Performance Models for Cluster-Enabled OpenMP Implementations

Jie Cai, Alistair Rendell, Peter Strazdins, H’sien Jin Wong

5th August, 2008
Presenting at 13th ACSAC
Outline

- Introduction
  - OpenMP
  - Cluster-enabled OpenMP implementations
- Performance models
  - Critical path model
  - Aggregate model
- Experimental results
- Conclusion
What is OpenMP?

- Standard multi-processing programming model for shared memory architecture
  - consists of a set of compiler directives, library routines, and environment variables
  - is supported by most C/Fortran compilers, e.g. icc/ifort, gcc/g77..
  - fork-join parallelism approach
Implementing Cluster OpenMP

- A software system to extend OpenMP program on cluster.
- Typically based on software Distributed Shared Memory (sDSM) systems, a virtual shared memory environment.
Memory is managed in fixed sized block, called pages. Memory pages are protected by using “mprotect”, and pages are kept consistent via detecting and servicing different types of page-faults. “Invalid”, “Read-Valid”, and “Write-Valid”.

Accessing to “Invalid” pages will require modified page been fetched, and writing to “Read-Valid” pages will require twin copy of the page to be made for later diffs calculation. This is known as memory consistency work. The major overhead of sDSM systems is the memory consistency overhead.
Page Transitions in CLOMP

- Released in 2006 with Intel C/Fortran Compiler.
- Derived from TreadMarks, which uses the Lazy Release Consistency model.
- **Write fault**: all local operations.
- **Fetch fault**: involves network communications.

Write notices from other process (barrier, lock, flush..)
Invalid
Write-Valid
FETCH faults (first read)
WRITE faults (first write)
Read-Valid
Write notices
Changes consumed by other process
(Initial state)
Memory consistency work is the major overhead of cluster-enabled OpenMP systems, which sometimes dominates the overall performance. Page fault detection and servicing are the major activities for the memory consistency work.

- parallel execution time = (sequential execution time / nprocs) + overhead

Therefore, the numbers and costs of different types of page faults when running an OpenMP application could be utilized to rationalize its performance.

- Critical path model – finding the number of page faults along the critical path of each parallel regions and combine them to estimate the overhead
- Aggregate model – uses aggregated page fault numbers with a serialization fraction to estimate the overhead
Critical Path SDP Model (1)

- \(T^c\): computing
- \(T^i\): idling/waiting
- \(T^o\): overhead (page faults)

The diagram illustrates the critical path model with segments labeled as follows:

- \(P_{\text{start}}\): implicit barrier, wake up slave threads
- \(R^p_1\), \(R^p_2\), \(R^s_1\): various segments
- \(P_{\text{end}}\): endpoints

Timed Section:

- Green segments: \(T^c\)
- Red segments: \(T^o\)
- Dashed segments: \(T^i\)
Critical Path SDP Model (2)

Timed Section

Critical Path

P_start (implicit barrier, wake up slave threads)

$P_0$ $P_1$ $P_2$ $P_3$

$T^c$: computing
$T^i$: idling/waiting
$T^o$: overhead (page faults)

$R^p_1$ $R^s_1$ $R^p_2$
Critical Path SDP Model (3)

- Find the critical path which contains longest computation and overhead for each parallel region.
  - Assumes the page faults happening on different threads within a same parallel region is overlappable.
  - Assumes that the costs for servicing page faults are constant on a given hardware platform.
  - Assumes that the OpenMP application is load-balanced, therefore computation time is same for each thread.
  - Overhead from the number of page faults will be the determinant of the critical path for the parallel region.

- Assumes the sequential region contained in the timed section of an OpenMP application is negligible.

- The combined critical paths for each parallel region could be used to estimate the overall performance.
Critical Path SDP Model (4)

- Critical path SDP model:

\[ T_{est}^{crit} = (T(1)/p) + \sum_{R_1^p}^{R_n^p} \frac{P_p}{P_0} \text{Max}(N_W C_W + N_F C_F) \]

- \( T(1) \) stands for the total elapsed time for timed section.
- \( N_w, N_f \) stand for number of write and fetch fault for homeless sDSMs.
- \( C_w, C_f \) stand for cost of write and fetch fault for homeless sDSMs.
Aggregate SDP Model (1)

As the critical path SDP model requires detailed knowledge of the number and types of page faults occurring in each parallel region, a simplified SDP model is proposed to avoid this.

For any given application some fraction, \( f \), of the total page faults will be serialized, while the remaining \((1-f)\) fraction will be able to overlapped.

Generally, this is a simpler model, which can be used when detailed information on parallel regions is not available.
Aggregate SDP Model (2)

- Aggregate SDP model:

\[ T_{est}^{aggr} = \left( T(1)/p \right) + \left( N_w C_w + N_f C_f \right) f + \left( 1 - f \right)/p \]

- \( f \) is the serialized factor with a value between zero and one.
  - \( f = 0 \) is generally the best fit.
Page Fault Cost Measurements

As we assumed that cost of a page faults is constant on a given hardware platform, we can measure it by running a simple OpenMP program with different cluster-enabled implementations.

R is a private array, S is a shared array. npages is the size of R and S in pages. $D^w$, $D^r$ represent reference times for accessing private array R.

Number of page faults can be obtained by using a profiling tool, SEGVprof, for CLOMP.

1. $D^w \leftarrow \text{WRITE}(R)$
2. $D^r \leftarrow \text{READ}(R)$
3. READ(S)
4. Barrier
5. if thread-0:
6. WRITE(S)
7. Barrier
8. if thread-1:
9. $C_f \leftarrow (\text{READ}(S) - D^r)/\text{npages}$
10. $C_w \leftarrow (\text{WRITE}(S) - D^w)/\text{npages}$
## Experimental Setup

### Hardware

<table>
<thead>
<tr>
<th># of nodes</th>
<th>CPU</th>
<th>Clock speed</th>
<th>Memory</th>
<th>Interconnection</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>AMD Athlon dual core</td>
<td>2.2GHz</td>
<td>4GB</td>
<td>Giga-Ethernet</td>
</tr>
</tbody>
</table>

### Software

| CLOMP Intel C/Fortran Compiler 10.0 | Linux 2.6.11.4-21.11 kernel |

### Benchmarks

NPB-OMP 3.2 (Only class A and C of BT, SP, CG are shown in the presentation.)

---

<table>
<thead>
<tr>
<th>measured coefficient (usec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOMP</td>
</tr>
<tr>
<td>Cw</td>
</tr>
<tr>
<td>21.6</td>
</tr>
</tbody>
</table>
CLOMP Estimation Results (BT)

average absolute fractional error for Crit-Est is: BT-A 6%
BT-C 4%
CLOMP Estimation Results (SP)

average absolute fractional error for Crit-Est is: SP-A 8%
SP-C 4%

Observed
Crit-Est
Aggr(f=0)
CLOMP Estimation Results (CG)

Average absolute fractional error for Crit-Est is: CG-A 83% CG-C 11%
The SPD model estimates agree well with observed speedup. The aggregated SDP model is slightly worse than the critical path SDP model.

Estimates agree better with observations for larger problem sizes.

The estimation is getting more optimistic with increasing number of threads.

Overall, the SDP models are an easy rational model, which perform quite well for most of the applications.
Questions?

Acknowledgment:
This work is funded by Australian Research Council Grant LP0669726 and Intel Corporation.