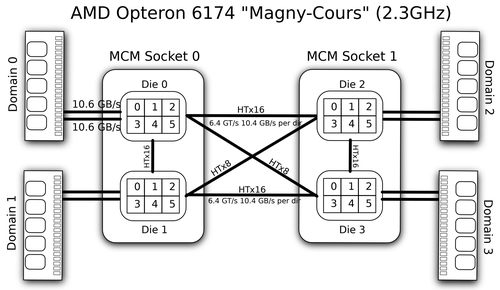
Techniques for Managing Data Distribution in NUMA Systems

Alexander M. Merritt
merritt.alex@gatech.edu
Center for Experimental Research in Computer Systems
College of Computing, Georgia Institute of Technology
Atlanta, GA

Kevin T. Pedretti
ktpedre@sandia.gov
Sandia National Laboratories
Albuquerque, NM

Karsten Schwan
schwan@cc.gatech.edu
Center for Experimental Research in Computer Systems
College of Computing, Georgia Institute of Technology
Atlanta, GA

Introduction



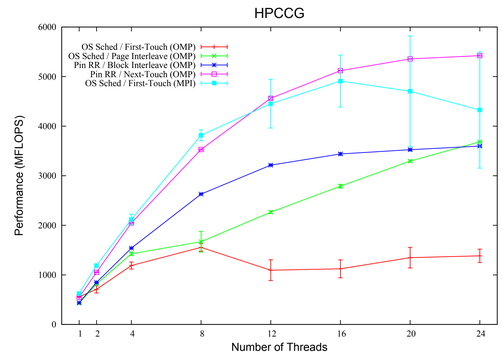
Background

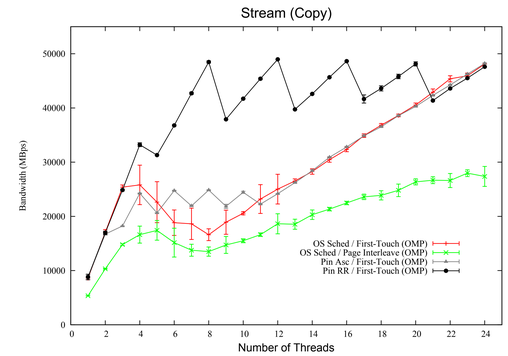
* Application developers increasingly use hybrid programming models: threading within a node, MPI between nodes.
* Threading models assume shared UMA and avoid intra-address space data distribution.
* Supercomputing hardware becoming less uniform: more cores and deeper NUMA latencies (diagram above): *latencies limit scalability*

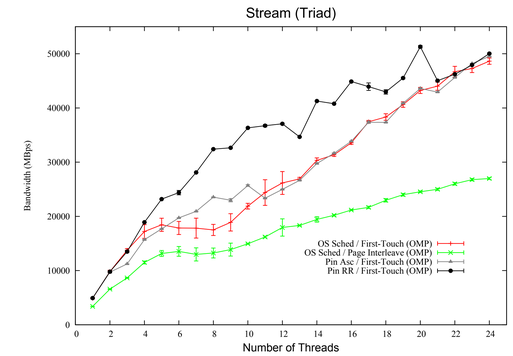
Motivation

* Magny-Cours the basis for Sandia/LANL "Cielo" supercomputer
* Operating system process scheduler unaware of affinity between thread and data.
* Programmer required to explicitly manage memory distribution within address space. Methods for doing so are primitive and intrusive.
* Application runtime phases influence memory access behavior. Static intra-address space memory distribution policies not adaptive.

Observations







Methods

First-Touch
Default allocation strategy in Linux kernel. Pages allocated in NUMA domain where first access is observed. No migration is performed.

Interleaving
Kernel-supported page interleaving among domains (numactl)
BIOS-supported cacheline interleaving among domains (MMU)

Next-Touch
Kernel patch [Goglin 2009]. madvise() system call enhanced to trap pages on access using page table protection bits.

Discussion

* Current methods are static and only improve performance for specific phase characteristics:
  - first-touch with thread pinning is most beneficial for STREAM, but is a scalability limiter for HPCCG
  - next-touch with thread pinning is most beneficial for HPCCG
* The Linux process scheduler is unaware of thread/data affinities. This is evident with HPCCG: even with MPI (turquoise) where state is replicated and allocated locally, the scheduler will still shuffle tasks among cores (wide error bars).
* "Pin Asc" indicates threads were pinned to cores using Linux-logical IDs (gray). Their mapping may change, giving different performance results. In our case, core IDs 1-12 correlated to NUMA domains 0-1, saturating memory bandwidth using only 4 threads.

Dynamic Runtime Migration

Proposal

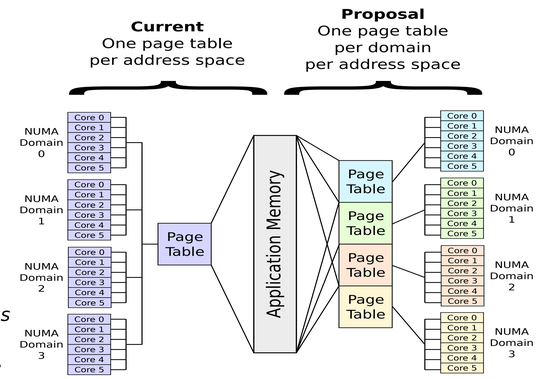
* Automate memory distribution to account for application phases
  - Periodically profile application at runtime to observe access patterns
* Provide transparency to application; no recompiling

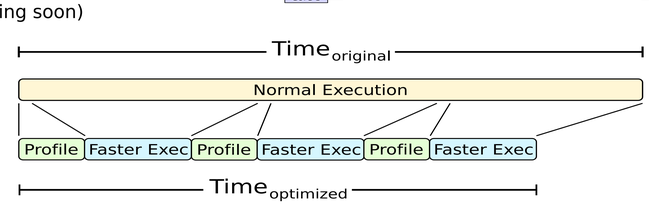
Technique

* Use the page table!
* Need source and destination domain, frequency of memory accesses
* Each memory page represented by page table entry
  - Processor updates access bit. *But only one bit per page across domains*
* Solution: duplicate page tables to increase access bits, and install appropriate page table in CPU core's cr3 register when thread scheduled.
* Implementation in Kitten Lightweight kernel (coming soon)

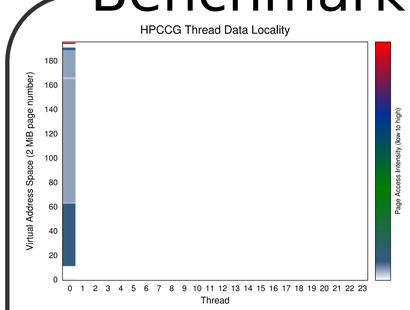
Optimizations

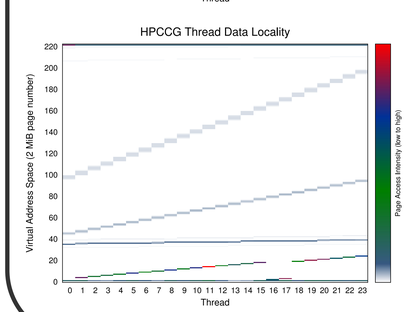
* Use of large pages
* Linear allocation of page table entries
* Widen access bit to a saturating counter





Benchmark Phases





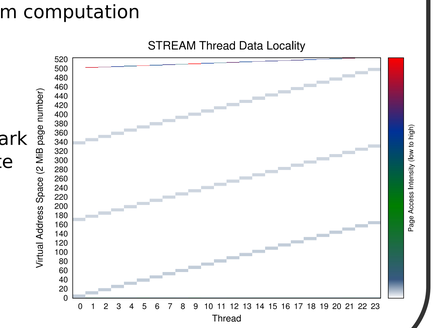
* Pintool, a dynamic binary instrumentation tool, allows us to capture each memory access: address, access type, etc.
* For our purposes, a phase relates only to memory access patterns.
* Visualizing application phases enables developers to identify behavioral patterns.

HPCCG (left two)

* Linear system solver using the conjugate gradient method. Important representative compute kernel for HPC codes at Sandia
* Phase 1: main thread allocates and initializes all data
* Phase 2: fork threads, perform computation

STREAM (right)

* Synthetic parallel (OpenMP) memory bandwidth benchmark
* One phase: spawn & compute



Future

* Examine additional codes
* Automate application phase visualization and detection during instrumentation

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.