First Experiences with the SCC and a Comparison with Established Architectures

Stefan Lankes, Carsten Clauss
Chair for Operating Systems
RWTH Aachen University
Agenda

- First RCCE and MPI benchmark results
- Cache behavior of the P54C Architecture
- Optimization of `RCCE_put` and `RCCE_get`
  - Learning from the past
- Potential of MP-MPICH
  - e.g. clustering of SCC systems
- Capability of SVM systems
  - Future project aims
- Conclusions and Outlook
• Research topics
  – Operating Systems (of course)
  – Parallelization strategies
    ▪ Shared Memory
    ▪ Message Passing
  – Distributed Systems
  – Embedded and Real-Time Systems
  – …

➢ The Chair for Operating systems has developed an own MPI distribution
  – Based on MPICH
  – Support of different high performance interconnects (e.g. SCI)

➢ The “ultimate” MPI benchmark: Ping Pong
  - It is obvious to use RCCE with Message Passing Buffers
  - Enlarge RCCE example “PingPong” to send messages with variable size
Bottleneck: Bad Cache Behaviour

No cache line fill by a write miss!
Approach: Data Prefetching

Data Prefetching
No SSE → Prefetching by hand
Ping Pong

RCCE

RCCE + prefetching

Bytes

MB/sec
Metacomputing / Grid-enabled MPI

- Layered Design of MPICH:

  MPI Applikation

  MPI Interface (API)

  Profiling Interface (PMPI)

  MPIR Layer (platform independent)

  ADI2 Interface

  MPID Layer (platform dependent)

  OS / HW Interface

  Operating System Hardware

  Generic Implementation of the ADI Device

  Channel Interface

  Channel Device Implementation (e.g. ch_smi)
Metacomputing / Grid-enabled MPI

- Multi-Device Support \( \rightarrow \) **MetaMPICH**

![Diagram](image_url)

- MPI Applikation
- MPI Interface (API)
- Profiling Interface (PMPI)
- MPIR Layer (platform independent)
- ADI2 Interface
- MPID Layer (platform dependent)
- OS / HW Interface
- Operating System Hardware

**Generic Implementation of the ADI Device**

- Channel Interface
  - SMI Device ch_smi
  - RCCE Device ch_scc
  - P4 Device ch_p4

**Channel Interface**

- SMI
- iRCCE
- P4

...
• The Secondary Device → ch_usock
Ping Pong

- RCCE
- RCCE + prefetching
- MPI (eager)
- MPI (shmeager)

MB/sec vs Bytes

Bytes: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536
Pipelining on Shared Memory Architectures

• Most SCI adapters used PCI (express) as I/O bus
  ➢ Cache coherence not supported
  ➢ Only local segments are cache able
  ➢ On SCC, the cache is on all shared regions disabled

• Classic optimization technique: Pipelining
Parallelization Strategies

- Two basic strategies
  - Communication via Message Passing
    - Code restructuring
    - It’s difficult to use, because we learnt sequential programming (C, C++, Java)
    - Scales very well (→ MPI, URPC, Barrelish)
  - Communication via Shared Memory
    - The first contact is easier. Feels like sequential programming.
    - However, it is much more complex (False Sharing, Races, Deadlocks, NUMA).
    - Incremental parallelization
    - Scales mostly good…
**Benefits of Shared Memory Parallelization**

- Algorithms with a dynamic data structure and access pattern are easier to parallelize.
  - Adaptive PDE solvers
    - e.g. Structured Adaptive Mesh Refinements PDE solver
    - Not ideal for NUMA architectures
    - Using of Affinity-On-Next-Touch to redistribute pages
  - Airline flight scheduling module
    - Part of Lufthansa Systems’ decision support system
    - Searching for a flight between A and B with N connections
    - Consideration of departure- and arrival-time, capacity, costs,…
    - More complex as the *Shortest Path Problem*
    - Using of double-linked lists

- **Aim:** A scalable Shared Virtual Memory (SVM) system on the top of SCC
MetalSVM: A Virtual NUMA Architecture on SCC

- MetalSVM will be a small hypervisor, which uses paravirtualization techniques to run Linux on SCC.
- The integrated SVM system gives the Linux kernel a transparent (and cacheable) view of the memory.

- Linux defines a clear interface between the paravirtualized kernel and its hypervisor.
- MetalSVM uses this interface to paravirtualize Linux.
- For instance, the spinlock interface could be used to synchronize Linux threads over the MPB.
Device Emulation

- Linux provides already an I/O virtualization framework called Virtio.
- Virtio provides a common front end for e.g network and block devices.

- This increases the reusability of code across different hypervisors (Xen, KVM, lguest).
- MetalSVM will support this framework to minimize the changes to Linux kernel.
- The smooth integration of a new device into Linux could be realized by developing a specific device emulation layer for MetalSVM.
Conclusions and Outlook

• Established techniques could be used to increase the performance
• The cache behavior of P54C could be “nicer”.
  – More influences will be preferable
• Clustering of SCC via MP-MPICH already possible
• The SCC is an ideal architecture to build a scalable SVM systems
  – Fast collective operations
Ping Pong

Latency [usec] vs. Bytes

- RCCE
- RCCE + prefetching
- MPI (short, 512)
- MPI (eager)
- MPI (rndv)
- MPI (shmeager)
- RCCE (pipelined)