EWOMP03 OMPlab: Experiment with Cluster-enabled OpenMP: OpenMP for Software DSM system on PC Cluster

Mitsuhisa Sato,
Yoshinori Ojima, Taisuke Boku

(University of Tsukuba)
Omni OpenMP Compiler Project

• A Project Started at RWCP
  – (Real World Computing Partnership, 1996-2001)
  – Still continued by government research fund at U. of Tsukuba and some universities in Japan.

• Research Objectives
  – Portable implementation of OpenMP for SMPs
    – Sun Solaris (solaris threads, POSIX thread) for Sparc
    – Linux (POSIX thread) for x86 and alpha
    – SGI IRIX (sproc and POSIX thread) for O2K
    – IBM AIX, HP-UX, …
  – Design and implementation of Cluster-enabled OpenMP for PC/WS/SMP clusters
    • Support seamless programming from SMPs to clusters
    • Using page-based Software Distributed Shared Memory System
      – SCASH implementation
    • Compiler-directed SDSM approach (very preliminary…)
      – Instrument shared memory R/W by the compiler to insert memory consistency check code.
What’s Cluster-enabled OpenMP?

• Parallel programming for a cluster of PC, WS, SMP
  – Distributed memory systems
  – Use a message passing library (MPI, PVM)
  – High performance, but difficult and cumbersome

• OpenMP
  – proposed for shared memory multiprocessors
  – Easy-to-use parallel programming model

• Cluster-enabled OpenMP
  – Use OpenMP for programming on a cluster (distributed memory system)
  – Our approach: Use a software distributed shared memory system SCASH as an underlying runtime system on a cluster
Software Distributed Shared Memory System
SCASH in SCore

• A page-based software distributed shared memory system
  – The consistency of shared memory is maintained on a per-page basis.
  – Use kernel memory protection mechanism to detect R/W.
• Use PM communication layer for Myrinet, ethernet.
  – Low-latency and high bandwidth
  – One-side communication
  – Running on RWC SCore Cluster
• Two page consistency protocols
  – invalidate (default) and update
  – The home node of a page keeps the latest data of the page.
• Release Consistency (RC) memory model with the multiple writer protocol.
  – At a synchronization point (barrier and lock), only modified part (DIFF) is transferred.
• SCASH primitives are provided as a user-level library.
  – shared memory space allocation, home assignment, barrier, lock, ...
Cluster-enabled OpenMP

• **OpenMP** for a cluster (distributed memory system)
  – Use a software distributed shared memory system **SCASH** as an underlying runtime system on a cluster
  – compiler translates OpenMP into “shmем” memory Model

  • **OpenMP**
    – All variables are shared as defaults.
    – No explicit shared memory allocation

  **Omni OpenMP Compiler**

  • **“shmем” memory model**
    – All variables declared statically in global scope are private.
    – The shared address space must be allocated by a library function at runtime.
    – Example: **SCASH**, Unix “shmем” system call
Platform at Our University

- PC cluster “COSMO”

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>PentiumII Xeon 450MHz (4-way SMP)</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>1MB</td>
</tr>
<tr>
<td>Memory</td>
<td>2GB</td>
</tr>
<tr>
<td>Nodes</td>
<td>8</td>
</tr>
<tr>
<td>Network</td>
<td>800Mbps Myrinet, 100base-TX Ethernet</td>
</tr>
<tr>
<td>OS</td>
<td>Linux Kernel 2.4.18</td>
</tr>
<tr>
<td>SCore</td>
<td>version 5.0.1</td>
</tr>
<tr>
<td>Compiler</td>
<td>gcc 2.96 (Optimize option –O4)</td>
</tr>
</tbody>
</table>
How to use Omni/SCASH

- On Cosmo cluster in Japan
  - SCore running on Cosmo
  - via internet access to our university

- Compile
  - Just compile `omniconfig=scash options`
    - `omcc -omniconfig=scash foo.c`

- Run
  - Use `scrun` command to run your program on SCore.
    - `scrun -nodes=4x1 a.out`

- Compatibility to SMP
  - The variables used in external libraries (stderr, errno) must be thread private!
  - I/O(all private to nodes)
  - system calls (raw level read/write)
OpenMP extension for SCASH

- The home node allocation affects the performance on SCASH
  - needs to control the locality
- Data mapping directives

```c

dimension a(100,200,100)
$omn mapping(a(*,*),block)
int a[100][200][300];
#pragma omni mapping(a[block][*][*])

$OMP do schedule(affinity,a(*,*),i))
do I = 1,100
    a(....,...,i) = ...
#pragma omp for schedule(affinity,a[i+1][*][*])
for(i=1; i < 100; i++)
    a[i+1][....][....] = ...
```
Omni/SCASH on MPI/TCP/IP

• A new version (still in beta state)
  – use MPI and TCP/IP as communication layer of SCASH
    • TCP/IP for remote memory access
  – running on MPI environment
  – portable!?
  – no support for SMP node. one process runs one processor.

• compile & link & run
  – compile -omniconfig=scash options
    • omcc -c -omniconfig=scash foo.c
  – Link with MPI lib and SCASH lib
    • mpicc -o foo foo.o libscash.a
  – run
    • mpirun -np 4 foo

• We will finish the installation until the end of today (hopefully!?)
Hands on

• You can test Omni/SCASH, using SCore PC-cluster (COSMO) in Japan via Internet access
• Demo and test of “Omni/SCASH on MPI/TCP/IP” on PC-Cluster here
  – Please wait the installation

• Demo on iP/OMP
  – See our presentation done on the first day of OMPtalk.