Partitioning and Task Transfer on NoC-based Many-Core Processors in the Avionics Domain

J. Reinier van Kampenhout and Robert Hilbrich

30.06.2011

Fraunhofer
FIRST

Fraunhofer FIRST

Fraunhofer-Institut für Rechnerarchitektur und Softwaretechnik, Berlin

Departments:

- Modeling
- Systems Architecture
- Quality Assurance

Contents

I. TRENDS

Introduction

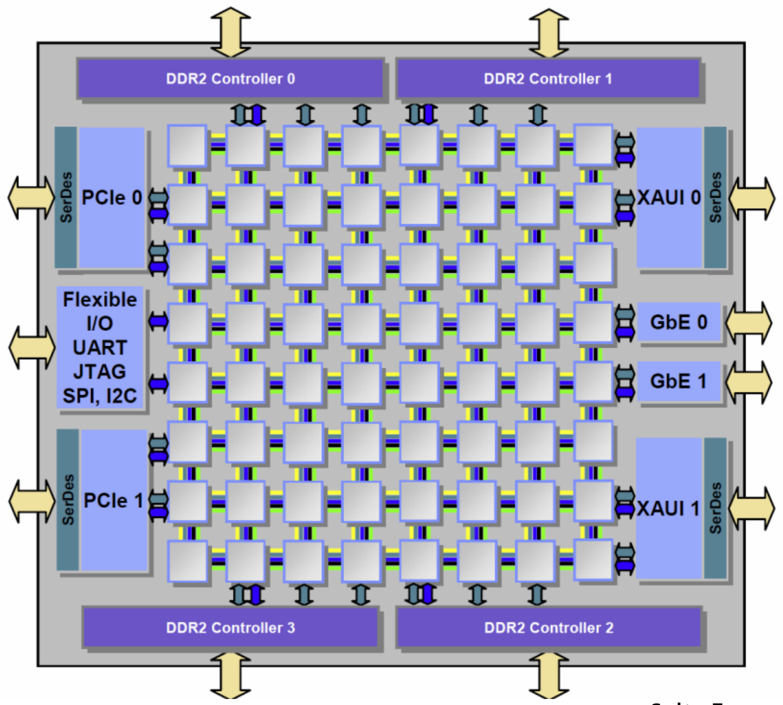
Many-core processors (> 32 cores):

- Increased performance without excessive power consumption
- Less complex cores ➔ simplifies timing analysis

Potentially save Space, Weight and Power (SWaP) in avionics systems

Challenges:

- Parallelization/consolidation of software
- Efficient deployment
- Reliability



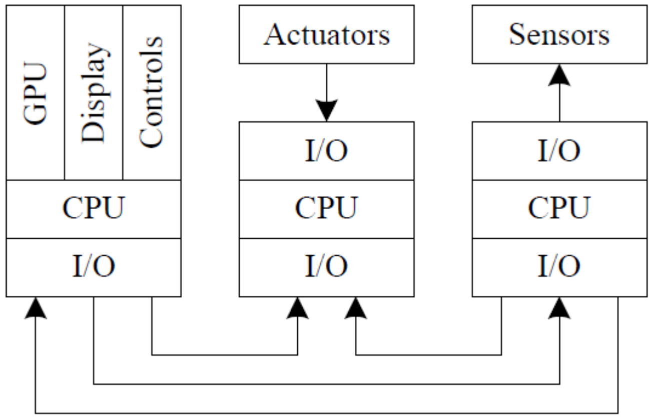
© Tilera Corporation.

Fraunhofer FIRST

Software Partitioning in Avionics Systems 1/2

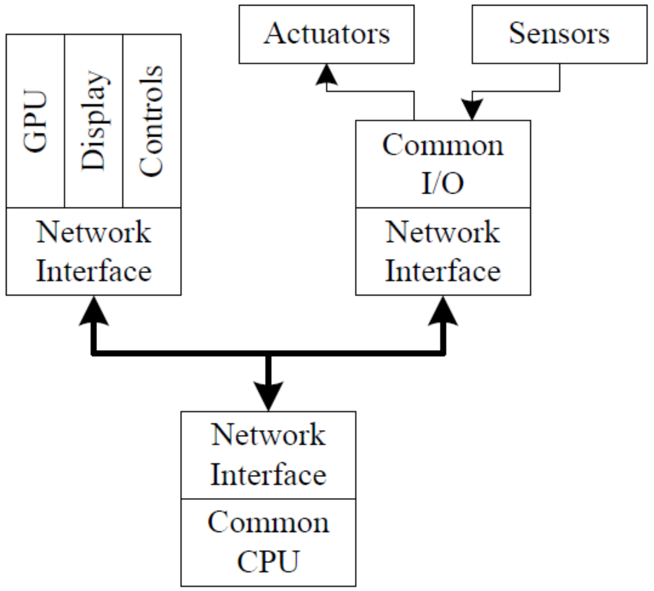
Federated architectures:

- "One function – one computer"
- Fault containment by design



Integrated Modular Avionics (IMA):

- Shared computing platform
- Reduce cost and SWaP requirements
- Achieve fault containment through software **partitioning**



Source: Watkins2007, Transitioning from federated avionics architectures to integrated modular avionics.

Software Partitioning in Avionics Systems 2/2

Partitioning:

- Isolate groups of related tasks:
  - Temporal partitioning
  - Spatial partitioning
- Avoid non-transparent fault propagation

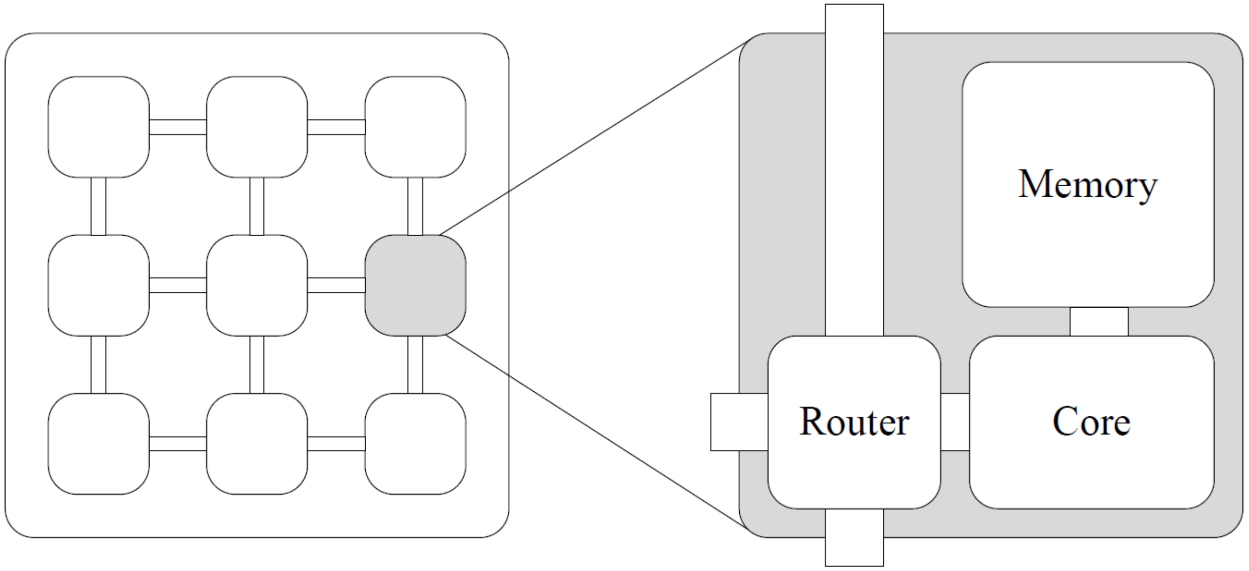
Task execution models:

- Synchronous tasks:
  - Periodic
  - Scheduled statically
- Asynchronous tasks:
  - Event or data driven

Many-Core Processors and Networks-on-Chip 1/2

Many-cores:

- Increase throughput instead of clock frequency
- Scalable on-chip interconnect: networks
- Communication-centric design:
  - Inter-core communication and access to shared resources
  - Complicates timing analysis



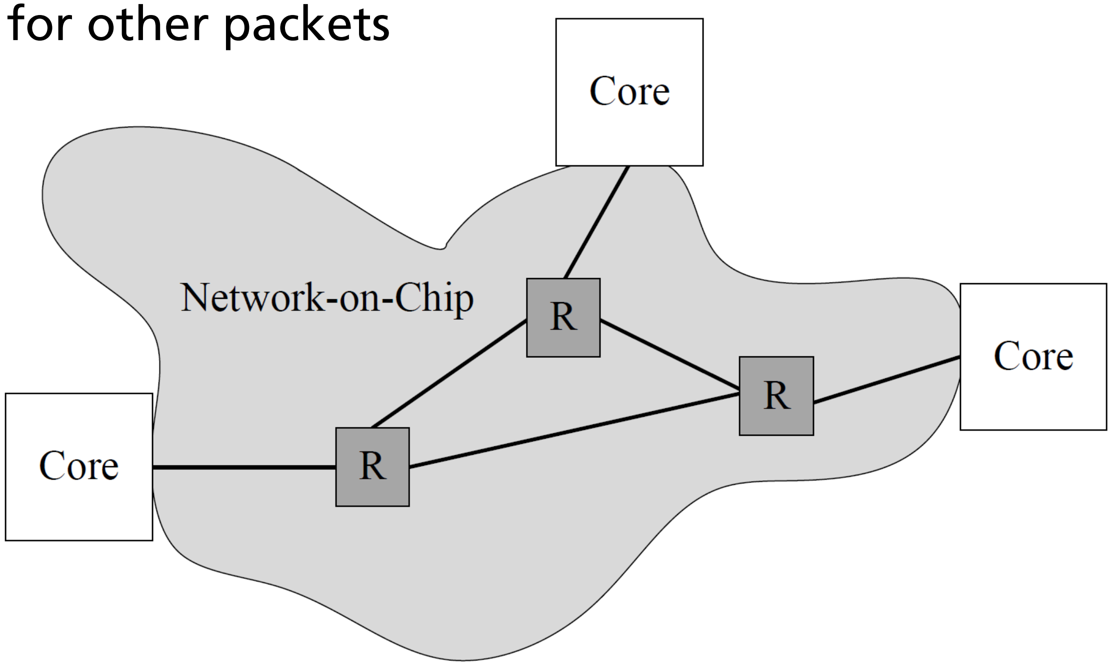
Fraunhofer FIRST

Many-Core Processors and Networks-on-Chip 2/2

Networks-on-Chip (NoC):

- Scalable networks based on packet switching
- Problems:
  - Contention: concurrent access of shared links and routers
  - Congestion: packets wait for other packets
  - Unpredictable delays

- Solutions:
  - Resource reservation
  - Quality-of-Service (QoS):
    - Flow control
    - Buffering



Fraunhofer FIRST

II. CONTRIBUTION

Partitioning on Many-Core Processors 1/3

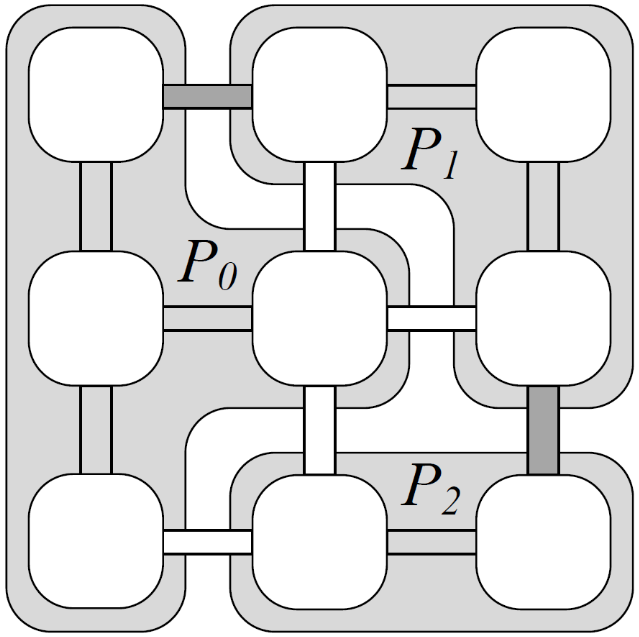
We propose to extend partitioning to deal with many-core architectures

Temporal partitioning:

– Per-core schedule

Spatial partitioning:

– Mapping of tasks to cores
– Mapping of traffic onto the NoC, requires:
  · Per-link schedule
  · Traffic profiling
  · Reservation mechanisms, e.g. circuit switching
  · Isolation

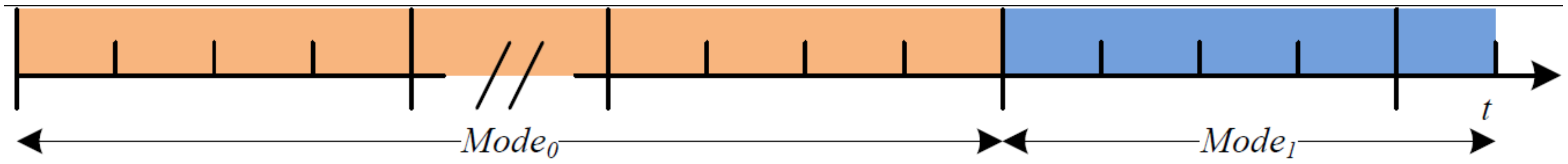


Fraunhofer FIRST

Partitioning on Many-Core Processors 2/3

Proposed partition types:

1. Fixed:
   - Highly critical synchronous tasks, statically mapped and scheduled
   - Completely deterministic but costly
2. Mode-based:
   - Mapping and schedule change according to *modes*
   - Optimization of resource usage
3. Flexible:
   - Asynchronous low-critical tasks
   - Dynamic reconfiguration allows to exploit idle resources



Fraunhofer FIRST

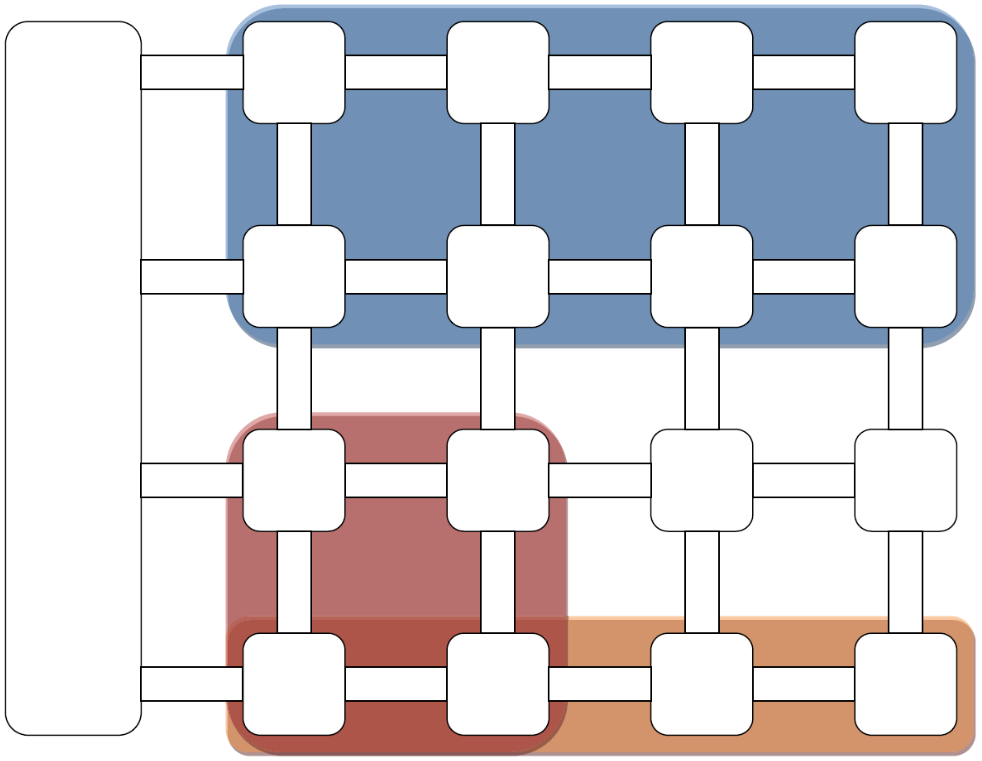
Partitioning on Many-Core Processors 3/3

Reconfiguration:

- Resize
  - Borrow idle resources
- Relocate
  - Decrease communication distance

- Fault-tolerance:
  - Duplicating faulty partition
  - Avoid use of faulty hardware

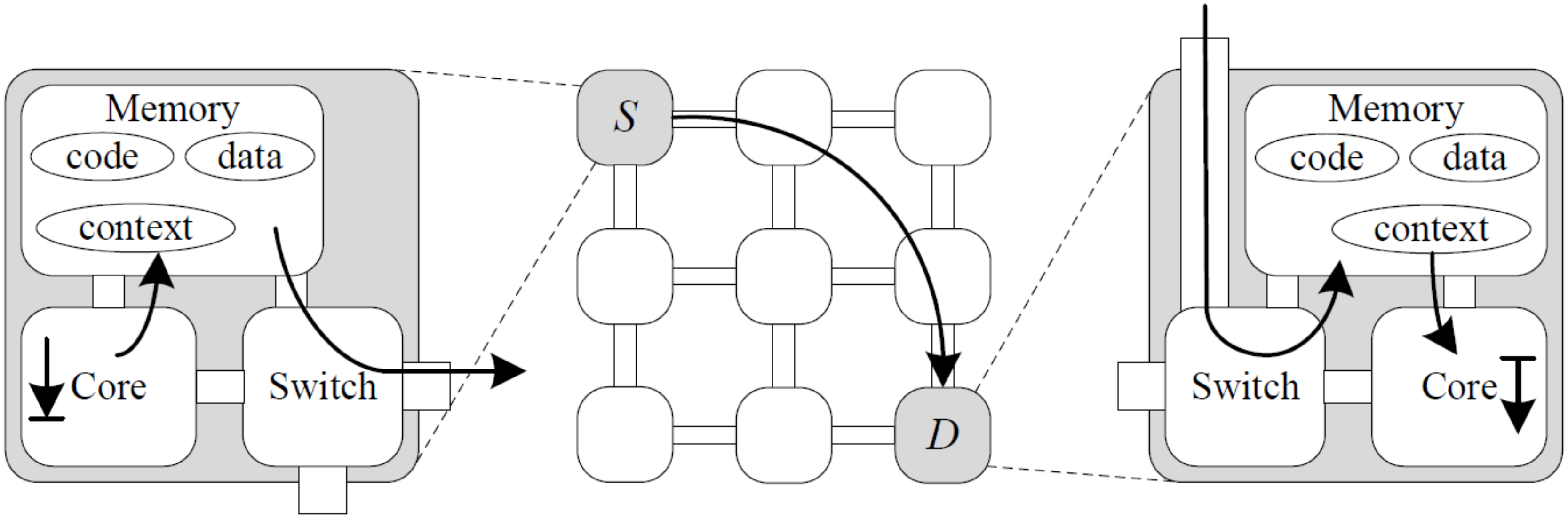
- Requires task migration



Task Migration

Different from migration in multi-processor systems:

- Small local memory
- Limited OS functionality
- Transfer over Network-on-Chip
  - Faster



III. EXPERIMENTS

Experimental Setup 1/2

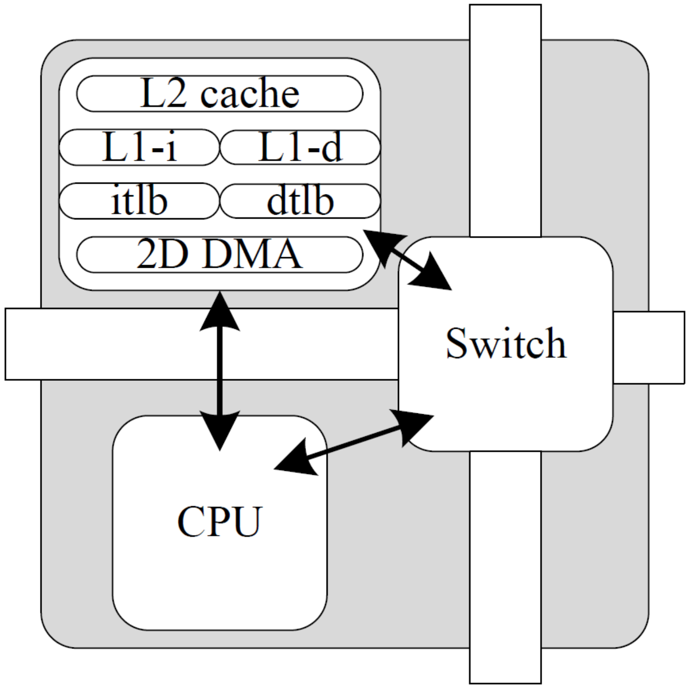
Goal: evaluate which transfer methods can offer deterministic task migration

Experiments:

– Migration of code and dataset

Test platform: Tilera TILE*Pro*64™

- 64 cores
- Separate L1-data and -instruction caches
- L2 cache, aggregate can be combined into coherent shared L3 cache
- Six 2-d mesh networks
  - XY wormhole routing
  - Two programmer accessible networks



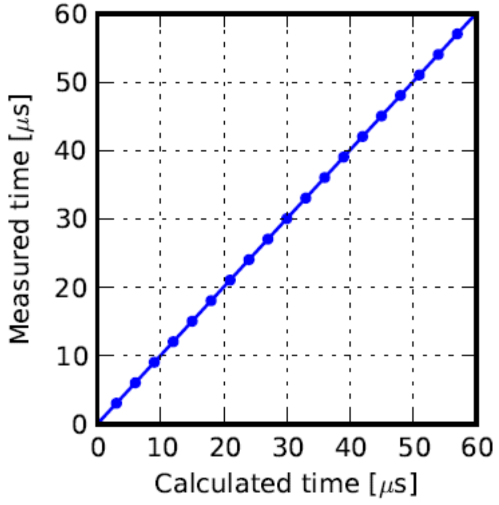
Experimental Setup 2/2

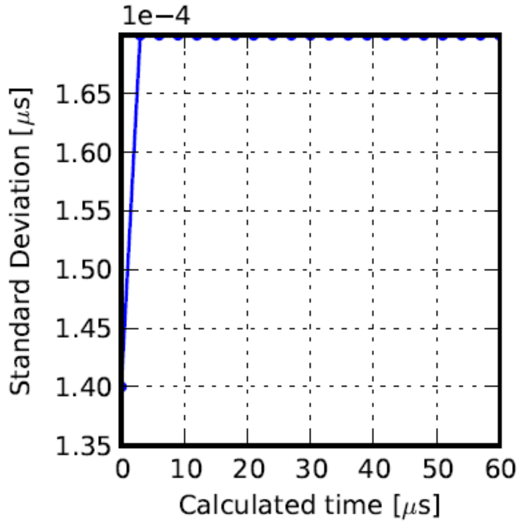
Data transfer methods:

- With coherent shared cache:
  - Cache pull
  - Prefetching
  - Explicit copy
- With message passing:
  - User Dynamic Network (UDN)
  - STatic Network (STN)

Time measurements:

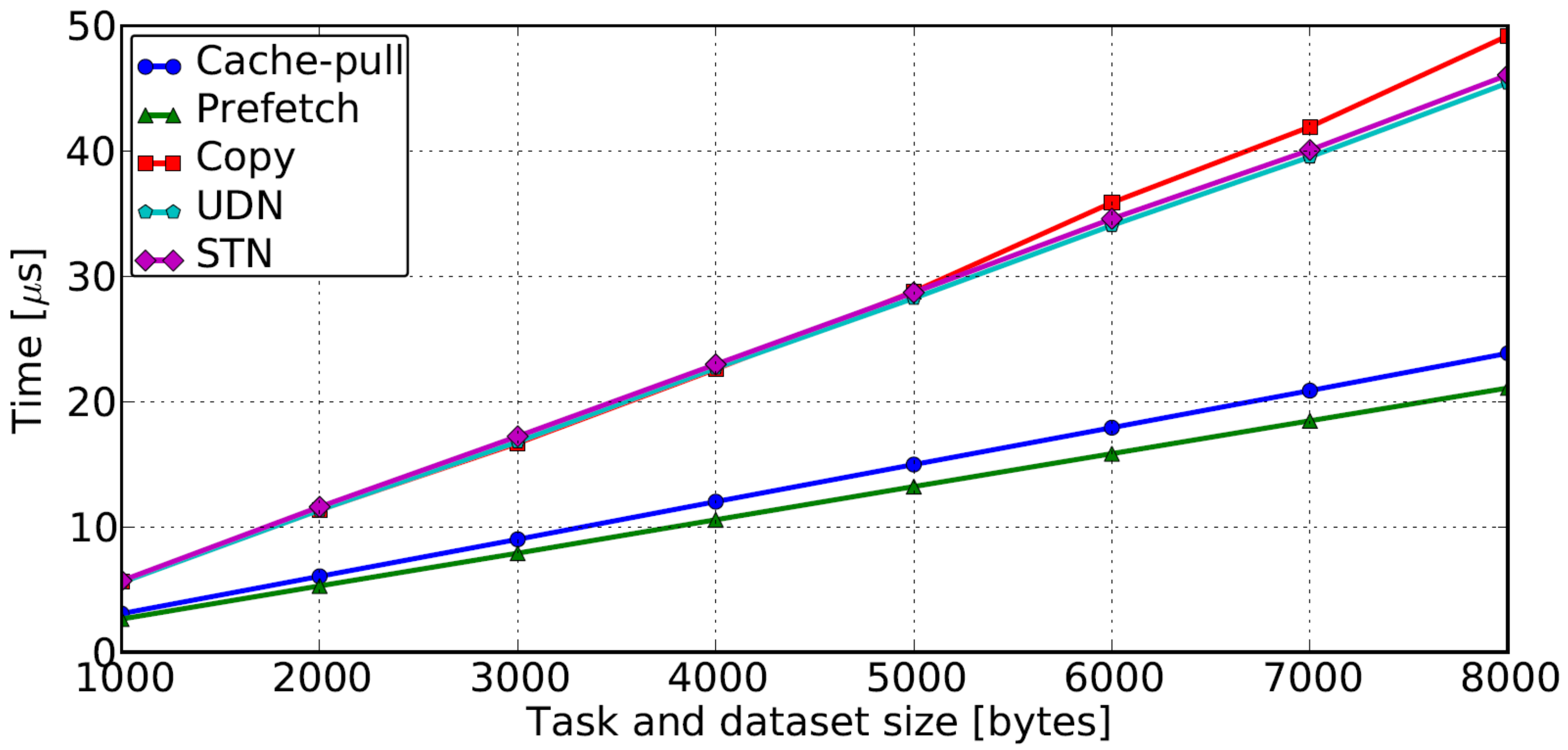
- Standard deviation < 0,2 ns
- Overhead is constant, 31 ns
- 10.000 iterations





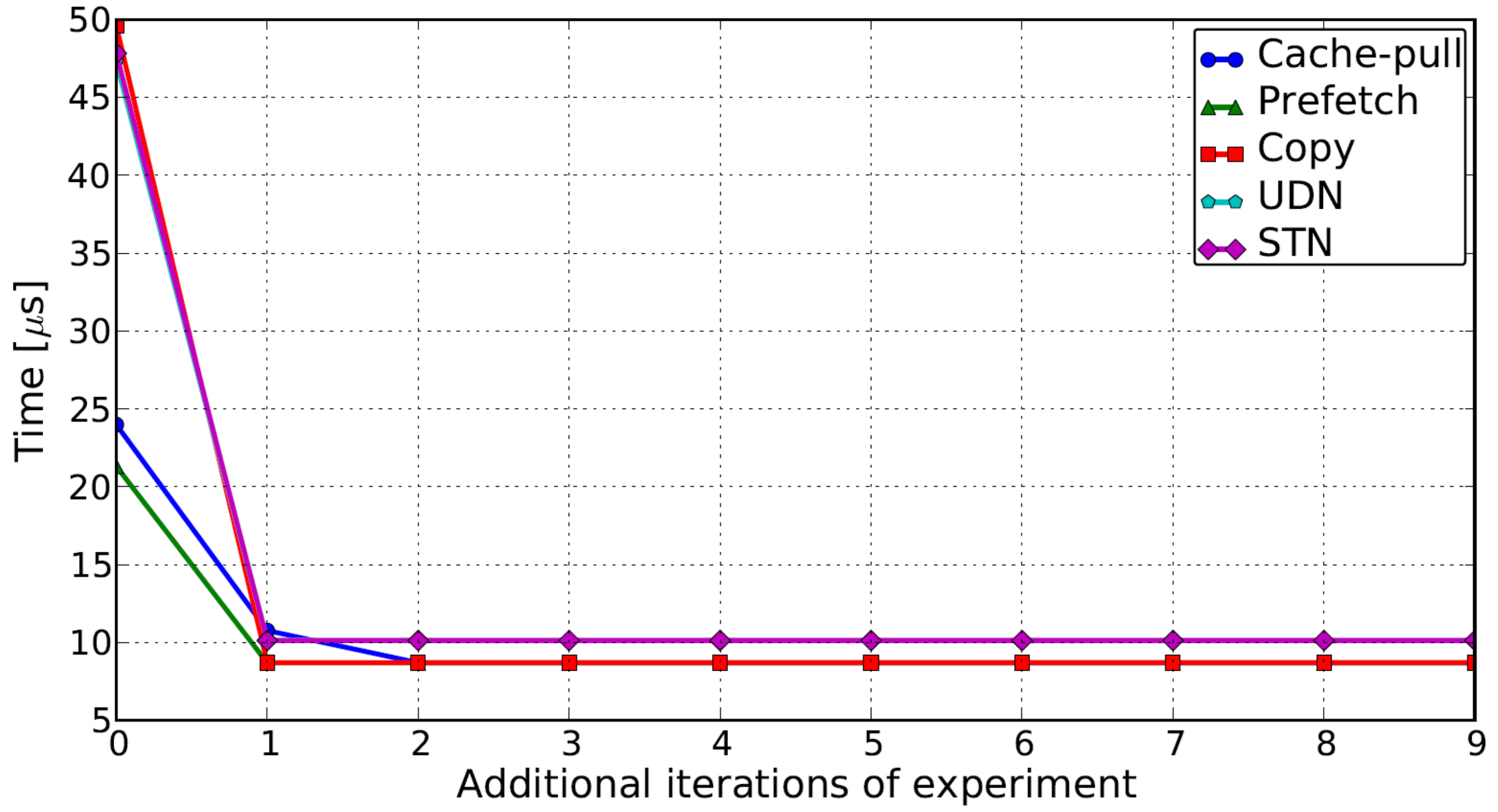
Fraunhofer FIRST

Results



– Linear growth: scalable
– Use of shared cache twice as fast: others need store instruction

Results 2/3



– Timing anomaly for cache-pull method shows unpredictability

Fraunhofer FIRST

Results

Analysis:

- All methods are scalable
- Prefetching
  - Fast
  - Deterministic, as opposed to cache-pull
- Maximum absolute deviation from mean is 0,4 µs
  - Upper bound on transfer time can be found
- Four out of five transfer methods have potential for deterministic task migration

IV. CONCLUSIONS

Conclusions 1/2

Trends and contribution:

- Account for on-chip interconnect:
  - Reservation of capacity
  - Guarantees on QoS
  - Isolation of traffic

- Extension of partitioning to many-cores
  - Mapping and scheduling of:
    - Tasks to cores
    - Traffic onto the NoC

- Mode-based and flexible partitions:
  - Optimize hardware usage
  - Requires deterministic task migration

Fraunhofer
FIRST

Conclusions 2/2

Experiments:

- Timing analysis of large coherent shared caches not feasible
  - Determinism can be achieved with cache locking and prefetching
  - Reservation and thus partitioning of memory networks difficult

- UDN slower, but can be partitioned in software
  - "Hardwall" enables traffic isolation
- STN naturally implements circuit switching
  - Requires programming of switches

- Combination of transfer methods: strong isolation

- Experiments show feasibility of concepts

END OF PRESENTATION