Efficient Hardware Support for the Partitioned Global Address Space

Holger Fröning and Heiner Litz
Computer Architecture Group
University of Heidelberg
Outline

- Motivation & Goal
- Architecture
- Performance Evaluation
- Conclusion
Motivation

PGAS

- Cache coherent shared memory does not scale
  - Neither Broadcast- nor Directory-based cache protocols
  - See also AMD’s *Probe Filter*

- Partitioned Global Address Space (PGAS)
  - Locally coherent, globally non-coherent
  - Yelick 2006: “*Partitioned Global Address Space (PGAS) languages combine the programming convenience of shared memory with the locality and performance control of message passing.*”

Leverage local coherency advantages,
avoid global coherency disadvantages
Motivation

Goal

- PGAS relies on
  - High bandwidth bulk transfers
  - Fine grain accesses for both communication and synchronization purposes

- Goal
  - Provide best support for fine grain accesses with minimal software overhead
1. Lean **Shared Memory Engine**
   - Address Translation
   - SrcTag Management
   - Stateless on *Origin* side
   - Virtualized

2. Reliable network with in-order delivery
   - HT requests supposed to be answered

3. Leverage HyperTransport’s latency advantage and direct CPU connectivity

4. Minimal protocol conversion
   - CPU → HT → On-chip network → Network
1. LOAD / STORE instruction
2. HT request to SMFU
3. SMFU performs address translation, target node determination
4. Request is send as Extoll network packet to target
5. SMFU performs SrcTag translation
6. HT request to target MC
7. MC handles request
8. HT response to SMFU
9. SMFU re-translates SrcTag
10. HT response encapsulated in Extoll network packet
11. HT response to CPU

Architecture
Working Principle

EXTOLL
Custom FPGA-based high performance interconnection network
Efficient Hardware Support for the Partitioned Global Address Space

Holger Fröning
Computer Architecture Group
Efficient Hardware Support for the Partitioned Global Address Space

Holger Fröning

RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Architecture
Address Translation

\[ gAddr = oAddr - oStartAddr \]

\[ tNodeID = (gAddr \& mask) \gg \text{shift\_count} \]

\[ tAddr = (gAddr \& \sim mask) + tStartAddr \]
**Architecture**

**Matching Store**

- Stores origin information
  - Node ID, HT SrcTag
  - Only used on target side

- Ingress Responder Unit
  - Stores oSrcTag, oNodeID
  - tSrcTag returned

- Egress Responder Unit
  - Uses tSrcTag for lookup
Architecture Framework

- **HT-Core:**
  - Direct CPU connection
  - Fully synchronous
  - Efficient pipelined structure
  - Incoming / Outgoing: 12 / 6 cycles

- **HTAX:**
  - Non-blocking crossbar
  - HT-derived protocol
  - 3(+2) cycles latency
Efficient Hardware Support for the Partitioned Global Address Space

Architecture Framework

- **Network Switch:**
  - In-order delivery of packets
  - Hardware retransmission
  - Virtual Output Queuing
  - Virtual channels
  - Cut-through switching
  - Source-Path routing
  - Credit based flow-control
  - Fault tolerance
  - Remote management access
Performance Evaluation

- Two nodes, each:
  - 4x AMD Opteron 2.2GHz Quad Core
  - 16GB RAM
  - Standard Linux
- Virtex-4 FX100 Prototype
  - 156MHz core clock
  - HT400 interface (1.6GB/s)
  - 6 links, each 6.24 Gbps
  - FPGA: 75% utilization
Performance Evaluation
Limitations & Solutions

- **CPU microarchitecture**
  - Only one outstanding load transaction on MMIO space
  - Max. size of load on uncachable memory 64bit
  - Move to DRAM space
  - Cacheable memory

- **BIOS**
  - MMIO space per PCI
    - B:D:F number max. 256MB
  - Extended MMIO (EMMIO)
  - Move to DRAM space
Performance Evaluation
Remote Loads

- Little’s Law (1961, queuing theory)
  - Number outstanding $N = 1$
  - Response Time $R = 2$ usec (approx.)
  - Throughput $X$

$$X = \frac{N}{R} = \frac{1 \cdot 8B}{2 \mu s} = 4 MB / s$$
Efficient Hardware Support for the Partitioned Global Address Space

Holger Fröning

Performance Evaluation
Remote Stores

- At network peak bandwidth for 64 bytes
- Outstanding transaction rate
Performance Evaluation
Put it into context!

Comparing SMFU vs. EXTOLL (API)
(two nodes, 4x 2.2GHz K10 Opteron)
Conclusion

- Prove of concept of Distributed Shared Memory
- Fine grained remote stores & loads
- Efficient and slim design
- First performance numbers outstanding and encouraging (taken into account the technology differences)

Future

- Atomic operations
- DRAM space (requires coherency)
- Consistency supporting strict and relaxed operations
- Application level evaluation
  - UPC/GASNet
  - Aggregating memory
  - …