ExaNoDe Programming Environment to Exploit ARM, UNIMEM and FPGAs

Babis Chalios

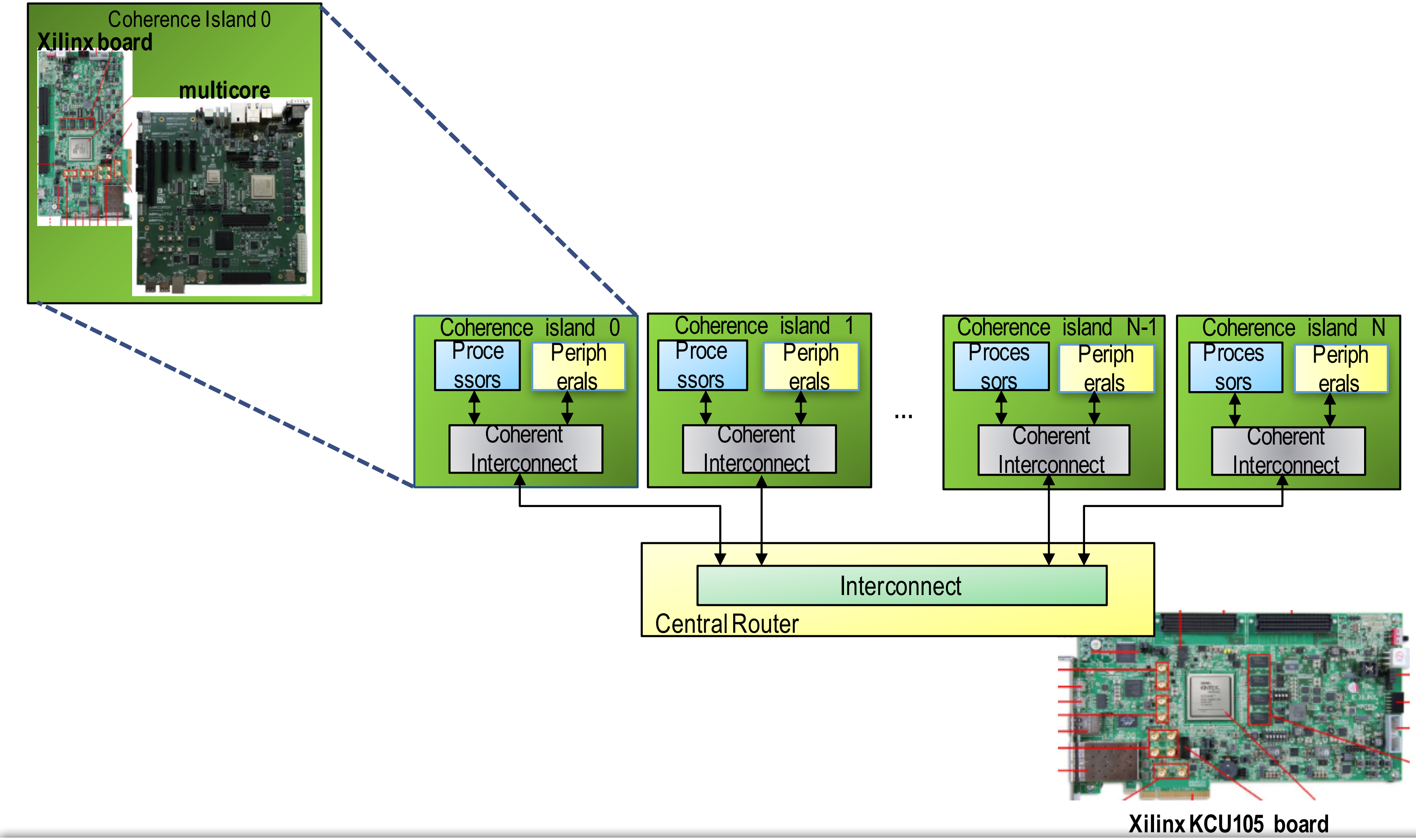
Barcelona Supercomputing Center

Tuesday 23 January 2018

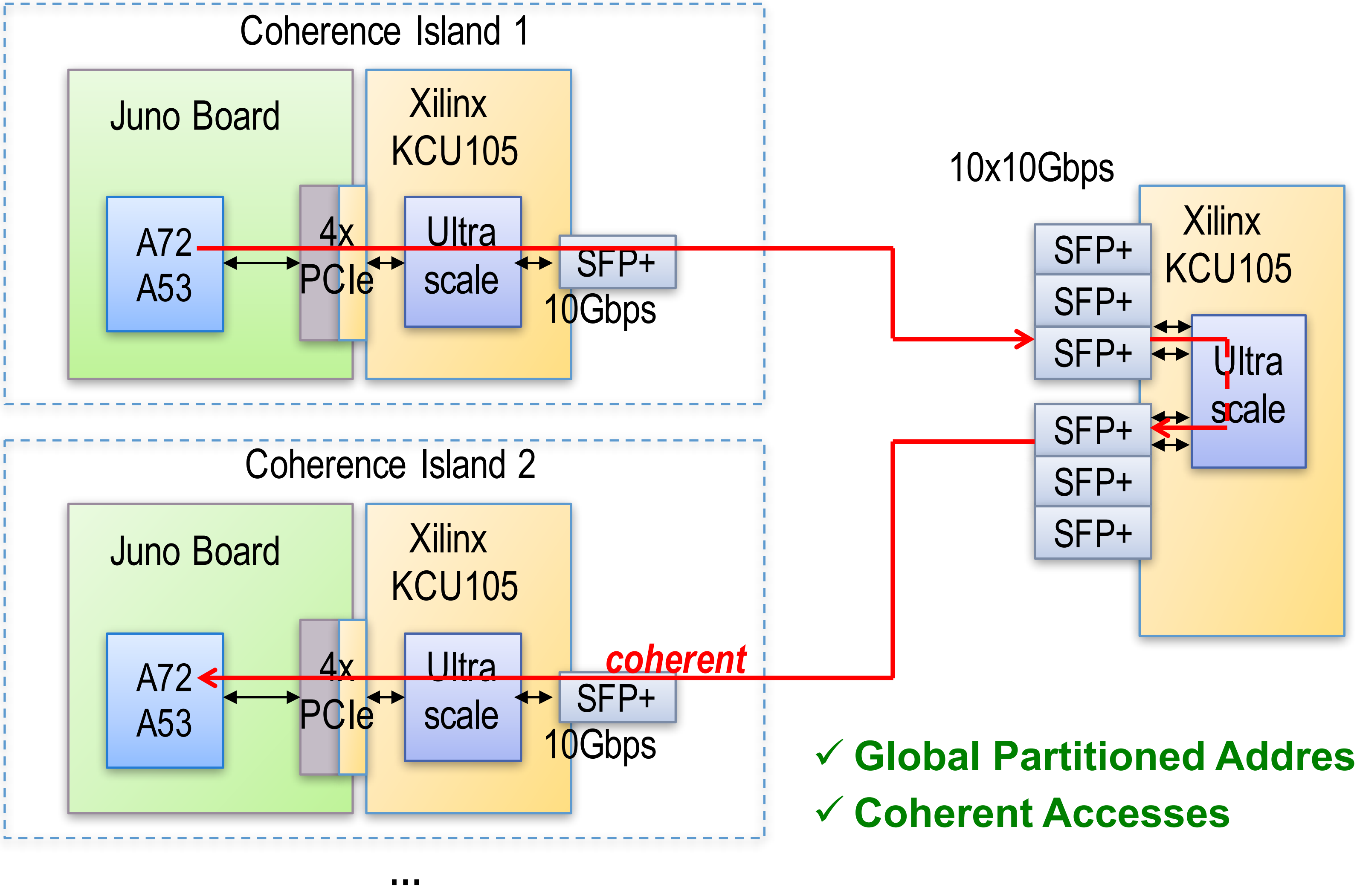
Disclaimer: This presentation does not represent the opinion of the EC and the EC is not responsible for any use that might be made of information appearing herein.

The UNIMEM architecture

T5.1: High-level Architecture of 64-bit DP



UNIMEM: Remote Coherent Memory Accesses

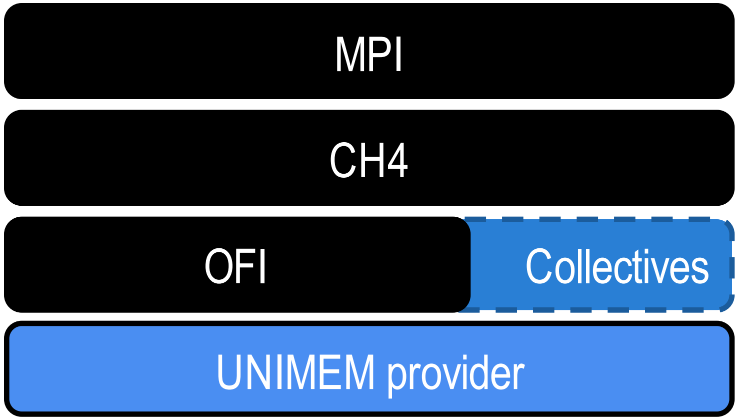


- ✓ **Global Partitioned Address Space**
- ✓ **Coherent Accesses**

ExaNode communication libraries

MPI over UNIMEM Design

- **Our approach**



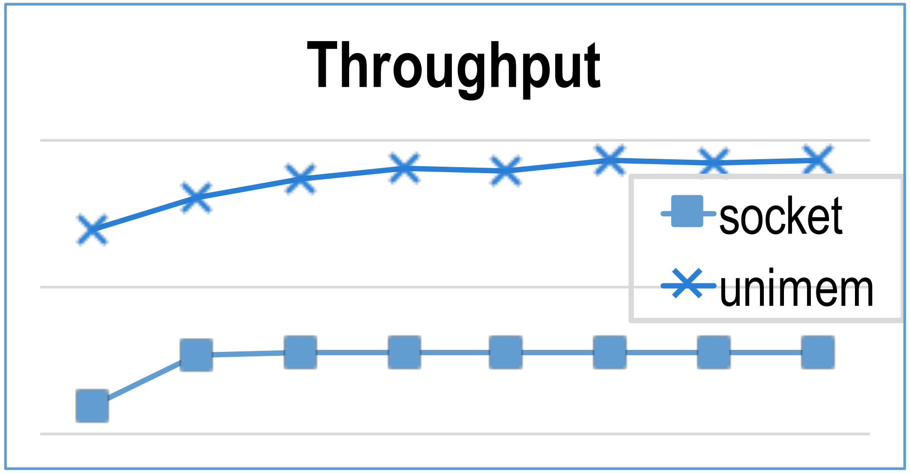
MPI

CH4

OFI

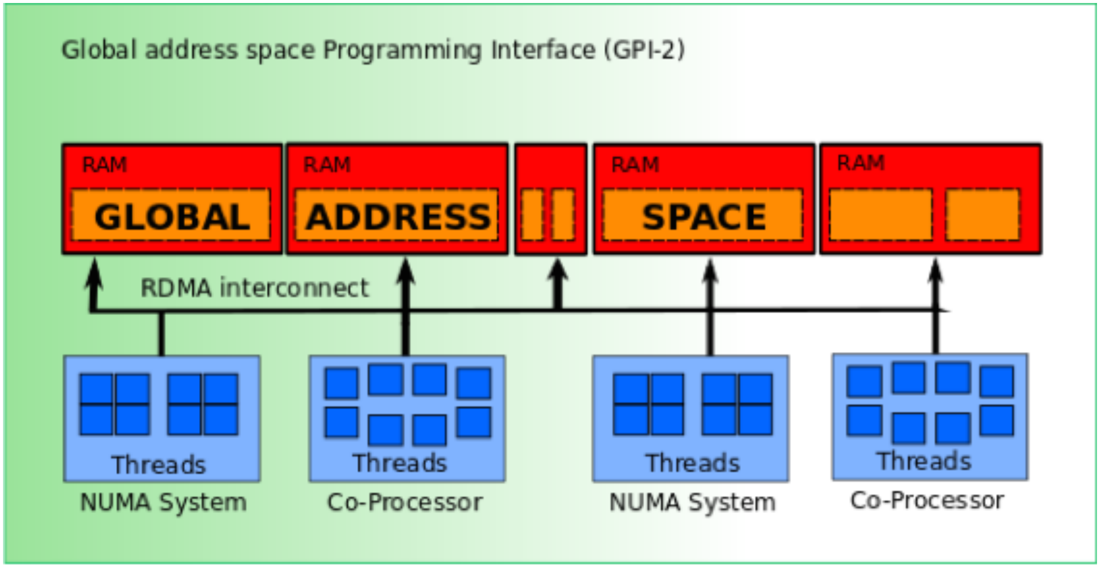
Collectives

UNIMEM provider



- **MPICH**
  - With its derivatives, default MPI in 9/10 top in TOP500
- **CH4**
  - Non-scalable structures restricted to non-scalable nets
  - Full communication semantics provided to networks
  - Shared memory improvement
  - Latency improvements
- **OFI / libfabric**
  - Designed to minimize mismatch between apps/libraries & comm. HW
- **Work in Progress**
  - Performance improvements using better RDMA support in UNIMEM

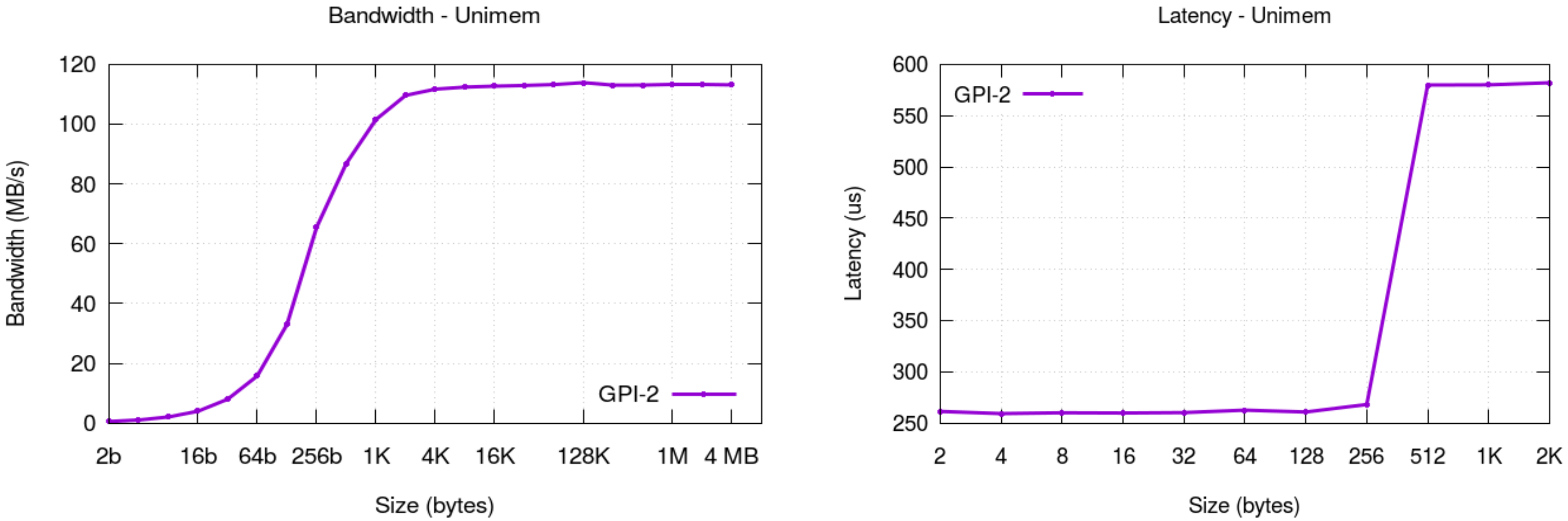
GPI-2 for ExaNode/UNIMEM



- GPI (Global Address Space Programming Interface): asynchronous communication library and programming model
- GPI combines the advantage of a global address space with the accumulated performance of separated memory subsystems
- GPI aims to initiate a paradigm shift from bulk-synchronous two-sided communication patterns towards an asynchronous communication and execution model
- GPI delivers the highest communication performance and scalability on all RDMA-Networks available today

GPI-2 for ExaNode/UNIMEM

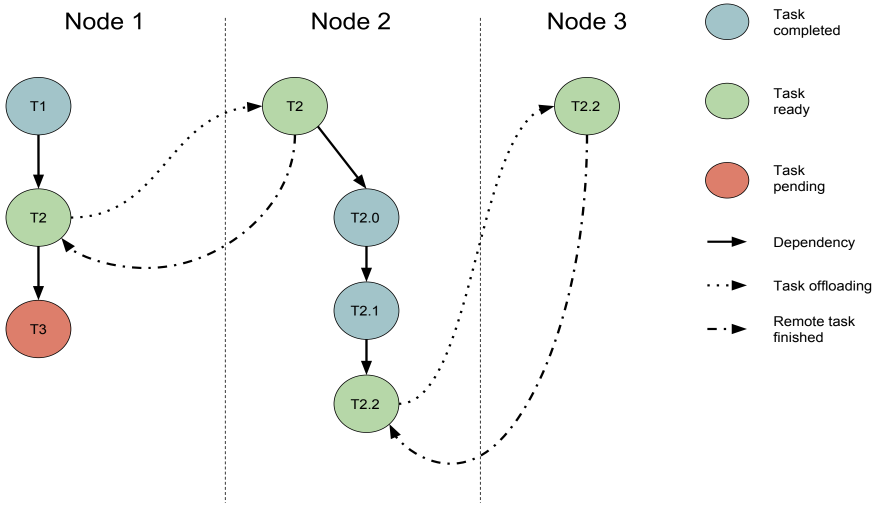
- Within the ExaNode project most of the GPI Modules were ported to UNIMEM
- Different performance limitations were reported when using RDMA in UNIMEM
- GPI-2/GASPI developments are currently supported on an UNIMEM Emulation Framework (UniEF) as well as on Socket over UNIMEM
- Early performance characteristics of UNIMEM-Sockets are available (see below)

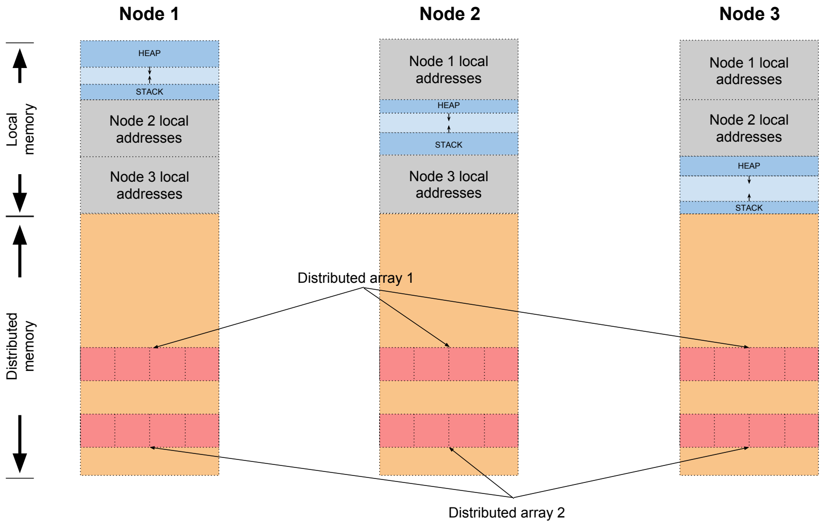


Performance results of two Trenz-Prototype-Boards@Forth connected via UNIMEM

ExaNode programming models

OmpSs for distributed memory systems



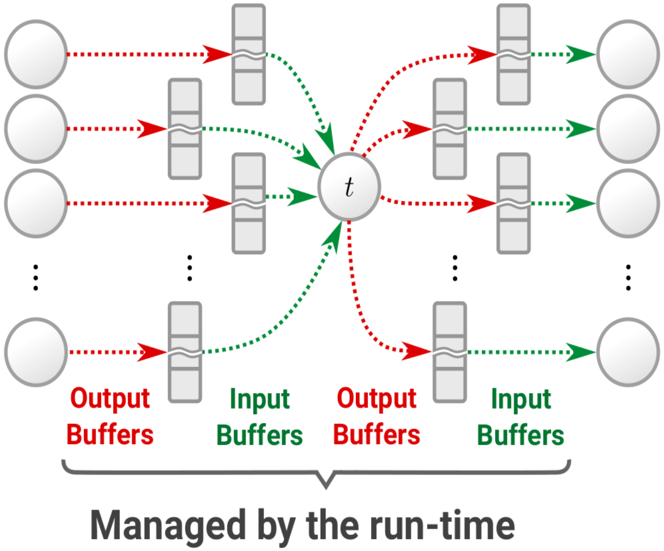


- ## Task-based parallel programming model
  - Parallelism defined through **task** constructs
  - Synchronization between tasks using data **dependencies**
- ## Single global virtual address space abstraction
  - No need for explicit memory transfers
  - Programmer focuses in algorithm and parallelism design
- ## Runtime support for physically distributed memory systems
  - The run-time system is responsible for memory transfers across cluster nodes
  - Scheduling based on locality and load-balancing
  - Opportunities for run-time optimizations for irregular parallelism.

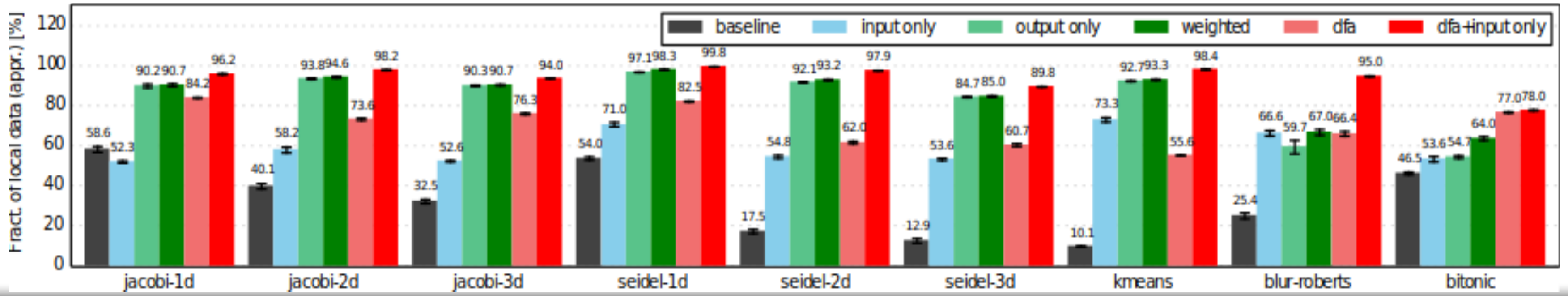
OmpSs for distributed memory systems

- **Current status**
  - Support of distributed arrays
  - Scheduling based on locality of task data
  - Communication layer independent of underlying library
    - Current implementation based on MPI
  - Release of beta version by the end of the month
- **Work in progress**
  - Performance profiling based on kernels and mini-apps
  - Improvements on scheduling policies
- **Future work**
  - Integration with UNIMEM-capable MPI
  - Implement support for offloading tasks to FPGAs using OpenCL

OpenStream on UNIMEM



- **Task data-flow programming model**
  - Express task-dependent parallelism
  - Implicit privatization of data
  - Runtime has full control over data management
- **Uniform, shared memory abstraction is preserved for programmers**
  - No need to explicit data placement or transfer
  - No need to customize parallelization to the topology of the system
- **Dynamic work and data management**
  - Load balancing through work-stealing
  - Data locality optimized by work-pushing
  - Communication mapped to UNIMEM RDMA and overlapped with computation



Thank you!

ExaNoDe

European Exascale Processor & Memory Node Design

www.exanode.eu