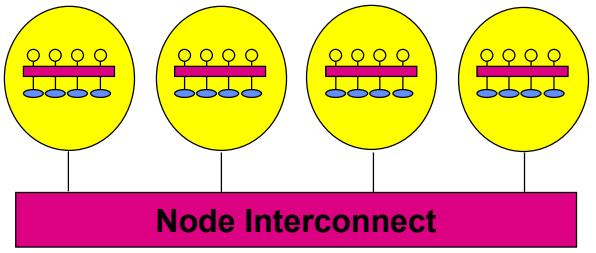
OpenMP - Cluster Extensions

Thomas Bönisch, Matthias Müller
{boenisch,mueller}@hlrs.de

SMP - Cluster (Hybrid System)

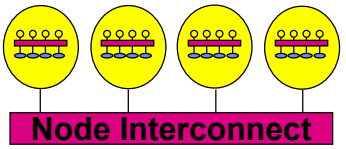
- Most modern high-performance computing (HPC) systems are clusters of SMP nodes



- DMP (distributed memory parallelization) on the node interconnect
- SMP (symmetric multi-processing) inside of each node

SMP - Clusters Current Solutions for Programming

- MPI based:
  - the MPP model
    - **massively parallel processing**
    - **each CPU = one MPI process**
  - MPI + OpenMP
    - **each SMP node = one MPI process**
    - **MPI communication on the node interconnect**
    - **OpenMP inside of each SMP node**
    - **DMP with MPI & SMP with OpenMP**
  - MPI + automatic parallelization

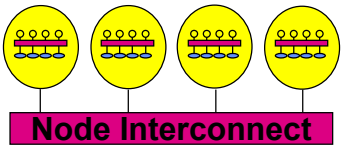


MPI + OpenMP on SMP-Clusters

- Advantages
  - Could be effective utilizing heavyweight communications between nodes and lightweight threads within a node.
  - Less communication packets and larger communication packets than pure MPI on SMP Clusters.
  - Straight forward exploitation of multiple levels of parallelism.
- Disadvantages
  - Very difficult to start with OpenMP & modify for MPI (non-incremental)
  - Very difficult to program, debug, modify and maintain
  - Generally, cannot do MPI calls from within parallel regions
  - You need experience in MPI and OpenMP
- Could provide highest performance (at a cost!)

SMP - Clusters Current Solutions for Programming

- MPI based:
  - the MPP model
    - **massively parallel processing**
    - **each CPU = one MPI process**
  - MPI + OpenMP
    - **each SMP node = one MPI process**
    - **MPI communication on the node interconnect**
    - **OpenMP inside of each SMP node**
    - **DMP with MPI & SMP with OpenMP**
  - MPI + automatic parallelization



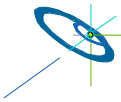
- Other models: HPF, MLP, ...
- **What about an OpenMP-like approach ?**

"Distributed OpenMP" Current Approaches

- New directives for distributing data
  - Portland Group (following slides)
  - Compaq
- No extensions to OpenMP itself, but an new more intelligent run time environment
  - Real World Computing Partnership (RWCP) Japan
- Intel/KSL announced KAP/Pro Toolset Network Edition
  - DVSM approach based on TreadMarks
  - This is still an internal research project
- Other groups are also working on similar stuff

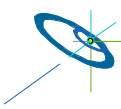
A New Parallel Programming Paradigm for SMP Clusters ?

- Ideally, a mixed parallel programming paradigm would exist that
  - enable parallelism using low level, efficient one-way communication between PEs
  - using an efficient multi-threaded mechanism between CPU within a PE
- Such a mixed parallel programming paradigm should
  - allow for full efficiency within an SMP System while also
  - retaining efficiency for distributed memory clusters
- Other desired features:
  - incremental parallelism
  - ease-of-programming
  - maintainability and debugability should be maintained

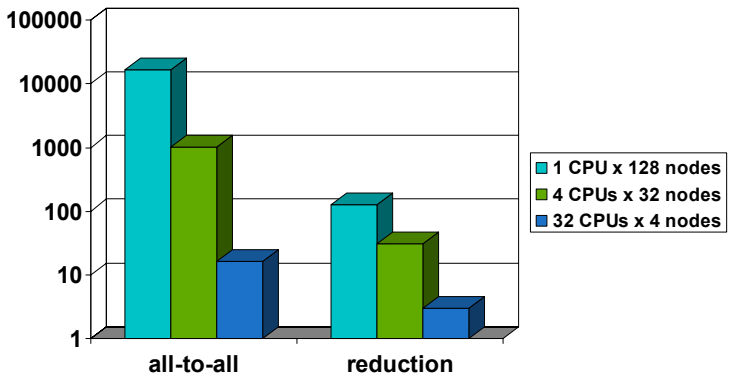


Distributed OMP - Characteristics

- Each node is treated as a single process with one-way communication between the nodes
- lightweight threads within a node
- Each node's program would utilize multiple threads according to OpenMP directives and OpenMP library calls
- Directives for data distribution to the nodes
  no explicit communication
- Less data movements between nodes plus larger packets of data.



Number of Communications



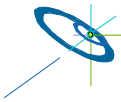
But the message size is increasing in the first case

Distributed OpenMP - Advantages

- One can have a single portable code that can be executed on SMP systems and distributed clusters and hybrid combinations of SMP clusters
- Single SMP node performance will not degrade
- Incremental parallelization capability
- Managers don't have to risk "betting" on SMP systems being the dominant system

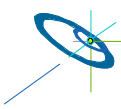
Distributed OpenMP - Disadvantages

- This is still a research project
- You need a fast network (expensive)
- User still must give some thought about data partitioning/distribution



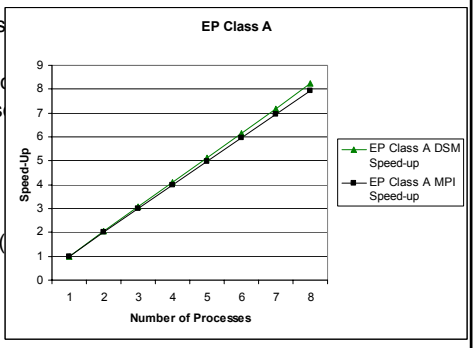
Case Studies

- NAS Parallel Benchmarks EP, FT, and CG:
  - Message passing and sequential version
- Automatically generate OpenMP directives for sequential code using CAPO (www.nas.nasa.gov/Groups/Tools/CAPO)
- Omni Compiler
- Compare speedup of:
  - Message passing vs. OpenMP/DSM
  - OpenMP/DSM vs. OpenMP/SMP
- Hardware platforms:
  - DSM Test Environment
    - **Use only one CPU per node**
  - SMP 16-way NEC AzusA
- Case study was conducted with Gabrielle Jost from NASA/Ames and Matthias Hess, Matthias Mueller from HLRS



The EP Benchmark

- Embarrassing Parallel:
  - Generation of random numbers
  - Loop iterations parallel
  - Global sum reduction at the end
- Automatic Parallelization without user interaction
- MPI implementation:
  - Global sum built via MPI_ALLREDUCE
  - Low communication overhead (< 1%)
- OpenMP/DSM:
  - OMP PARALLEL
  - OMP DO REDUCTION



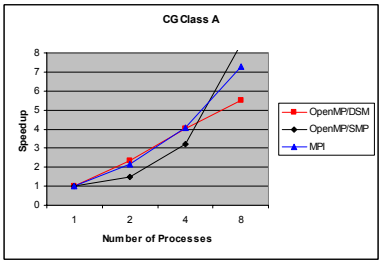
Linear speedup for MPI and OpenMP/DSM. No surprises.

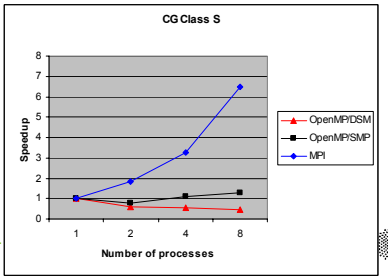
The CG Benchmark

- Conjugate gradient method to solve an eigenvalue problem
- Stresses irregular data access
- Major loops:
  - Sparse Matrix-Vector-Multiply
  - Dot-Product
  - AXPY Operations
- Same major loops in MPI and OpenMP implementation
- Automatic parallelization without user interaction

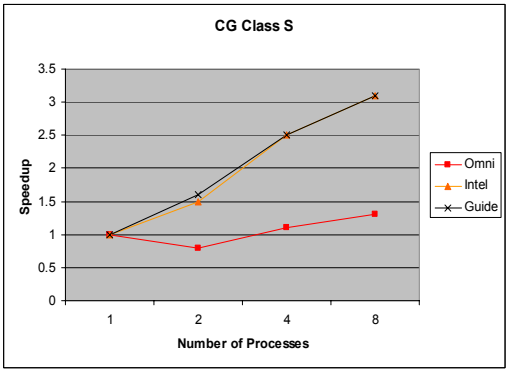
CG Benchmark Results (1)

- Class A:
  - Problem size: na=14000, nz=11
  - OpenMP/DSM efficiency about 75% of that of MPI
- Class S:
  - Problem size: na=1400, nz=7
  - MPI about 20% communication.
  - No speedup for OpenMP/DSM due to:
    - **Large Communication to Computation Ratio**
    - **Inefficiencies in the Omni Compiler**





CG Benchmark Results (2)

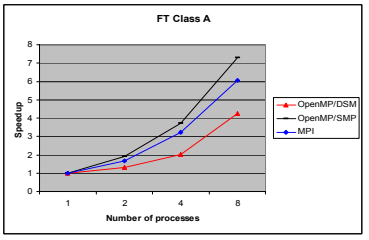


The FT Benchmark

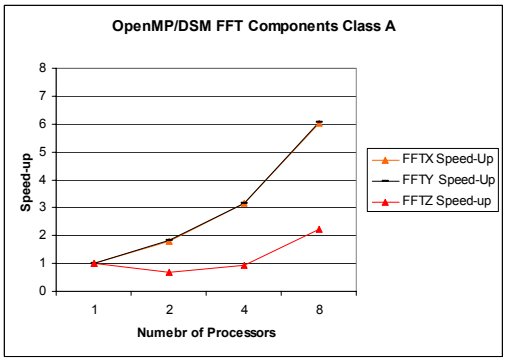
- Kernel of spectral method based on 3D Fast Fourier Transform (FFT)
- 3D FFT achieved by a 1D FFT in x, y, and z direction
- OpenMP parallelization required some user interaction
  - Privatization of certain arrays via the CAPO user interface.

FT Benchmark Results (1)

- MPI Parallelization:
  - Transpose of data for FFT in z-dimension
  - 15% in communication
- OpenMP Parallelization:
  - OMP DO PARALLEL
  - Order of loops changes for z-dimension
- OpenMP/DSM efficiency about 70% of MPI
  - Extra communication introduced by DSM system (false page sharing?)
  - Remote data access required for FFT in z-dimension



FT Benchmark Results (2)



Conclusions:

- Rapid development of parallel code running across a cluster of PCs was possible
- OpenMP/DSM delivered acceptable speedup if the communication/computation ratio is not too high:
  - OpenMP/DSM showed between 70% and 100% efficiency compared to MPI for benchmarks of Class A
- Problems encountered:
  - High memory requirements for management of virtual shared memory (> 2GB)
  - Potential scalability problems
- Need for profiling tools