sends data upon allow and releases slave

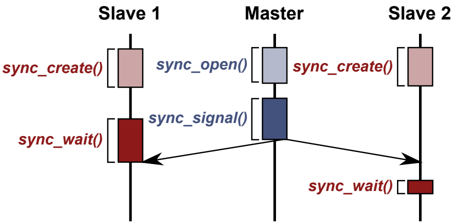
Sync Abstration

- Provides cluster synchronization across **distributed barriers**
- Similarly to **POSIX Signals**
- Two modes
  - **ALL_TO_ONE**
  - ONE_TO_ALL



Sync Abstration

- Provides cluster synchronization across **distributed barriers**
- Similarly to **POSIX Signals**
- Two modes
  - ALL_TO_ONE
  - **ONE_TO_ALL**

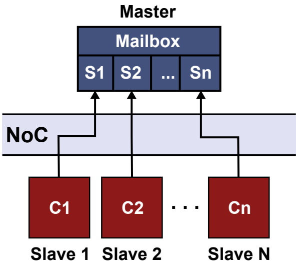


Sync Abstration Implementation

- Uses only **C-NoC** Resources
- **Node list** of involved Logic IDs
- Master node must always be the first
- RX resources **related to the Master ID**

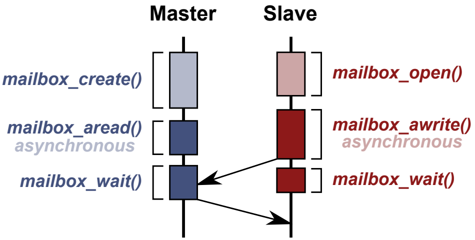
Mailbox Abstration

- Allows exchange **fixed-length message**
- Similarly to **POSIX Message Queue**
- Receiver allocates space to N messages
- Sender transfer to predefined location



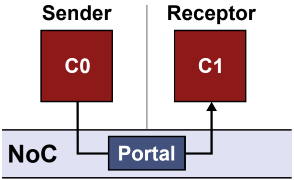
Mailbox Abstration Implementation

- Merges the use of **D-NoC and C-NoC**
- Message queue allocated within the kernel space
- **Quality of Service**
  - **One message** per NoC Node



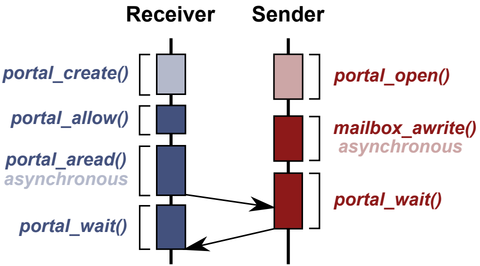
Portal Abstration

- Allows exchange **arbitrary amounts of data**
- Similarly to **POSIX Pipes**
- **One-way** channel for data transfer



Portal Abstration Implementation

- Merges the use of **D-NoC and C-NoC**
- No intermediate copies
- **Quality of Service**
  - *Receiver notifies sender* that it is able to receive



Integration with Nanvix Microkernel

- **Rich resource management** is forwarded to the microkernel
- Nanvix Microkernel seek **to provide**:
  - Protection
  - Management
  - Multiplexing
- **Impacts of the Master-Slave OS Model**
  - Only master changes the internal structures of the OS
  - Slaves allowed to read lock addresses to wait for async calls

Microkernel

Protection, Management and Multiplexing

- **Protection** and **management** involve two phases
  - **Slave phase**
    - valid file descriptors
    - non-null buffer pointers
    - buffer sizes within the stipulated limit
  - **Master phase**
    - conflicting operations
    - measures communication parameters
    - interacts with Nanvix HAL detecting errors
- **Multiplexing**
  - Identifies creations/openings with **same arguments**
  - Structures keep a **reference counter**
  - Operations of read/write set resources **to busy**
  - Forces **serialization** of the operations

LAPESD

Validation and Correctness Tests

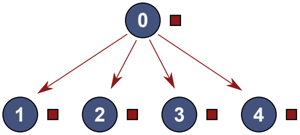
- Extra effort in **ensuring semantics** of implementations **across multiple architectures**
- **API tests**
  - Arguments within valid value ranges
  - Operations with valid semantic
- **FAULT tests**
  - Arguments outside domain range
  - Operations with invalid semantic
  - Expected error value

LAPESD

Experiments

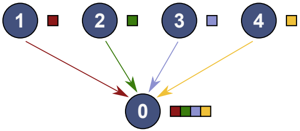
Methodology

- Evaluation of the performance of **data transfer services**
  - Sync + Mailbox
  - Sync + Portal
- Four well-known **collective communication routines**
  - **Broadcast**
  - Gather
  - AllGather
  - Ping-Pong



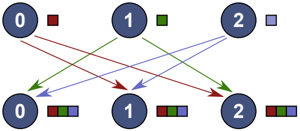
Methodology

- Evaluation of the performance of **data transfer services**
  - Sync + Mailbox
  - Sync + Portal
- Four well-known **collective communication routines**
  - Broadcast
  - **Gather**
  - AllGather
  - Ping-Pong



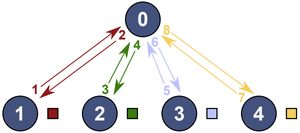
Methodology

- Evaluation of the performance of **data transfer services**
  - Sync + Mailbox
  - Sync + Portal
- Four well-known **collective communication routines**
  - Broadcast
  - Gather
  - **AllGather**
  - Ping-Pong



Methodology

- Evaluation of the performance of **data transfer services**
  - Sync + Mailbox
  - Sync + Portal
- Four well-known **collective communication routines**
  - Broadcast
  - Gather
  - AllGather
  - **Ping-Pong**

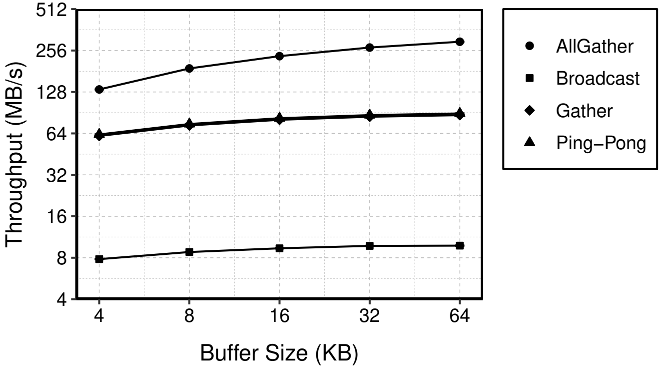


Experimental Design

- **Throughput of the Portal**
  - Constant: 1 IO and 16 CCs
  - Variable: 4, 8, 16, 32, 64 KB
- **Latency of the Mailbox**
  - Constant: 120 B
  - Variable: 1 IO and 1 to 16 CCs
- **50 iterations**, first 10 discardard to warmup
- Metrics do not represent an aggregation
- Standard error inferior to **1%**

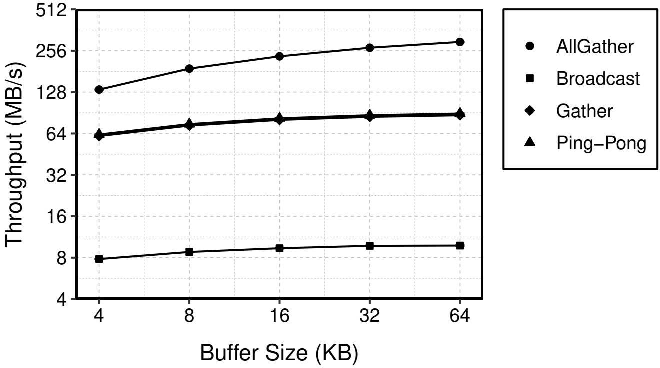
LAPESD

Portal Analysis



Three distinct behaviors

Portal Analysis



Optimum size between 8 KB and 16 KB

Mailbox Analysis

Three distinct behaviors too

Mailbox Analysis

Well-known **distributed algorithms can be efficiently supported** by Nanvix OS

Conclusions

Conclusions

- **Motivation**
  - Historical evolution from single-cores to Lightweight Manycores
- **Contribution**
  - A Inter-Cluster Communication Facility for LW Processors
- **Results**
  - **Optimal sizes** for large data transfers
  - Well-known **distributed algorithms can be efficiently supported** by Nanvix OS
- **Future Works on Nanvix OS**
  - Remove limitation on asynchronous send
  - MPI port (BSc dissertation)
  - Shared Memory Service (MSc dissertation)
  - **Distributed Process Scheduling (MSc dissertation)**

LAPESD

Thank you!
Questions?

João Vicente Souto

joao.vicente.souto@grad.ufsc.br

Graduação em Ciência da Computação
Depto. de Informática e Estatísitca
Universidade Federal de Santa Catarina - Florianópolis

Orientador: Prof. Márcio Bastos Castro, Dr.
Coorientador: Pedro Henrique Penna, Me.

September 27, 2020

References I

BAUMANN, A. et al. The Multikernel: A New OS Architecture for Scalable Multicore Systems. In: *Proceedings of the 22nd ACM Symposium on Operating Systems Principles*. Big Sky, Montana, USA: ACM, 2009. (SOSP '09), p. 29–44. ISBN 978-1-60558-752-3. Disponível em: <https://dl.acm.org/citation.cfm?doid=1629575.1629579>.

DINECHIN, B. D. de et al. A Distributed Run-Time Environment for the Kalray MPPA-256 Integrated Manycore Processor. In: *Procedia Computer Science*. Barcelona, Spain: Elsevier, 2013. (ICCS '13, v. 18), p. 1654–1663. ISBN 1877-0509. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1877050913004766>.

KENDALL, W.; NATH, D.; BLAND, W. *A Comprehensive MPI Tutorial Resource*. 2019. <https://mpitutorial.com/>, Last accessed on 2019-10-22.

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/>.

LAPESD

References II

PENNA, P. H. et al. Using The Nanvix Operating System in Undergraduate Operating System Courses. In: *VII Brazilian Symposium on Computing Systems Engineering*. Curitiba, Brazil: [s.n.], 2017. Disponível em: <https://hal.archives-ouvertes.fr/hal-01635880>.

PENNA, P. H. et al. RMem: An OS Service for Transparent Remote Memory Access in Lightweight Manycores. In: *MultiProg 2019 - 25th International Workshop on Programmability and Architectures for Heterogeneous Multicores*. Valencia, Spain: [s.n.], 2019. (High-Performance and Embedded Architectures and Compilers Workshops (HiPEAC Workshops)), p. 1–16. Disponível em: <https://hal.archives-ouvertes.fr/hal-01986366>.

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

RUPP, K. *Microprocessor Trend Data*. 2018. <https://github.com/karlrupp/microprocessor-trend-data>, Last accessed on 2019-06-26.

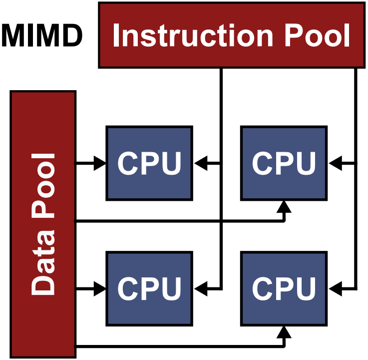
SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Operating System Concepts*. 9th. ed. [S.l.]: Wiley Publishing, 2012. ISBN 1118063333, 9781118063330.

References III

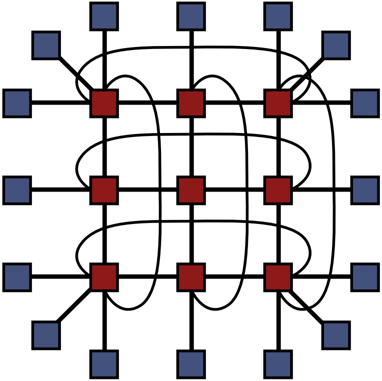
TANENBAUM, A. S.; BOS, H. *Modern Operating Systems*. 4th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2014. ISBN 013359162X, 9780133591620.

WICKRAMASINGHE, U.; LUMSDAINE, A. A survey of methods for collective communication optimization and tuning. *CoRR*, abs/1611.06334, 2016. Disponível em: <http://arxiv.org/abs/1611.06334>.

MIMD workload

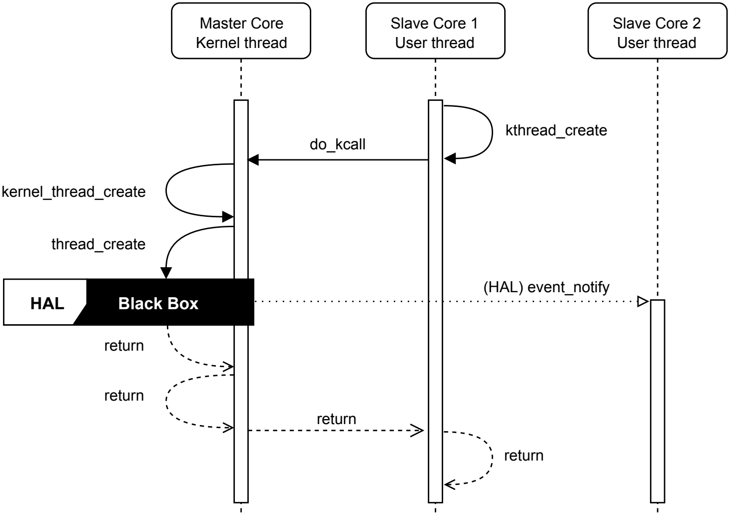


Type of network-on-chip of the MPPA-256

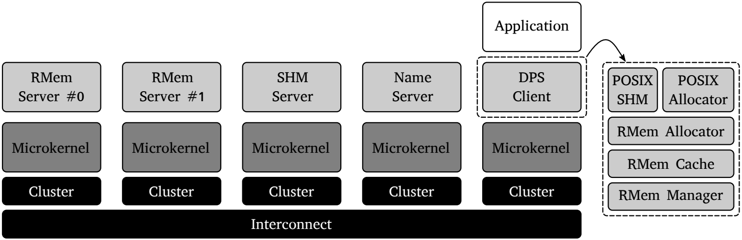


Torus network

Execution example of the Nanvix Microkernel



POSIX Compliance example of Nanvix Multikernel



Network-on-Chip Identifiers

	Physical ID	Logical ID
IO 0	128-131	0-3
IO 1	192-195	4-7
CCs	0-15	8-23

Resource Identifiers

	C-NoC		D-NoC	
	RX ID	**TX ID**	**RX ID**	**TX ID**
Mailbox	0-23	0	0-23	1-3
Portal	24-47	1-2	24-47	4-7
Sync	48-71	3	-	-

Simplified NoC handler algorithm

Require: $status[M_{Interfaces}][N_{Resources}]$, interrupt status of a resource.

Require: $handlers[M_{Interfaces}][N_{Resources}]$, interrupt handler of a resource.

1: **procedure** NOC_HANDLER
2:     **for** $i \in [1, M_{Interfaces}]$ **do**
3:         **for** $j \in [1, N_{Resources}]$ **do**
4:             **if** $status[i][j] == InterruptTriggered$ **then**
5:                 CLEAN_STATUS(i, j)
6:                 HANDLERS[I][J](i, j)

Simplified lazy transfer algorithm

Require: *resources*, Abstraction Resource Table

```
        ▷ Configures data transfer.
procedure ASYNC_WRITE(id, message, size)
    resources[id].message ← message
    resources[id].size ← size
    if resources[id].has_permission then
        DO_LAZY_TRANSFER(id)
    else
        resources[id].is_waiting ← True
        ▷ Receives permission.
procedure ABSTRACTION_HANDLER(id)
    if resources[id].is_waiting then
        DO_LAZY_TRANSFER(id)
    else
        resources[id].has_permission ← True
        ▷ Transfers the data.
procedure DO_LAZY_TRANSFER(id)
    resources[id].is_waiting ← False
    resources[id].has_permission ← False
    TRANSFER_DATA(resources[id].message, resources[id].size)
    UNLOCK(resources[id].lock)                ▷ Releases slave core.
```

LAPESD

References used

- Processor trend (RUPP, 2018)
- Kalray MPPA-256 (DINECHIN et al., 2013)
- Multikernel (BAUMANN et al., 2009)
- Nanvix Project (PENNA et al., 2017b; PENNA et al., 2017a; PENNA et al., 2019; PENNA et al., 2018)
- Concepts about OSs (TANENBAUM; BOS, 2014)
- Models of OSs (SILBERSCHATZ; GALVIN; GAGNE, 2012)
- MPI collective communication routines (WICKRAMASINGHE; LUMSDAINE, 2016; KENDALL; NATH; BLAND, 2019)