A Dynamic Optimization Framework for OpenMP

Besar Wicaksono, Ramachandra C.N., and Barbara Chapman

High Performance Computing and Tools Group (HPCTools)
Computer Science Department
University of Houston, Texas
Agenda

• Introduction
• Overview of DARWIN
• Case study in ccNUMA data distribution optimization
• Experiments
• Summary
• Future work
Motivation

- OpenMP simplifies shared memory parallel programming
  - More productivity
- Scalability can be a challenge
  - Data locality
  - False sharing
- Needs a deep understanding about program’s dynamic behavior
OpenMP Collector API (1)

http://www2.cs.uh.edu/~hpctools/
OpenMP Collector API (2)

- Interface for profiling OpenMP program
- Enables performance tool to interact with OpenMP runtime library
- Event based communication
  - OMP_EVENT_FORK, OMP_EVENT_JOIN
  - OMP_EVENT_THR_BEGIN_IBAR, OMP_EVENT_THR_BEGIN_EBAR
- Hybrid of library and runtime instrumentation
  - OMPRTL is instrumented by calls to callback handler
  - Performance tool implements the action in the callback handler
- Baseline of the dynamic optimization framework
OpenMP Collector API (3)

Example:

```c
#pragma omp parallel for reduction (+:sum)
for(i=0; i < N ; i++)
    sum += a[i];
```

Serial State

- scheduling
- prepare for fork
Example:

```c
#pragma omp parallel for reduction (+:sum)
  for(i=0; i < N; i++)
    sum += a[i];
```

OpenMP Collector API (3)
Example:

```c
#pragma omp parallel for reduction (+:sum)
for(i=0; i < N ; i++)
    sum += a[i];
```

```
Example:
#pragma omp parallel for reduction (+:sum)
for(i=0; i < N ; i++)
    sum += a[i];
```
Example:

```c
#pragma omp parallel for reduction (+:sum)
for(i=0; i<N ; i++)
    sum += a[i];
```

```c
for(i=0; i<N ; i++)
    sum += a[i];
```
Overview of DARWIN

DARWIN

OpenMP Runtime

Persistent Storage

Collector Tool

Data Management

Data Allocation

Performance Monitoring

Optimizer

Utilities

Event notification
Overview of DARWIN

DARWIN

OpenMP Runtime

Persistent Storage

Event notification

Collector Tool

Retrieve data-structure information

Data Allocation

Performance Monitoring

Optimizer

Utilities

Data Management

UNIVERSITY of HOUSTON

11
Overview of DARWIN

DARWIN

OpenMP Runtime

Event notification

Collector Tool

Retrieve data-structure information

Start and stop hardware counters

Data Allocation

Performance Monitoring

Optimizer

Utilities

Data Management

Persistent Storage
Overview of DARWIN

OpenMP Runtime

Event notification

Collector Tool

- Create data-centric information
- Retrieve optimization strategy

Data Structure Information

Start and stop hardware counters

Data Allocation

Performance Monitoring

Optimizer

Utilities

Persistent Storage

Data Management
Overview of DARWIN

DARWIN

OpenMP Runtime

Event notification

Collector Tool

Retrieve data-structure information

Start and stop hardware counters

Apply optimization technique

Data Allocation

Performance Monitoring

Optimizer

Utilities

Data Management

Retrieve data-structure information

-Persistent Storage

-Create data-centric information

-Retrieve optimization strategy

Persistence
Overview of DARWIN

DARWIN

OpenMP Runtime

Event notification

Collector Tool

Retrieve data-structure information

Start and stop hardware counters

Apply optimization strategy

Data Allocation

Performance Monitoring

Optimizer

Utilities

Persistent Storage

Data Management

- Create data-centric information
- Retrieve optimization strategy

- Import to TAU profiles
- Insert analysis result

15
Case Study: ccNUMA Data Distribution Problem

• Main objectives:
  – Identify un-optimized data placement
  – Place data close to the processing node

• Two execution phases:
  – Monitoring
    • Initial run
    • Capture performance data for offline analysis
  – Optimization
    • Subsequent runs
    • Apply optimization technique
Monitoring Phase

- Capture per-thread memory references
  - thread id
  - parallel region id
  - data virtual memory address
  - data page number
  - latency
- Capture data allocation information
  - Variable name
  - Address
  - Size
- Associate each reference to its corresponding variable name

```c
int main()
{
    #pragma omp parallel
    {
        start profiling
        ....
    }
    stop profiling
    ....
    #pragma omp parallel
    {
        start profiling
        ....
    }
    stop profiling
    return 0;
}
```
Analysis (1)

- Visualize the collected information using TAU Paraprof
Analysis (2)

- Classify data access pattern type
  - Vertical axis & color code: references count
  - Horizontal axis: page number
  - Depth axis: thread id

Block access pattern for colidx on CG
Analysis (2)

- Classify data access pattern type
  - Vertical axis & color code: references count
  - Horizontal axis: page number
  - Depth axis: thread id

Block access pattern for *colidx* on CG

Cyclic access pattern for *rhs* on SP
Analysis (3)

- Identify un-optimized data placement
  - Vertical axis & color code: average latency
  - Horizontal axis: page number
  - Depth axis: thread id

Latency imbalance for colidx on CG
Analysis (3)

- Identify un-optimized data placement
  - Vertical axis & color code: average latency
  - Horizontal axis: page number
  - Depth axis: thread id

Latency imbalance for colidx on CG

rhs on SP is already optimized for ccNUMA
Analysis (4)

Cyclic access pattern for $rhs$ on SP

$rhs$ on SP is already optimized for ccNUMA
Analysis (4)

• Optimization strategy
  – Variable name
  – Access pattern type
  – Execution state
    • DARWIN’s initialization
      – Initial data placement
    • Start of a parallel region
      – Dynamic data, multiple access pattern types
  – Placement method
    • First touch
      – Initialization
    • Next touch
      – Beginning of a parallel region
Optimization Phase

- Capture data allocation information
- Get variables that need to be optimized
  - DARWIN’s initialization
  - Start of each parallel region
- Calculate the page numbers to distribute
  - Based on the access pattern type
- Call optimizer component
  - First touch: set affinity to the destination CPU & perform a write operation
  - Next touch: move_pages routine from the libnuma library

Example of applying an optimization strategy

Variable A, start address 0x1, size 100

Block access pattern type

1 - 25
26 - 50
51 - 75
76 - 100

TID-0
TID-1
TID-2
TID-3

First touch

Proc 1
Proc 2

Example of applying an optimization strategy
Experiment Setup

• NPB OpenMP C version
  – CG, MG, BT, FT, LU, IS, SP
  – Class A data size
  – Compiled with OpenUH, -O2

• SGI Altix 3700 ccNUMA Platform
  – 32 Nodes @ dual 1.3 GHz Itanium 2 processors
  – SUSE 10 Operating System

• Interactive PBS session
  – Two compute nodes
  – Four OpenMP threads
Monitoring Result (1)

• Captured parallel regions and data allocation

<table>
<thead>
<tr>
<th>Metric (counts)</th>
<th>CG</th>
<th>SP</th>
<th>FT</th>
<th>BT</th>
<th>LU</th>
<th>MG</th>
<th>IS</th>
</tr>
</thead>
<tbody>
<tr>
<td># of parallel regions</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td># of global/static data</td>
<td>41</td>
<td>126</td>
<td>39</td>
<td>128</td>
<td>76</td>
<td>27</td>
<td>33</td>
</tr>
<tr>
<td># of dynamic data</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>246797</td>
<td>0</td>
</tr>
</tbody>
</table>
Monitoring Result (2)

- Total overhead = $T_{\text{monitoring}} - T_{\text{original}}$
- Sampling period = 100 cache misses
- Generating data-centric information dominates the overhead
- Large data allocation affect the overhead of MG
Monitoring Result (3)

- Overhead with higher sampling period
  - Reduce the amount of collected data from hardware counter
  - Sampling period has no impact on data allocation
### Analysis Result

- **Optimization strategy for each program**

<table>
<thead>
<tr>
<th>Program</th>
<th>Variable Name</th>
<th>Access type</th>
<th>Execution State</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td>$a, colidx$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>SP</td>
<td>$lhs, forcing, u$</td>
<td>cyclic</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>FT</td>
<td>$u1$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>BT</td>
<td>$fjac, njac$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>LU</td>
<td></td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>MG</td>
<td>$main_211, main_222, main_235$</td>
<td>block</td>
<td>Par. Region 0</td>
<td>Next touch</td>
</tr>
<tr>
<td>IS</td>
<td>$key_array$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
</tbody>
</table>
Analysis Result

- Optimization strategy for each program

<table>
<thead>
<tr>
<th>Program</th>
<th>Variable Name</th>
<th>Access type</th>
<th>Execution State</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td>$a$, $colidx$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>SP</td>
<td>$lhs$, $forcing$, $u$</td>
<td>cyclic</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>FT</td>
<td>$u1$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>BT</td>
<td>$ffac$, $njac$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>LU</td>
<td></td>
<td></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>MG</td>
<td>$main_211$, $main_222$, $main_235$</td>
<td>block</td>
<td>Par. Region 0</td>
<td>Next touch</td>
</tr>
<tr>
<td>IS</td>
<td>$key_array$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
</tbody>
</table>
Analysis Result

- Optimization strategy for each program

<table>
<thead>
<tr>
<th>Program</th>
<th>Variable Name</th>
<th>Access type</th>
<th>Execution State</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td>$a, colidx$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>SP</td>
<td>$lhs, forcing, u$</td>
<td>cyclic</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>FT</td>
<td>$u1$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>BT</td>
<td>$fjac, njac$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
<tr>
<td>LU</td>
<td></td>
<td></td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>MG</td>
<td>$main_211, main_222, main_235$</td>
<td>block</td>
<td>Par. Region 0</td>
<td>Next touch</td>
</tr>
<tr>
<td>IS</td>
<td>$key_array$</td>
<td>block</td>
<td>DARWIN’s init</td>
<td>First touch</td>
</tr>
</tbody>
</table>
Optimization Result

- Speedup ($T_{\text{original}} / T_{\text{optimized}}$)
- Wall clock time = overhead + initialization + computation time
Optimization Result

- Overhead ($T_{\text{each\_overhead}} / T_{\text{optimized}}$)

<table>
<thead>
<tr>
<th>NPB Class A, 4 threads</th>
<th>% Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td>1.00%</td>
</tr>
<tr>
<td>SP</td>
<td>1.00%</td>
</tr>
<tr>
<td>FT</td>
<td>1.00%</td>
</tr>
<tr>
<td>BT</td>
<td>2.00%</td>
</tr>
<tr>
<td>LU</td>
<td>4.00%</td>
</tr>
<tr>
<td>MG</td>
<td>14.00%</td>
</tr>
<tr>
<td>IS</td>
<td>21.22%</td>
</tr>
</tbody>
</table>
Optimization Result

• Speedup ($T_{original} / T_{optimized}$)
• Wall clock time = overhead + initialization + computation time

![Speedup Graph](chart.png)

NPB Class A, 4 threads

Speedup (original/optimized):
- CG: 1.57
- SP: 1.72
- FT: 1.05
- BT: 1.14
- LU: 1.04
- MG: 0.99
- IS: 0.98

Wall clock time speedup:
- CG: 0.98
- SP: 1.04
- FT: 1.05
- BT: 1.14
- LU: 1.04
- MG: 0.99
- IS: 0.98
Optimization Result

- Speedup ($T_{\text{original}} / T_{\text{optimized}}$)
- Wall clock time = overhead + initialization + computation time

NPB Class A, 4 threads

NPB Class B, 8 threads
Summary

• DARWIN is built around various components
• Achieved good results on ccNUMA data distribution case study
• Major overheads
  – Creating data-centric information
  – Tracking dynamic data allocation
  – Applying next-touch method
Future Work

• More case studies
  – Currently working on false sharing

• Extending collector
  – Work sharing
  – Tasking support

• Reduce the overheads

• Compiler supported optimization

• Explore performance data collection on x86 platforms
  – AMD : IBS
  – Intel : PEBS
Questions ?