Enabling Multi-threaded Applications on Hybrid Shared Memory Manycore Architectures

Tushar Rawat and Aviral Shrivastava
Arizona State University, USA
Port Multi-threaded Applications

Multicore System → HSM Manycore System
Threading and Shared Memory

Multicore System

Process

Global Space

Thread

Threads have Shared and Implicit Access to Program Data
Data is not implicitly shared among processes
Convenience Hardware

Multicore System

Hardware-based cache coherence

HSM Manycore System
Convenience Hardware

Multicore System

- Hardware-based cache coherence

HSM Manycore System

- Small scratchpad-like on-chip memory
- Lacks hardware cache coherence
Contribution

- Identify shared data in multi-threaded program
- Map identified shared data to shared memory

| On-chip | Off-chip |
Five Stage Approach

Analysis

Variable Scope
Within Threads
Pointers
Partition Data
Thread To Process

Translation
Stage 1 – Variable Scope Analysis

- **Input**: Multi-threaded program source code
- **Output**: Name, size, type, read count and write count for each variable

```plaintext
int array[10];
array[0] = 6;
int foo = array[0];
```

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Size</th>
<th>Read Count</th>
<th>Write Count</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>array</code></td>
<td>int array</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><code>foo</code></td>
<td>int</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Tushar Rawat / Arizona State University
Stage 2 – Inter-thread Analysis

- **Input**: A given variable (name) e.g. “total_threads”
- **Output**: Result stating whether the given variable exists within 1 thread, 2+ threads or none

```c
void thread
...
printf("%d",
total_threads);
...  
pthread_exit(NULL);
```
• Variable w has a **Definite** relationship with ptr1

• Variables x and y have **Possibly** a relationship with ptr2

```
ptr1 = &w
ptr2 = &x
if-path
```

```
ptr2 = &y
else-path
```
Stage 4 – Data Partitioning

array = (double *)RCCE_malloc(size)

Processing Core(s)

RCCE_get

RCCE_put

On-chip shared memory (SRAM)

array = (double *)RCCE_shmalloc(size * sizeof(double))

Off-chip shared memory (DRAM)
Stage 5 – Program Translation

• Convert Pthread source to RCCE application code

<table>
<thead>
<tr>
<th>Pthread</th>
<th>RCCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>pthread_self()</td>
<td>RCCE_ue()</td>
</tr>
<tr>
<td>pthread_mutex_lock()</td>
<td>RCCE_acquire_lock()</td>
</tr>
<tr>
<td>pthread_create(&amp;thread, NULL,</td>
<td></td>
</tr>
<tr>
<td>funcName, (void *)arg)</td>
<td></td>
</tr>
</tbody>
</table>

Any remaining pthread code is removed from the source

• Add RCCE-specific instructions and libraries

<table>
<thead>
<tr>
<th>RCCE_init(argc, argv)</th>
<th>RCCE_finalize()</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCCE.h</td>
<td>RCCE_lib.h</td>
</tr>
<tr>
<td>SCC_API.h</td>
<td></td>
</tr>
</tbody>
</table>
On-chip shared memory - MPB

- CC: Cache Controller
- MPB: Message Passing Buffer
- MIU: Mesh Interface Unit
- P54C: Pentium® processor core

Tile

- 256 KB L2 cache
- MIU [to router]
- 16 KB MPB
- P54C Core L1 cache
- P54C Core L1 cache
Intel SCC Experiment Configuration

- Using 32 of 48 available cores
- 384 KB on-chip SRAM, up to 64 GB off-chip DRAM
- One Linux operating system *per core*
- 800 MHz core frequency
- 1600 MHz network mesh frequency
- 1066 MHz off-chip DDR3 frequency
Benchmarks

- Both Core and Memory intensive programs
- Originals developed for Pthread Multicore systems
- Compiled using Intel C++ Compiler 8.1 (gcc 3.4.5), RCCE API version 2.0
- Originals run on SCC for baseline
- Translated into SCC RCCE applications
  - Run with only off-chip shared memory
  - Run with mix of on-chip and off-chip shared memory
RCCE vs Pthread Performance

The diagram compares the performance of RCCE and Pthread on SCC for various tasks. The x-axis represents the different tasks: Count Primes, Dot Product, Pi Approximation, 3-5 Sum, LU Decomposition, Stream Add, Stream Copy, Stream Scale, and Stream Triad. The y-axis shows the normalized run time.
Off-chip vs On-chip Mem. Performance
Enabling Manycores – Performance

![Graph showing Pi Approximation Performance vs Number of Cores](image)

- Relative Performance
- Number of Cores

- 35
- 30
- 25
- 20
- 15
- 10
- 5
- 0

- 2
- 4
- 8
- 16
- 32

11-Mar-15

Tushar Rawat / Arizona State University
Takeaways

• Analyzer identifies all shared data within Pthread program
• Translator maps data to both on-chip and off-chip memory
• Enables execution of multi-threaded programs for HSM architecture after conversion to many-core applications
• Important to use fast on-chip memory when possible: 8x improvement on average for benchmarks when using on-chip SRAM (MPB) vs only off-chip DRAM
Thank you

Questions
Intel SCC and MCPC