A Remote Memory Access Infrastructure for Global Address Space Programming Models in FPGAs

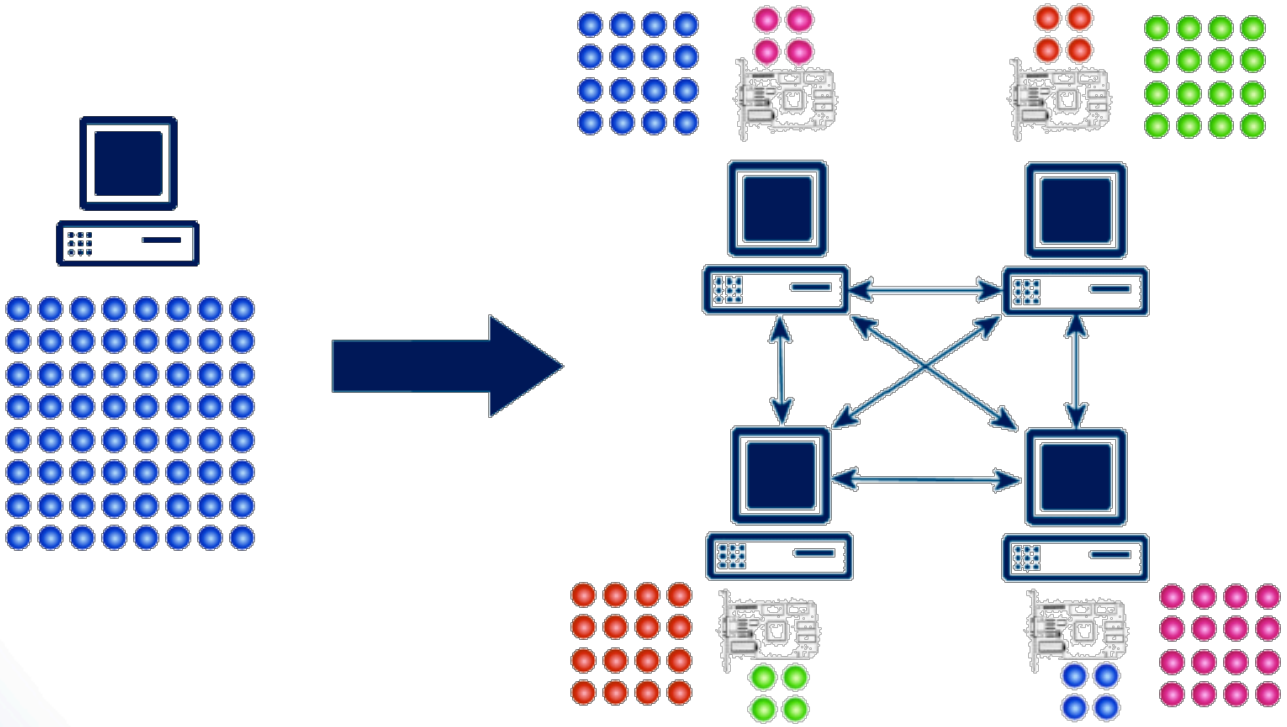
Ruediger Willenberg and Paul Chow

High-Performance Reconfigurable Computing Group
University of Toronto

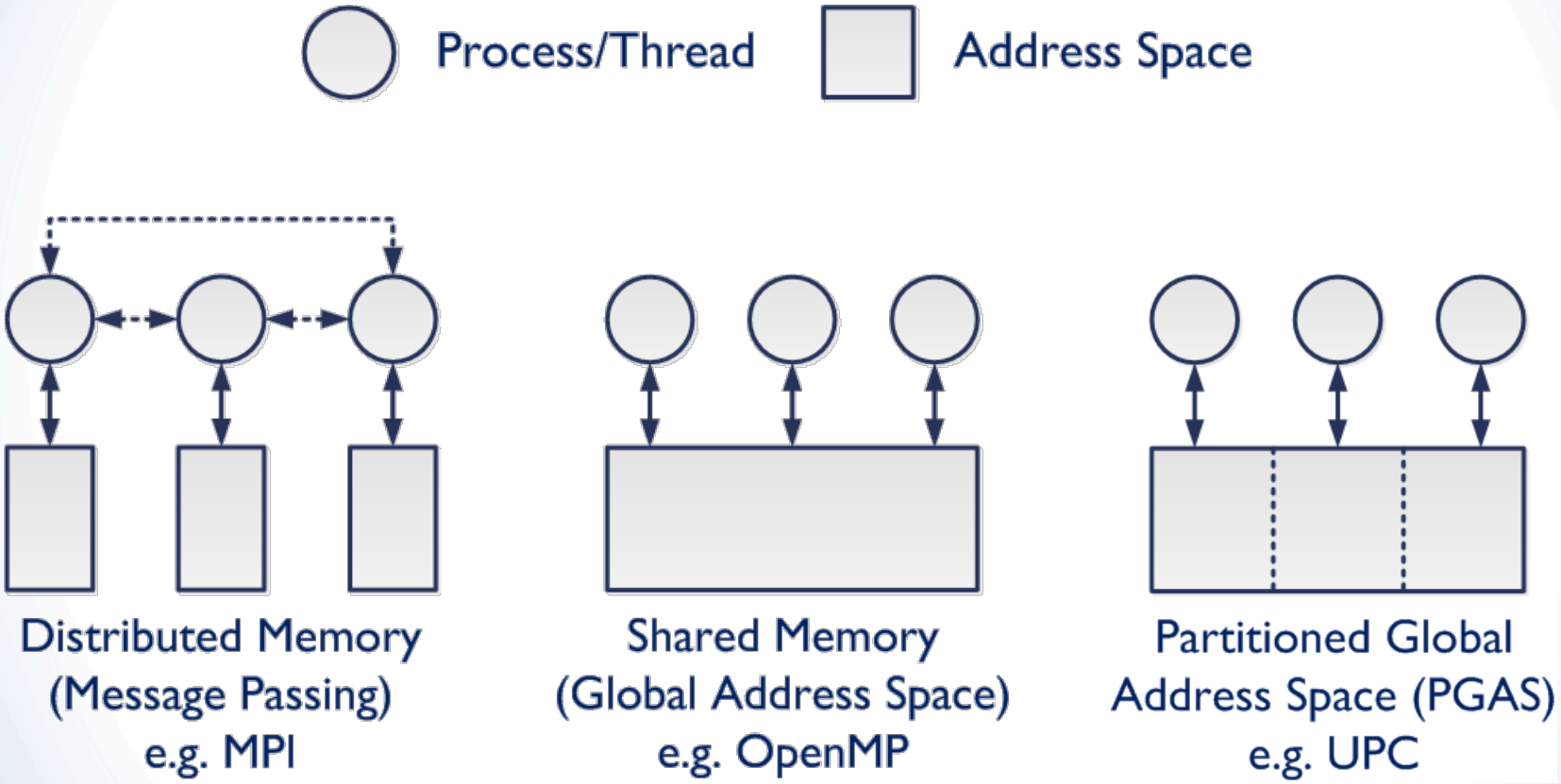
March 16, 2013

Parallelizing Computation

- How to partition and communicate data
- How to synchronize computation and data access



Parallel Programming Models

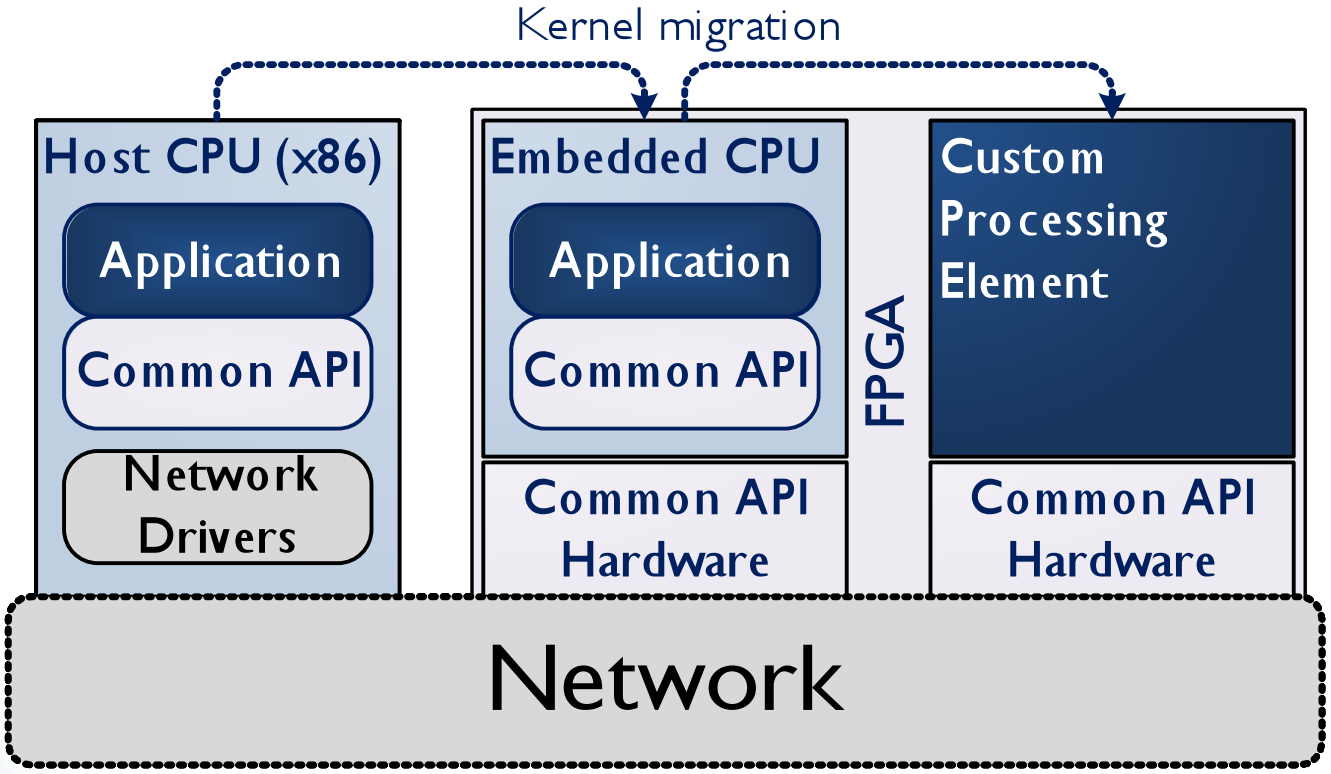


Partitioned Global Address Space

- Shared Memory access, but…
- …visible difference between local and remote data
- PGAS allows one-sided communication
- Many PGAS implementations
  - Modified standard languages (Unified Parallel C)
  - New parallel languages (Chapel)
  - Application libraries (Global Arrays)

Our programming model philosophy

- Use a common API for Software and Hardware



Common SW/HW API

- CPU and FPGA components can initiate data transfers
- SW and HW components use similar call formats
- For distributed memory and message-passing, this was implemented by TMD-MPI

- <u>Our contribution:</u>
  Building hardware infrastructure for a common API for PGAS

Rationale

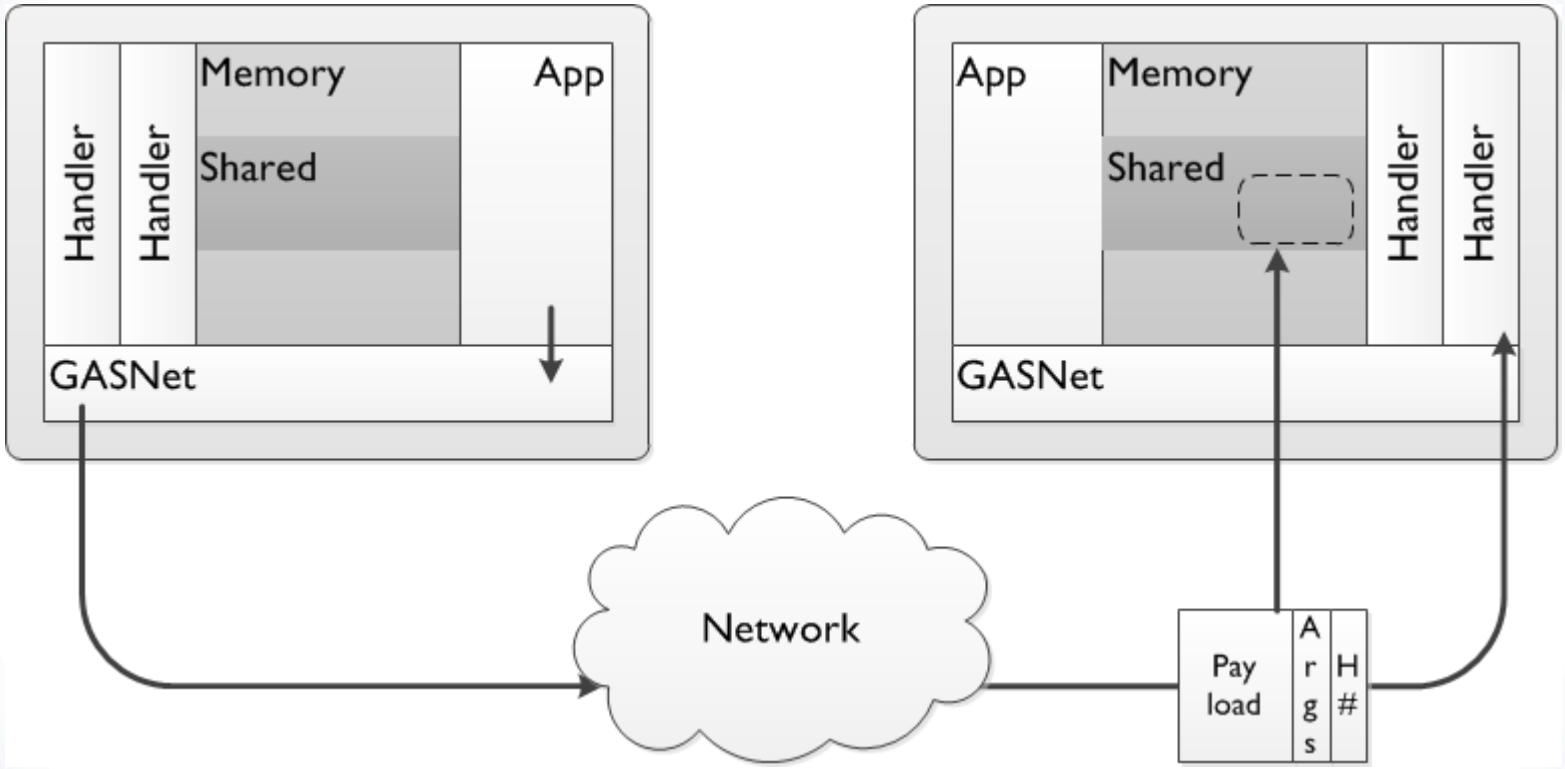
- Why a common API ?
  - Easier development: SW Prototyping → Migration
  - Model makes no distinction between CPUs and FPGAs
  - FPGA-initiated communication benefits performance

- Why PGAS instead of Message-passing ?
  - Higher programming productivity, easier maintenance
  - Performance benefits through one-sided communication
  - Shared memory more suitable for certain applications
    - Graph-based algorithms, pointer chasing

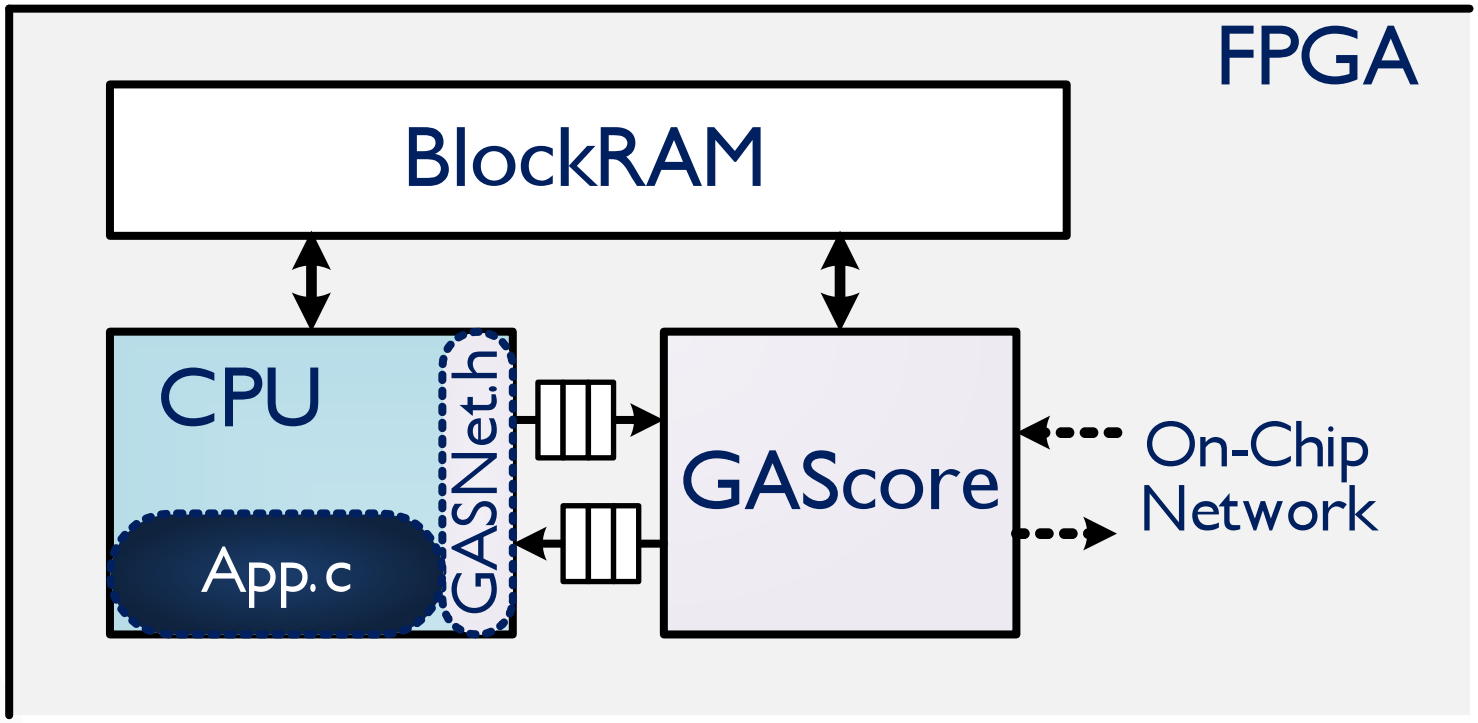
GASNet instead of MPI

- **G**lobal **A**ddress **S**pace **Net**working
- Software communication library for (P)GAS
- Developed as a communication layer for UPC, Co-Array Fortran and Titanium (Java)
- GASNet's core API is built on the concept of "Active Messages"

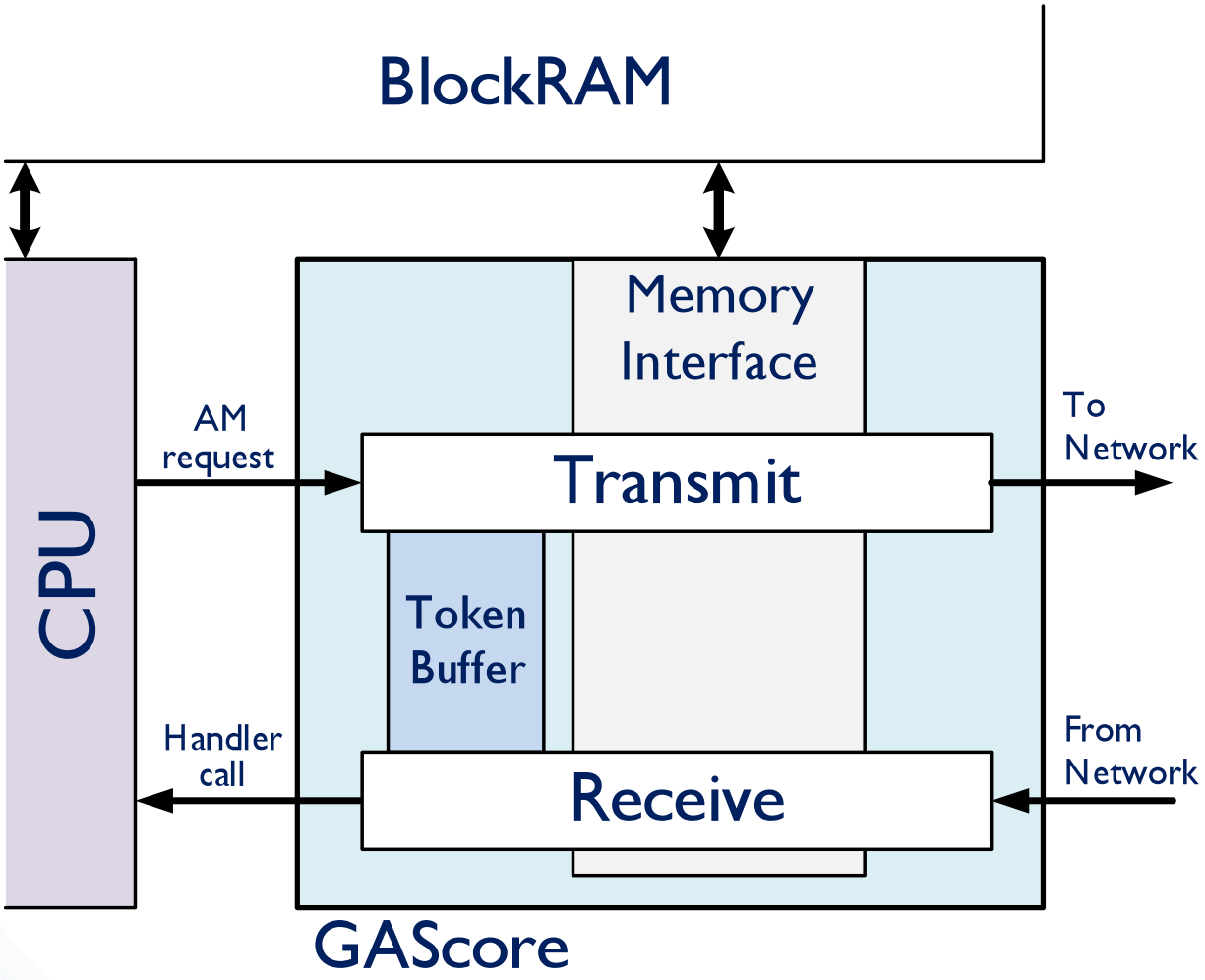
GASNet Active Messages



GASNet on FPGA: GAScore



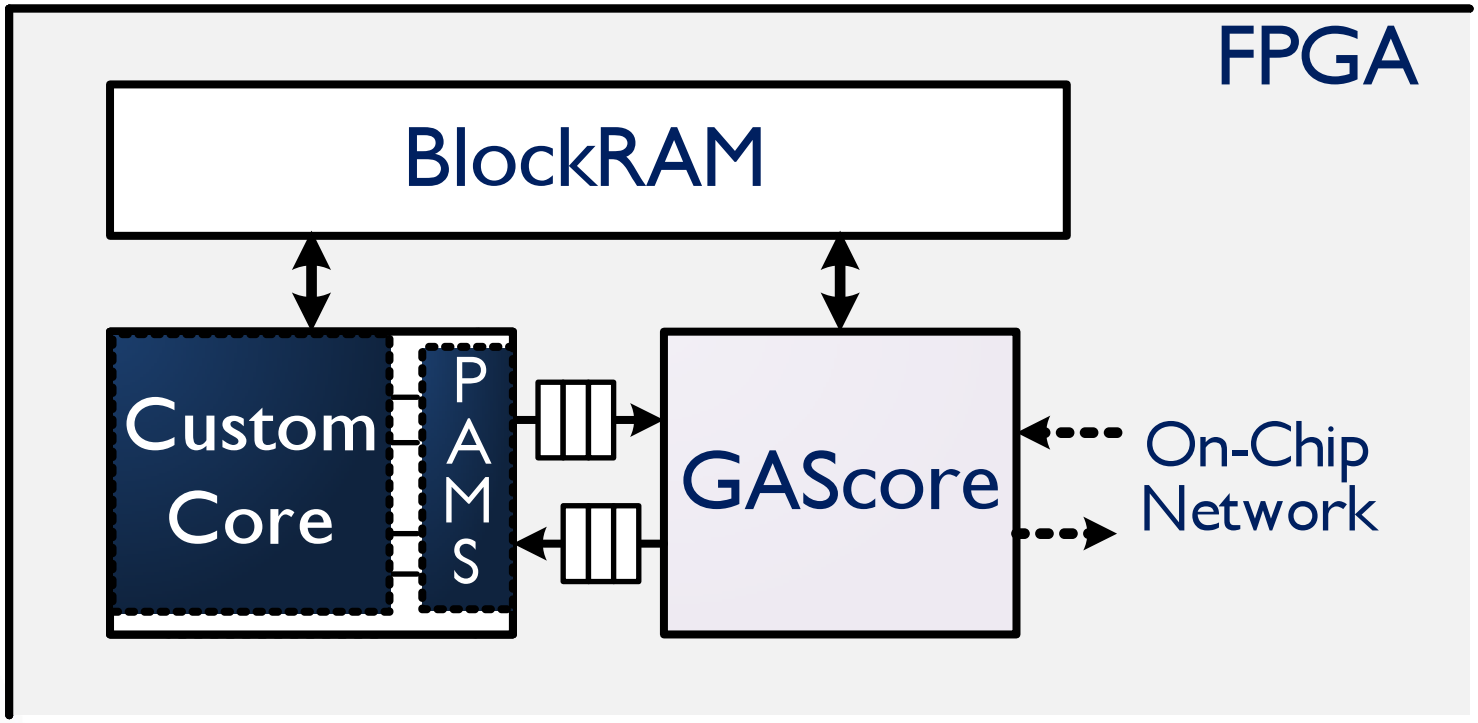
GAScore


BlockRAM
CPU
Memory Interface
AM request
Transmit
To Network
Token Buffer
Handler call
Receive
From Network
GAScore

GAScore

- Remote memory communication engine (RDMA)
- Controlled through FIFOs (FSLs)
- Configuration parameters are the same as for GASNet Active Message function calls
  - Simple software layer for embedded CPUs
- Independent receive/transmit channels
  - No deadlocks

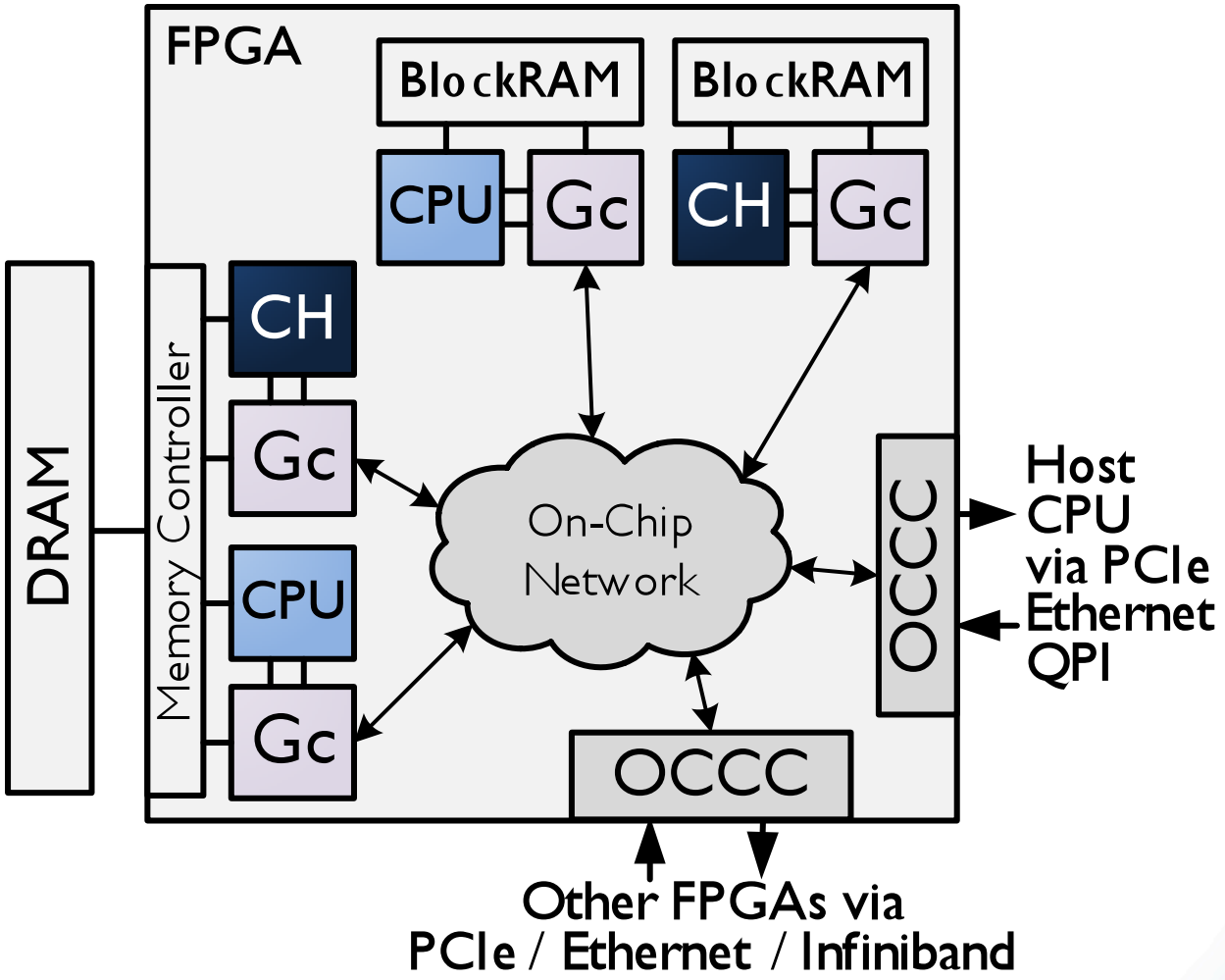
GASNet on FPGA: GAScore



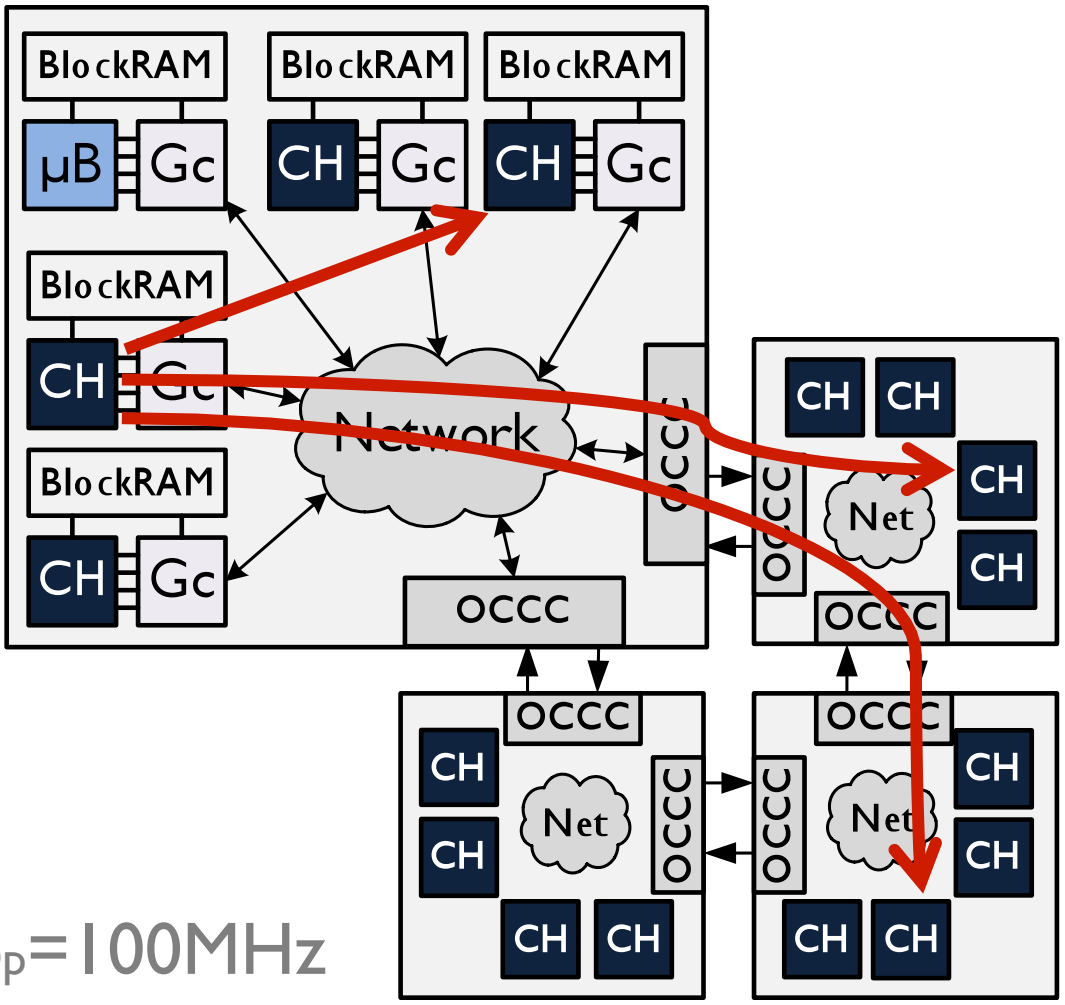
Programmable Active Message Sequencer

- Controls Custom Hardware core
- Handles reception/transmission of Active Messages
- Custom core operations and Active Messages are initiated based on:
  - Custom Hardware state
  - Number of received messages with a specific code
  - Amount of received data
  - Timer
- (Re-)programmable through Active Messages

On-chip GASNet system



BEE3 multi-FPGA platform



f_{OP}=100MHz

Active Message latencies

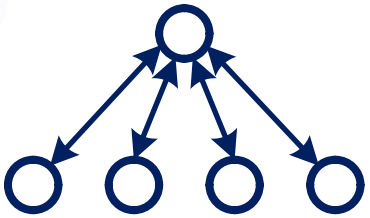
Short message latency (cycles)

FPGA hops	1-way	2-way
0	17	35
1	24	49
2	31	63

1-word memory transfer latency (cycles)

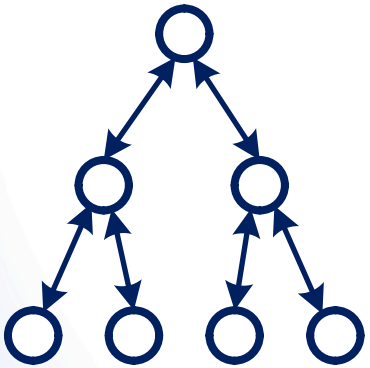
FPGA hops	1-way (remote write)	2-way (remote read)
0	29	47
1	36	61
2	43	75

Barrier latencies



Simple barrier latency (cycles)

Latency to node 0	87
Latency to node 0 and back	196



"Staggered" barrier latency (cycles)

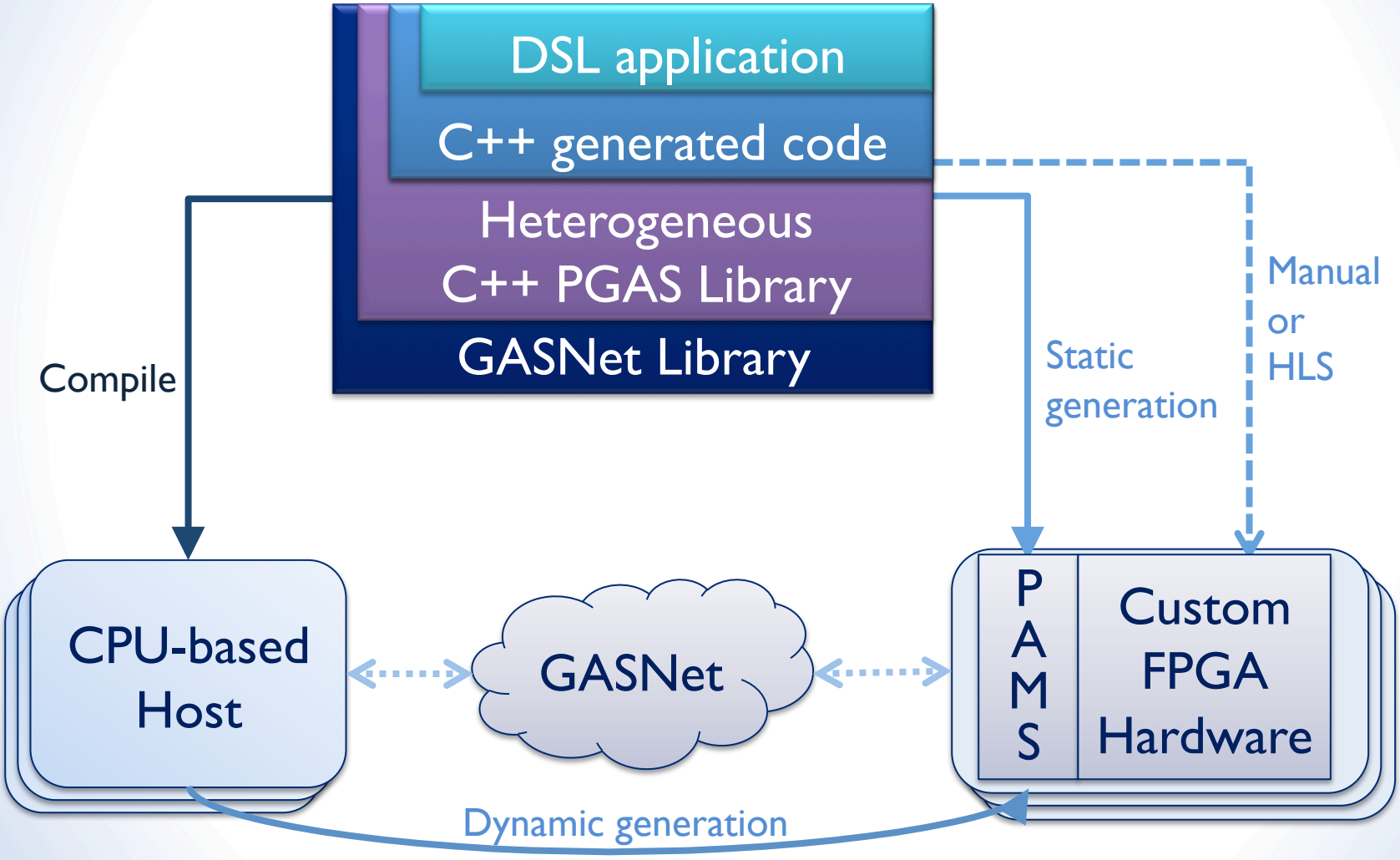
Latency to node 0	62
Latency to node 0 and back	148

Future Work

- Hardware components:
  - External memory support
  - Strided/vectored memory accesses
  - Host interface (PCIe, QPI, HyperTransport)
  - Latency/bandwidth optimizations

- Software components:
  - GASNet integration on CPU side
  - Application benchmarks
  - PGAS C++ library for heterogeneous systems

Heterogeneous C++ PGAS library



Thank you for your attention!

Questions ?

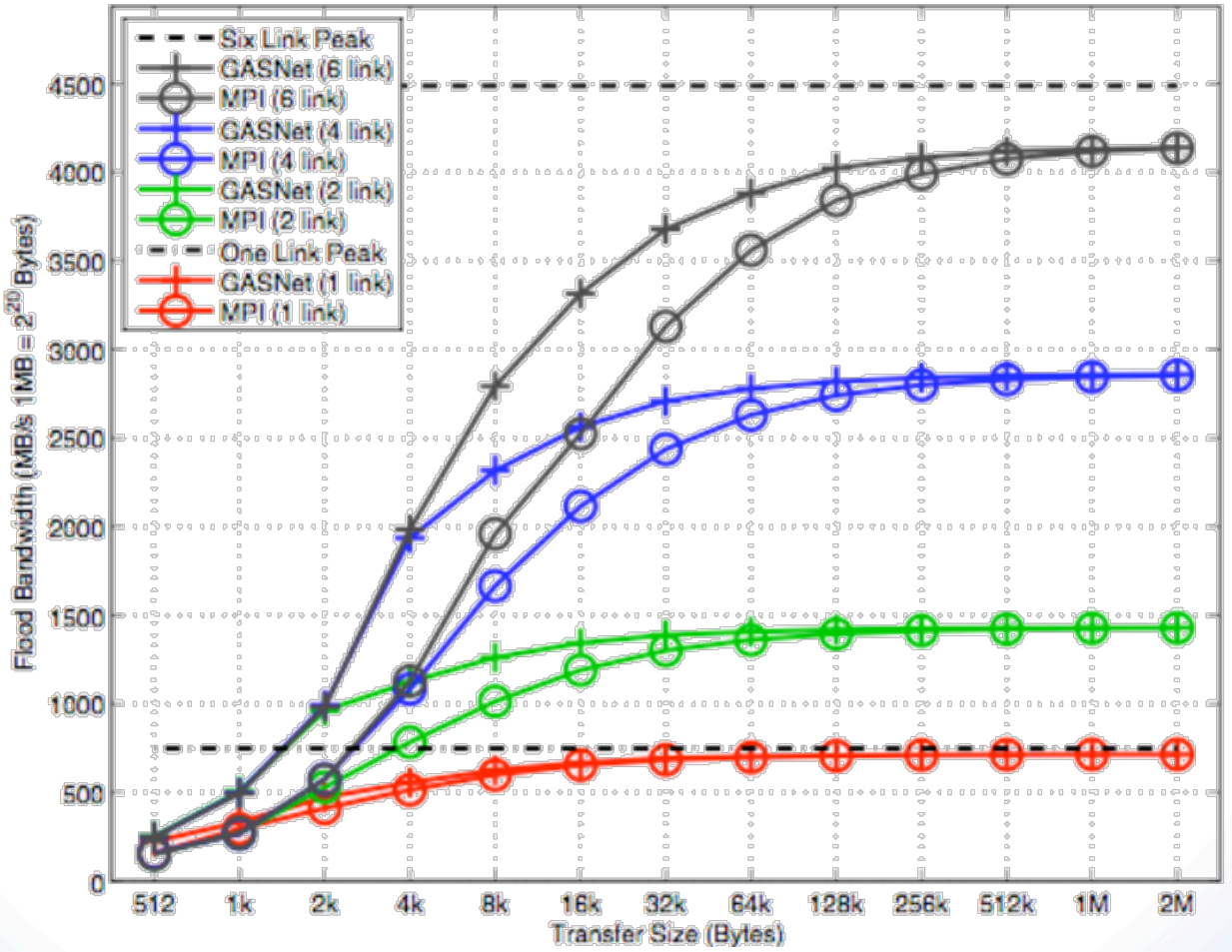
Backup slides

Short message - latency distribution

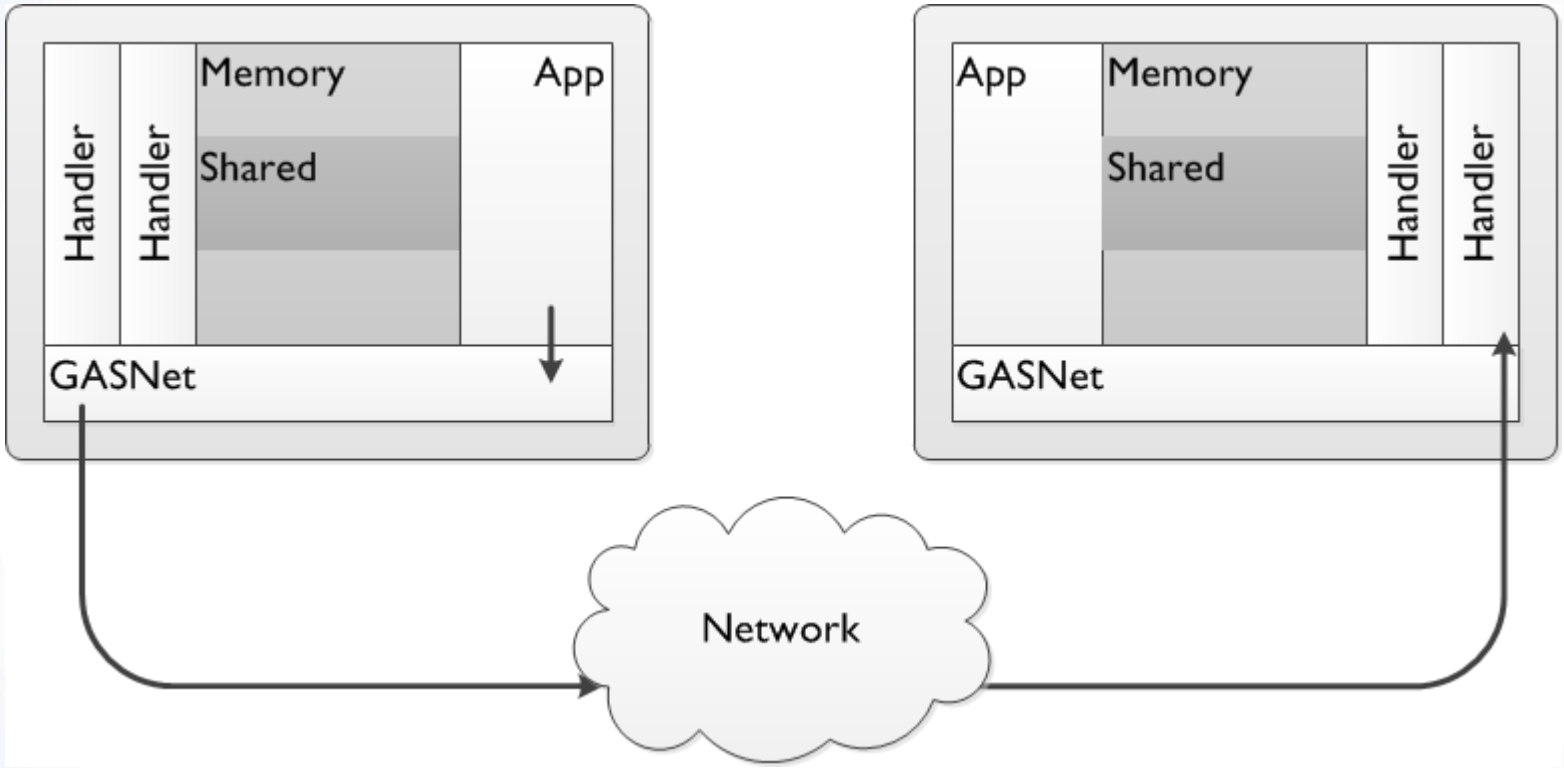
PAMS A start, tx	1 cycle
FIFO	1 cycle
GAScore A tx	3 cycles
FIFO	1 cycle
NetIf A	1 cycle
FIFO	1 cycle
NetIf B	1 cycle
FIFO	1 cycle
GAScore B rx	4 cycles
FIFO	1 cycle
PAMS B rx, store time	2 cycles

Example 1-sided vs. 2-sided comm.

GASNet vs. MPI Bandwidth on BG/P

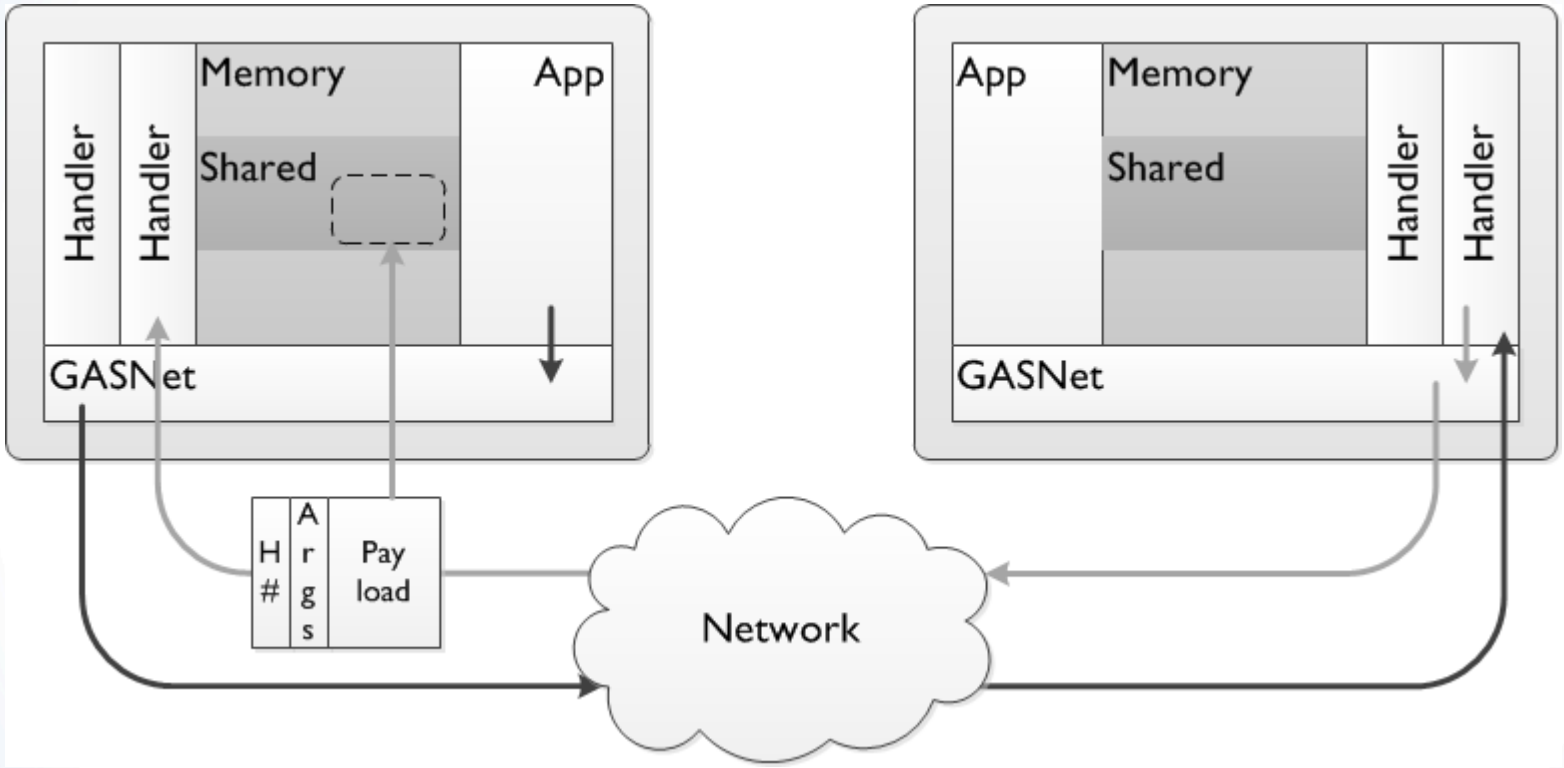


GASNet Active Messages



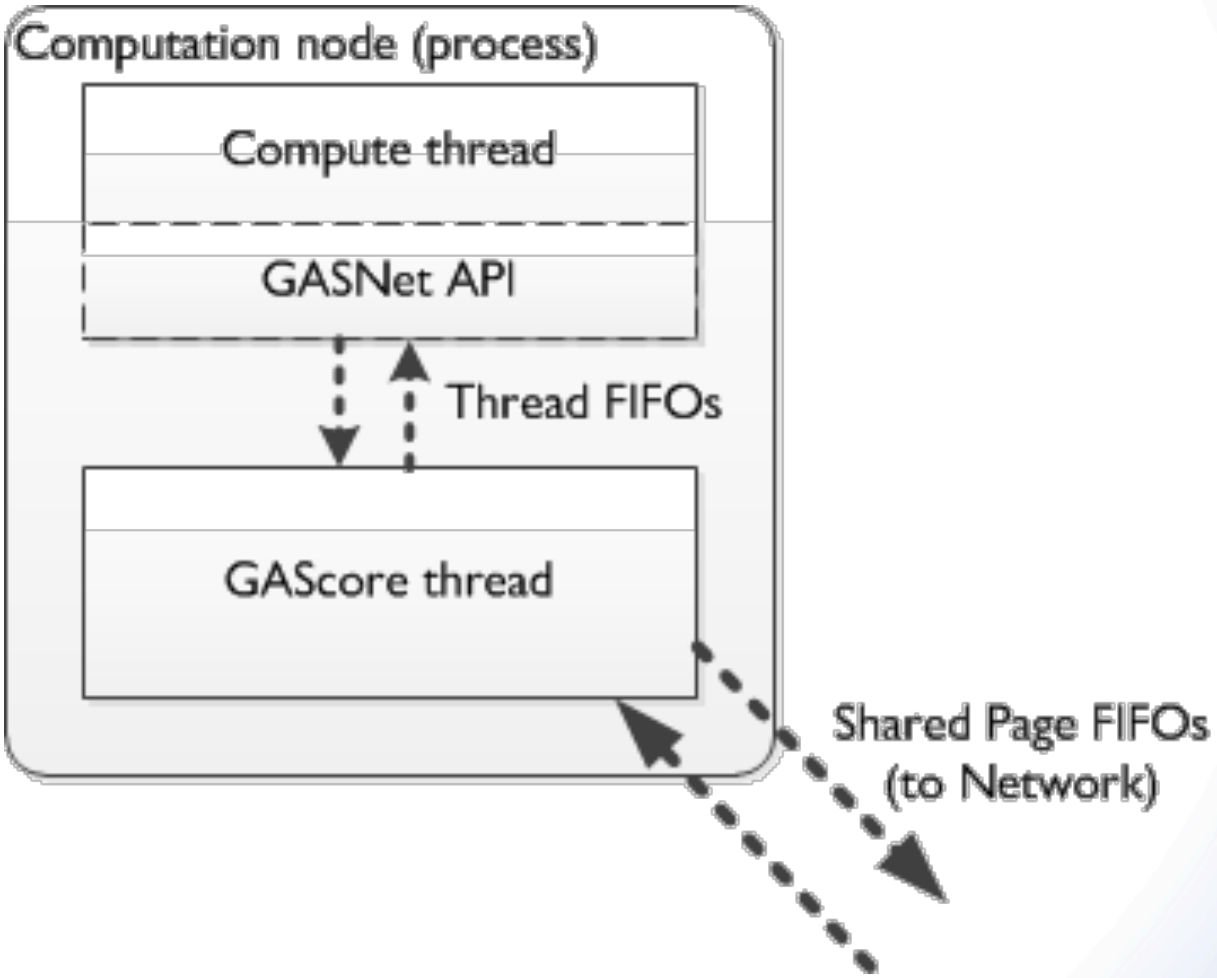
Medium Request Message

GASNet Active Messages

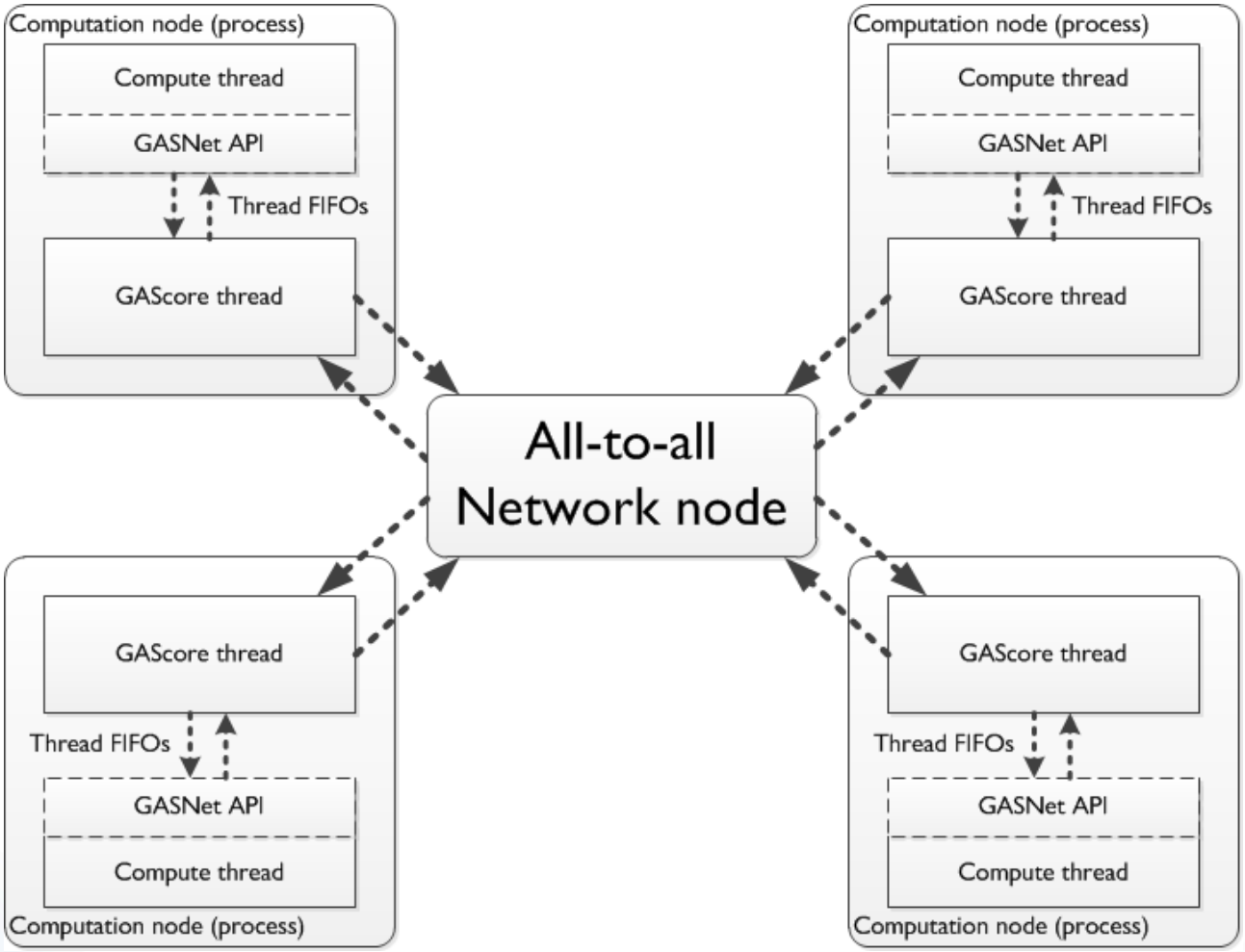


Long Reply Message

Software simulation

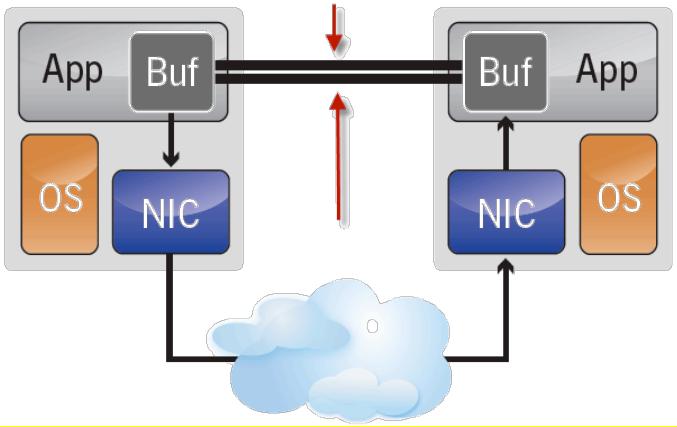


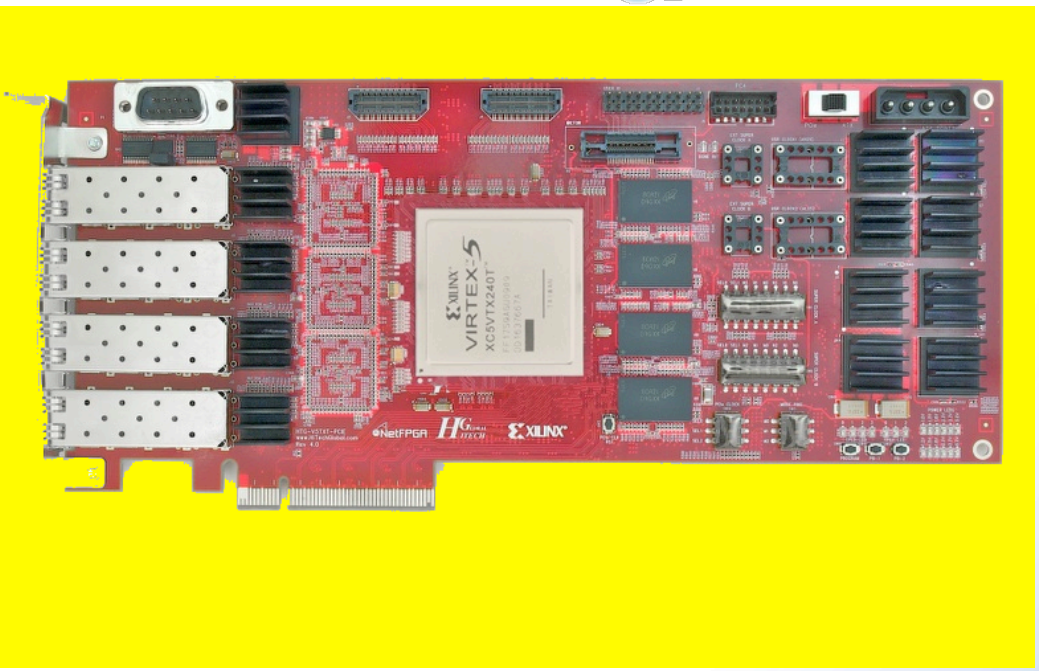
Software simulation (2)



NetFPGA10G

- 4x 10Gbit Ethernet
- PCIe 8x
- Virtex5-TX240T
- 288MB RLD-RAM
- 27MB QDRII-RAM



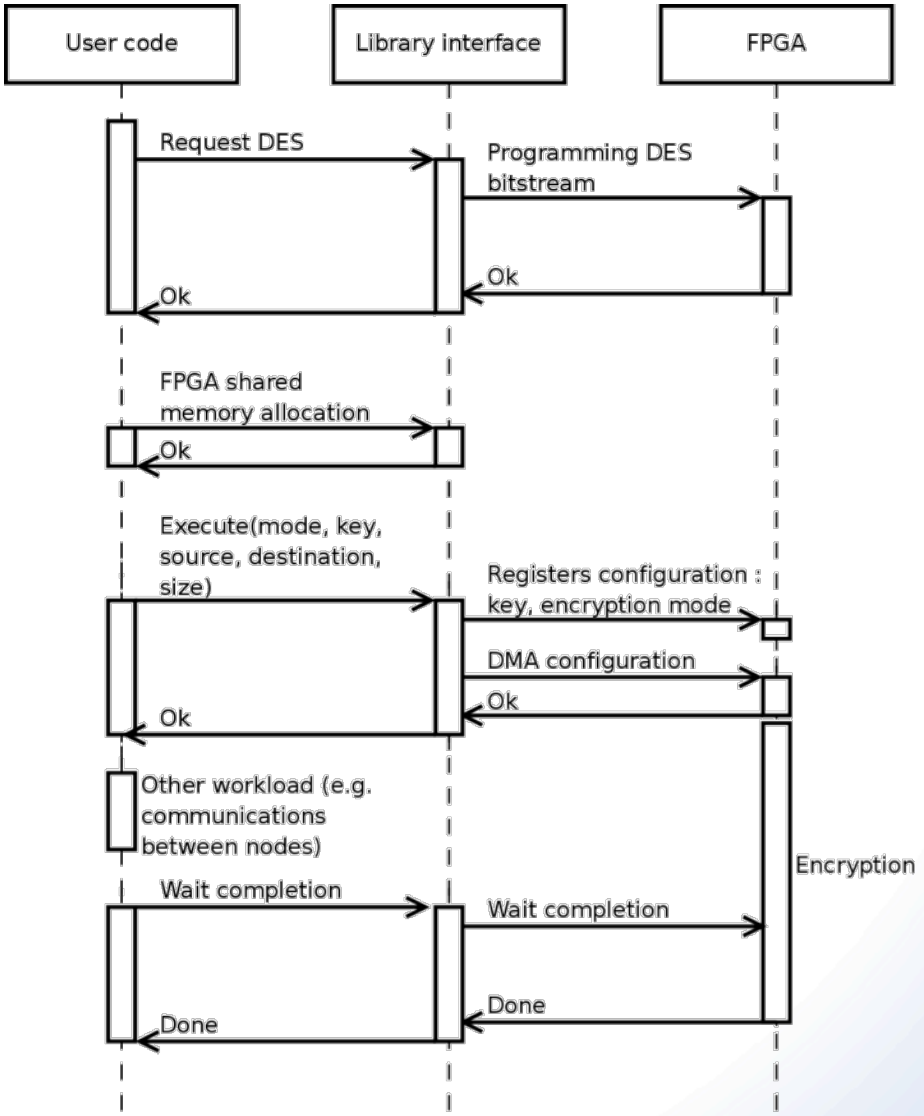


RDMA on NetFPGA10G?

- RDMA right now:
  - HW: Infiniband or iWARP Ethernet, soon RoCE
  - SW: OpenFabrics RDMA software stack
- NetFPGA-based RoCE would mean FPGA-style flexibility for experimentation:
  - Routing with up to 4 neighbours could be included
  - Higher-level RDMA/GASNet functions could be included on-chip: HPC hardware support
  - Possibly easier PCI-to-PCI access for FPGA cards

GWU: Parallel Programming of HPRCs with UPC

- Approach 1:
  IP Core Library



GWU: Parallel Programming of HPRCs with UPC

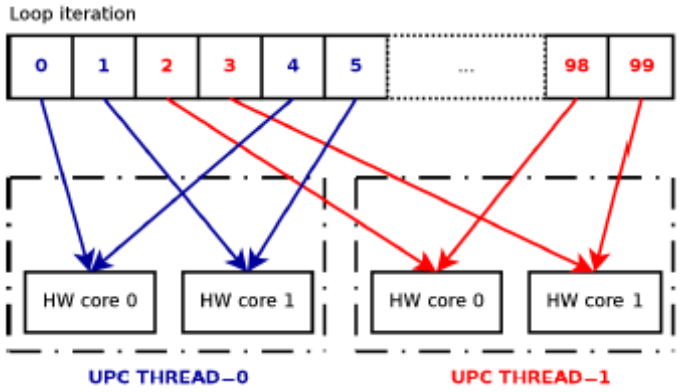
- Approach 2: forall extension -> Impulse C

  Extending the upc_forall semantic

  - Nesting of upc_forall represents multi-level parallelism
  - Will still be valid UPC programs.

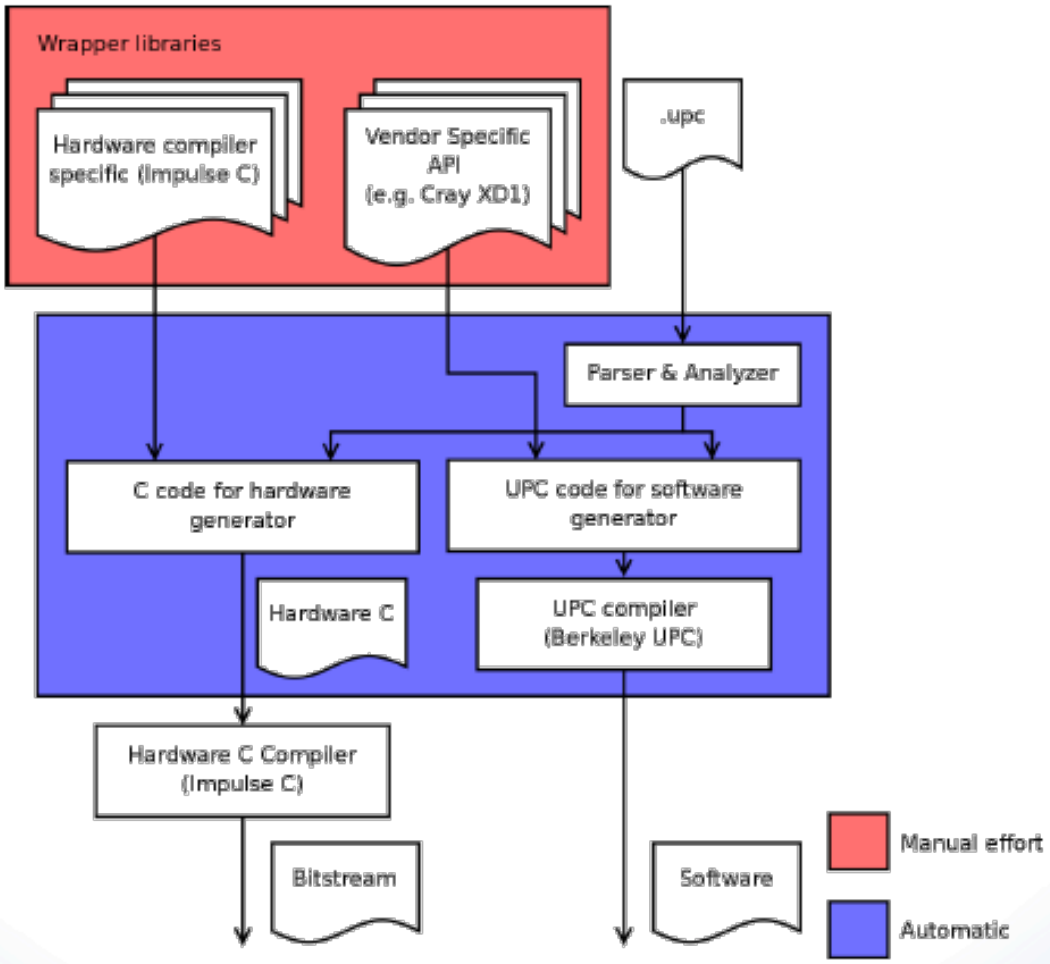
  Work distribution

  - upc_forall(int i=0; i < 100; i++; i/2)



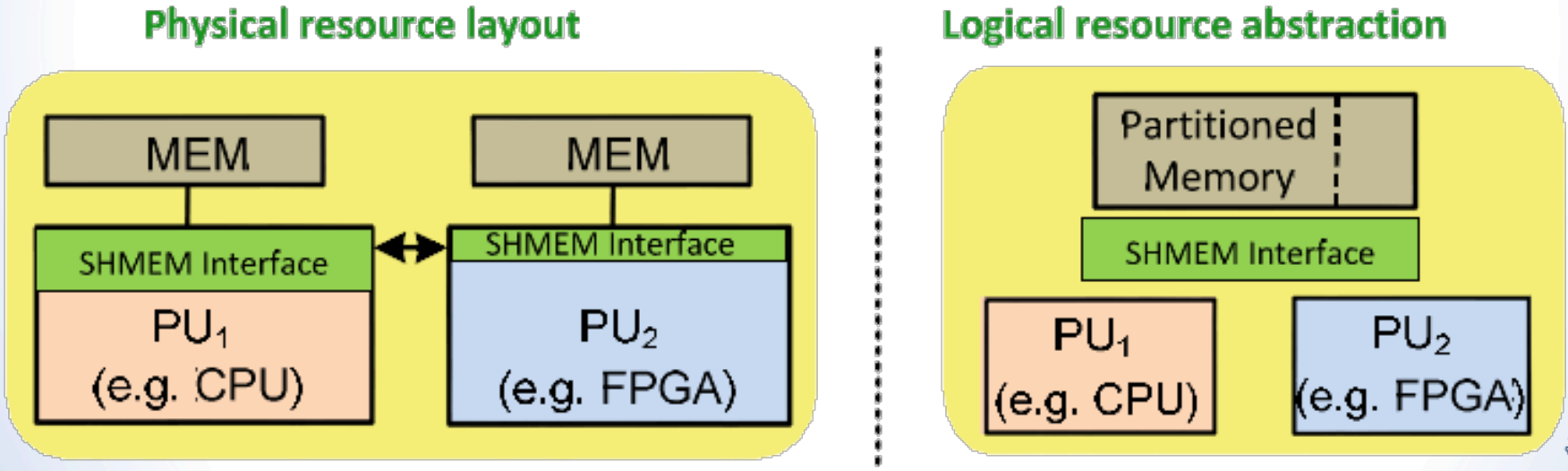
GWU: Parallel Programming of HPRCs with UPC

- Approach 2: forall extension -> Impulse C



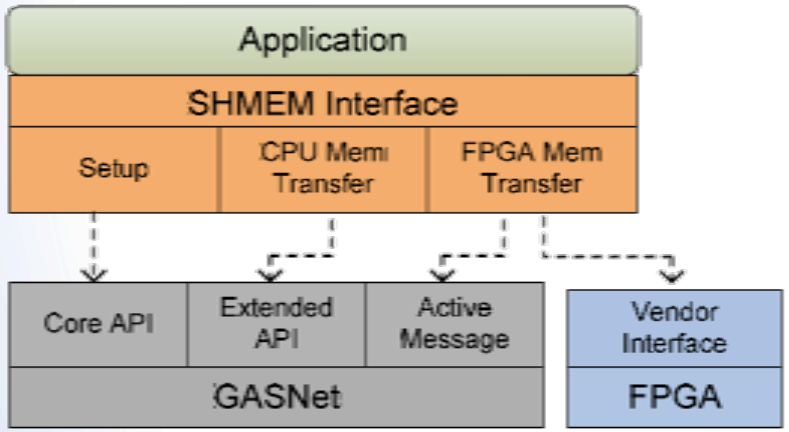
UF: Multilevel PGAS and SHMEM+

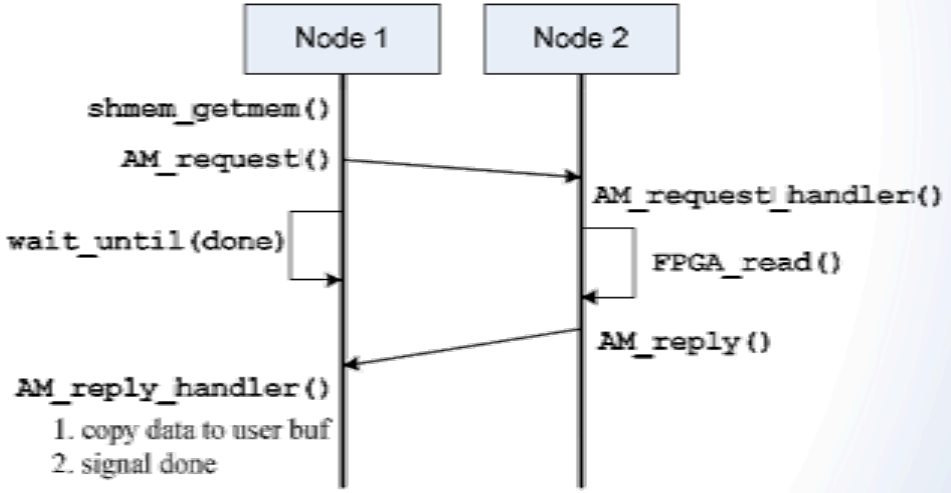
- Leaves building a contiguous local address space to the platform designer (Multi-level doesn't exist for the application writer)



UF: Multilevel PGAS and SHMEM+

- SHMEM library distributes put, get calls to the correct node and hardware





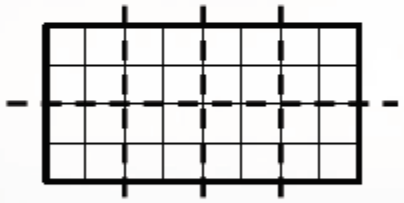
Chapel Distributions

Distributing a Domain

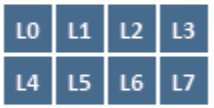
Domains are associated to a distribution

```
const Dist = new Block(rank=2, bbox=[1..4, 1..8]);

var Dom: domain(2) distributed Dist = [1..4, 1..8];
```



distributed over



The distribution defines:

- Ownership of domain indices and array elements
- Default distribution of work (task-to-locale map)
  E.g., forall loops over distributed domains/arrays

Why PGAS, not shared memory?

- Doesn‘t represent architectural reality well
- FPGA platforms will have non-uniform memory (BEE3), x86 goes NUMA with QPI