MEMSCALE: A New Memory Architecture For Clusters

Presentation at CERCS Systems Seminar at Georgia Tech

Holger Fröning¹, Federico Silla², Hector Montaner², Jose Duato²

¹ Institute of Computer Engineering, University of Heidelberg, Germany
² Parallel Architectures Group. Technical University of Valencia, Spain

April 11th, 2012
Outline

- Motivation

- The MEMSCALE Architecture
  - Evaluation of exclusive use model

- Overcoming Scalability Limitations of Shared Memory
  - Evaluation of shared use model

- Conclusion & Future Work
Motivation – The Data Deluge

- **Memory requirements**
  - Merck Bio Research DB: 1.5TB/quarter
  - Typical Oil Company: 350TB+
  - Facebook: 150TB
  - Twitter: more than 340M tweets per day
  - Particle Physics, Large Hadron Collider: 15PB
  - Human Genomics: 200PB+ (1GB/person, 200% CAGR)

- **Fourth paradigm in science: Data-Intensive Scientific Discovery**
  - After theory, experiment and simulation
Motivation – Technology Constraints

- Performance disparity for different technologies
  - Memory hierarchy only helps if enough locality is present

- Effective memory capacity
  - Memory capacity per core is not keeping the pace of core count increase

![Graph showing access time and capacity per core for different years.](image)

Memory capacity per computing core for 4P servers at highest memory speed.
Motivation – In-Memory Computing

- DRAM for storage ➔ In-Memory Databases
  - Jim Gray: “Memory is the new disk and disk is the new tape”
    - HDD bad for random access, but pretty good for linear access
    - Natural for logging and journaling an in-memory database
  - Google, Yahoo!: entire indices are stored in DRAM
    - Bigtable, Memcached

- Litte or no locality for many new applications
  - “[...] new Web applications such as Facebook appear to have little or no locality, due to complex linkages between data (e.g., friendships in Facebook). As of August 2009 about 25% of all the online data for Facebook is kept in main memory on memcached servers at any given point in time, providing a hit rate of 96.5%”. [Ousterhout2009]
  - Typical computer: 2-8MB cache, 2-8GB DRAM. At most 0.1% of DRAM capacity can be held in caches
Motivation – Scalable Shared Memory Programming

- **Shared memory programming**
  + Pervasive, ease of use
  - No scalability

- **Message passing**
  + Scalable
  - Difficult to program
  - Overhead

- **Overcome message passing’s overhead**
  - Tag matching, progress, data distribution

> Combine advantages of Shared Memory with the **scalability** of message passing

Communication done by FPGAs, configured as message passing engine resp. shared memory engine.
Motivation – Datacenter Utilization

- Many efforts on power usage effectiveness (PUE), but poor utilization
  - Commodity-only approach for datacenters and clusters
  - Resource partitioning

- Overprovisioning of resources to avoid demand paging
  - Underutilization in the average case

Average CPU utilization at Google

Inter-server variation (rendering farm)

[Hennessy2011]

[Lim2010]
Research Questions

- Overcome memory capacity constraints
  - Enabling in-core computing for data-intensive applications

- Re-visiting Shared Memory to overcome MPI limitations
  - Avoiding the overhead associated with MPI

- Benefits of application-specific solutions for database appliances or datacenters
  - Overcoming commodity limitations
The MEMSCALE Architecture
Cluster Computing is a highly viable option, and has proven itself over more than one decade.

Distributed resources, more or less coupled:
- Datacenter
- Standard cluster
- High performance cluster
- Supercomputer

Coupling: local/remote access cost disparities

Appealing because of the strong scaling (and the cost-effectiveness of commodity parts)

Resource partitioning:
- Rules out remote accesses!
- Leads to data partitioning and data movement!
Current architectures

Shared Memory Computers (shared-everything)

- Single address space
  - Up to 64 Cores
  - Up to 512GB RAM
- Shared Memory programming model
- Global coherency among all processors
  - Limited scalability!
- Resource aggregation

Message Passing Systems (shared-nothing)

- Multiple address spaces
  - Nodes are identical – symmetric view
  - Memory over-provisioned
- Message Passing programming model
- No coherency among nodes, only within nodes
  - Unlimited scalability
- Resource partitioning
Our approach: MEMSCALE

- Selective aggregation of resources
  - Shared use of scalable resources (memory)
  - Exclusive use of resources with limited scalability (cores/caches)

- Memory regions can expand to other nodes
  - Overhead of global coherency is avoided

- Spanning up global address spaces
- Decoupled resource aggregation, no resource partitioning
Setting up global address spaces

- Distributed shared memory
- Local address space is split up:
  1. Private partition
  2. Shared partition
  3. Mapping to global address space
- Virtually unlimited
  - Only limited by physical address sizes
  - Currently: $2^{48}$ bytes or 256 TB
Remote memory access

- **Software-transparent access to remote memory**
  - Loads/stores to local mapping of the global address space
  - Parts of the global address will identify target node
  - Forward request over the network
  - Request hits shared local partition on target node
  - If appropriate, send response back

- **Direct, low-latency path to remote memory**
  - Shared Memory Engine (SME)
Remote memory access in detail

Node #0 (Source)
Issuing loads/stores on remote memory

Node #1 (Target)
Serving as memory host

Remote load latency:
1.89\,\text{usec} (R1, Virtex-4)
1.44\,\text{usec} (R2, Virtex-6)

Source-local address
Target node determination
Address calculation

Global address
Loss-less and in-order packet forwarding

Target-local address
Source tag management
Address calculation

Source-local address
Target node determination
Address calculation

Global address
Loss-less and in-order packet forwarding

Target-local address
Source tag management
Address calculation

2012/04/11 - CERCS Systems Seminar
Excursion: EXTOLL

- High performance interconnection network
  - Designed from scratch for HPC demands
  - Optimized for low latency and high message rate
  - Virtex-4 (156MHz, HT400, 6.24Gbps)
- 40% faster for WRF (compared to IBDDR)
Evaluation Methodology

- Prototype Cluster in Valencia
  - 64nodes, 1k K10 cores, 1TB, Virtex-4 FPGA-based network

1. Execution on remote memory only
   - Worst case, applications could also use local memory for locality reasons

2. Compared to execution on local memory
   - Serves as an upper performance bound

3. Compared to execution on remote swap
   - Remote swap as a software-only alternative – similar to ScaleMP
Evaluation – Data intensive calculations

- PARSEC operating on remote memory
  - Canneal has a huge footprint ➞ heavy use of remote swapping
  - Streamcluster too small ➞ no remote swapping, linear access pattern
Evaluation – Data intensive calculations

SPLASH2

(a) Matrix multiplication

(b) FFT

(c) Radix

(d) LU
- Idea is to borrow memory from other nodes
- Sensitivity to background noise?
  - MMULT on client and STREAM on server
- Impact of remote access latency

Each hop adds 600ns
Social network-like DB

- MySQL with modified memory storage engine
- approx. 100GB
- Many fine-grain accesses, almost no locality
- Currently: read-only
Evaluation – In-Memory-Database (I)

- **Comparing**
  - Local memory, **30x faster**
  - Remote memory (MEMSCALE), **reference**
  - SSD (SATA), **25x slower**

**Graph:**
- **Local Memory:** Upper performance bound, but insufficient capacity. Included for reference.
- **MEMSCALE:** unlimited capacity, closing the performance gap (2500 Q/s) to local memory.
- **SSD:** insufficient performance (90 Q/s), not scaling with the number of available cores per node.
- **HDD:** neither scalable nor providing performance

**Executed on single node**
Evaluation – Limitations

- Concurrent load requests to remote memory
  - Performance saturates for 4-5 flows
  - Limited number of outstanding requests for MMIO space
- Solution: coherent integration using cHT
  - Viewed as a memory controller from the OS
  - (Cache coherency)
Overcoming Scalability Limitations of Shared Memory
Collaborative Use of Memory Regions

Memory Region 1
Memory Region 2
Memory Region 3
Memory Region 4
Memory Region 5

Main Memory
Processors/Caches

Main Memory
Processors/Caches

Main Memory
Processors/Caches

Main Memory
Processors/Caches

Main Memory
Processors/Caches
On-Demand Consistency

- BSP-like execution model
- Consistency only guaranteed at synchronization points
  - Enriched Barriers/Locks
  - Cache invalidations (WBINVD, CLFLUSH, ...)
- Reverting from continuous consistency (pessimistic model) to partial consistency (optimistic model)
- No longer necessary to broadcast probes on every shared memory access
Evaluation: Barrier Performance

- Example synchronization primitive
  - Only using remote stores
  - Push-style communication
  - Local coherent spinning on updates
  - Tree-like structure

- “Fastest barrier in the west”
  - 256 threads in 9.3μsec
  - 1k threads in 18.0μsec

IB: 20-30μs for 64P [Graham2010]
Evaluation – On-Demand Consistency

- FFT
  - Focus on scalability
  - Matched problem size
  - 5 barriers per iteration

- Result
  - Scalability matches MPI execution
  - Unmodified program, only changes within library calls (barrier)
Cluster-level execution
- MEMSCALE vs. MySQL cluster

MySQL cluster
- 16 nodes, Gigabit Ethernet
- 450 queries per second
- Saturation starts at 20-30 flows

MEMSCALE
- 16 nodes, EXTOLL R1 w/t SME
- 128GB memory pool
- 35k queries per second, 77x
- Linear scalability up to 4/5 flows per node (64-80 total), then saturation

Limited by
- Number of outstanding loads
- Access latency
Conclusion and Future Work
Conclusion

- Sole use of commodity parts is hitting a wall
- MEMSCALE: A new memory architecture for clusters
  - Overcoming resources partitioning
  - Scalable memory aggregation
  - Enabling in-memory computing
- 500x gap reduced to 10-36x
Future Work

- Eventually consistent model for MEMSCALE-DB

- Use of global address spaces to improve utilization in datacenters – „Virtual DIMMs“
  - Reduce memory overprovisioning

- Support for „heavy queries“ (TPC-H like) - „Red Fox“
  - MEMSCALE is currently not very suitable for stream-like accesses

- Collaborations with Sudhakar Yalamanchili, Jeff Young et al. from GT
Future Work - MEERkAT

- **MEERkAT** – Improving datacenter utilization and energy efficiency
  
  - System-level virtualization & dynamic resource aggregation of heterogeneous resources, various migration levels

- **Analogy**
  
  - A meerkat is a small, highly socialized animal, living in large groups in big underground networks with multiple entrances. When outside, meerkats are mostly known for their altruistic behavior, one or more standing sentry while others are busy with foraging or playing.
  
  - The MEERkAT approach gears to mimic this behavior, relying on a tightly-coupled network with resources acting very socially, being altruistic when idle and thus aspiring to optimize the overall situation.
Let’s check the „compatibility“ of meerkats with red foxes, ocelots or oncillas ;)
Unconventional Cluster Architectures and Applications ([http://www.gap.upv.es/ucaa](http://www.gap.upv.es/ucaa))

- Platform for innovative ideas that might turn out to be disruptive in the future
- Co-located with ICPP2012
- Topics of interest include
  - High-performance, data-intensive, and power-aware computing
  - Application-specific clusters, datacenters, and cloud architectures
  - New industry and technology trends
  - Software cluster-level virtualization for consolidation purposes
  - New uses of GPUs, FPGAs, and other specialized hardware
  - ...

Submission deadline: **May 1st, 2012**
Thanks for your attention!

Holger Fröning
froening@uni-hd.de


Backup Slides
Barrier Performance

![Graph showing barrier performance vs network size. The graph illustrates the relationship between the barrier execution time and the network size (number of nodes)](image-url)
Most experiments made with outdated Virtex4 technology

Ventoux (HT600, 200MHz)
  • 1.45 predicted, 1.44 measured
Name of my friends
SELECT SQL_NO_CACHE u_name FROM users, friendships WHERE f_u1 = ? AND f_u2 = u_id

Number of new messages directly sent to me
SELECT SQL_NO_CACHE m_text FROM messages WHERE m_to_id = ? AND m_to_t = "user" AND m_new = TRUE

Number of new messages sent to any of my groups
SELECT SQL_NO_CACHE m_text FROM messages, affiliations WHERE a_user = ? AND m_to_id = a_group AND m_to_t = "group" AND m_new = TRUE LIMIT 50

100 last messages sent by me
SELECT SQL_NO_CACHE m_text FROM messages WHERE m_from = ? LIMIT 100

Name of my groups
SELECT SQL_NO_CACHE g_name FROM groups, affiliations WHERE a_user = ? AND a_group = g_id LIMIT 50

Text of 3 given messages
SELECT SQL_NO_CACHE m_text FROM messages WHERE m_id = ? OR m_id = ? OR m_id = ? OR m_id = ? OR m_id = ?