Linux on NUMA architectures

4th Annual Workshop on Linux Clusters for Super Computing

Linköping, October 22-24, 2003

Jes Sorensen
Wild Open Source
jes@wildopensource.com
http://www.wildopensource.com/

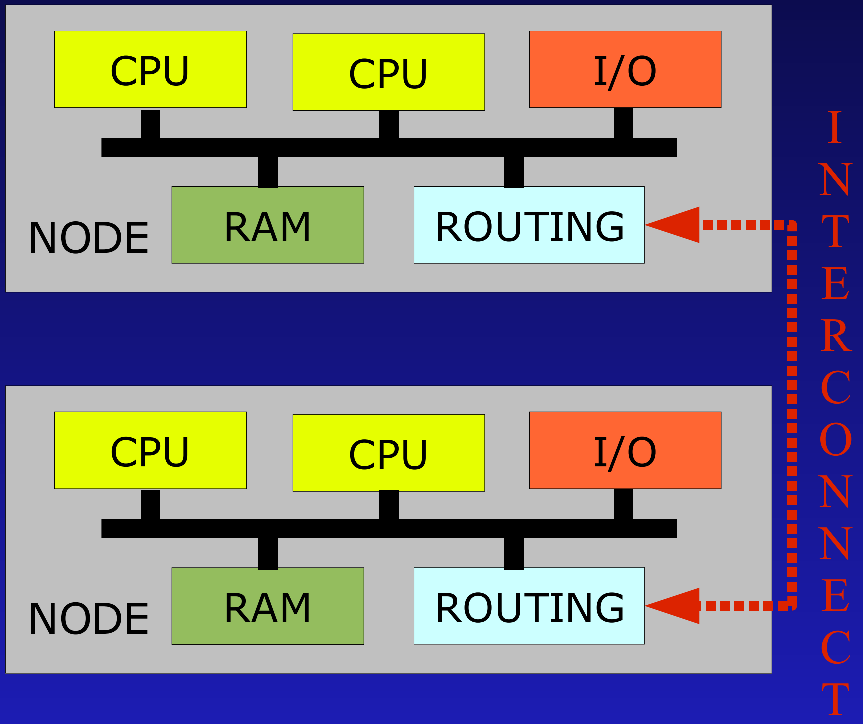
Agenda

- Why should we care about NUMA?
- NUMA architecture background
- Common NUMA problems and how they compare to SMP
- NUMA alternatives
- Q&A

Why this NUMA thing?

- Why care about NUMA? aren't clusters much better for the job?
  - One big NUMA box allows RAM sharing, major benefit with large datasets
  - AMD64/Opteron systems are NUMA even for small SMP systems .... HyperThreading
- NUMA optimizations ruins performance for UP/SMP systems
  - A few may be radical, however most cases are actually a benefit for all 2+ CPU systems

NUMA 101



- Local to each node:
  - CPU(s)
  - RAM
  - I/O (PCI)
  - Interconnect
- Between nodes:
  - Hypertransport, NUMAFlex, etc.
  - Node-to-node connections / routed

More NUMA

- Advanced NUMA configurations use routers to limit distance between nodes
  - Access to memory can be multiple hops away, increasing latency!
- Some systems have dedicated I/O and/or memory nodes
- Depending on platform, CPU nodes can contain one or more CPUs

Memory & Allocation

- To avoid unncessary remote memory access, kernel memory allocator must allocate memory on task's node (unless requested otherwise)
- Scheduler should prioritize CPUs on same node as task was previously run on
- Memory allocation and schduling on non-NUMA systems can be treated as 1-node systems (allows for simplifications at compile time)

Kernel Internal Issues

- All data structures frequently used for write must be cache line aligned
  - Cost of cache-line ping-pongs between nodes often 2-3+ times higher than on regular SMP
- Atomic operations cause atomic bus operations across interconnect:
  - gettimeofday() performance > 2x by eliminating atomic operation in gettimeoffset()
- Replication of kernel text segment in RAM on each node

Scheduling

- Node aware scheduling
- Node-affinity rather than CPU-affinity API required for userland (tasks/threads sharing datasets)

Spin Locks and other goodies

- Linux spin locks very simple and fast, no exponential backoff
- CPUs on remote nodes have higher latency to reach lock memory
- read/write locks suffer from exponential fairness problem when number of CPUs grow (alternative: read-copy-update – RCU)

PCI I/O

- DMA cache coherent as per spec!
- MMIO read/write operations stalls during DMA transactions
    - DMA transactions between nodes have high latency
    - NUMA hardware vendors often cheat and violate spec by allowing read/write to by-pass in-flight DMA
    - Compensate at device driver level (must be transparant to avoid custom device drivers for NUMA)

libnuma

- Export topology information to applications
- Node-affinity (CPU memsets)
- Node-aware memory allocation

Questions?

- Linux and NUMA?
- Linux kernel issues?
- Linux ia64?
- Anything!