Let’s Stay Close

Ruud van der Pas
Distinguished Engineer
SPARC Microelectronics
Santa Clara, CA, USA

SC’14 Talk at OpenMP Booth
Wednesday, November 19, 2014
We Will Look At The Big Picture (which can be an eye opener)
Let’s Stay Close

Ruud van der Pas
Put Things In The Right Perspective
Summary

(Dedicated To The Impatient)
Place: where threads can execute
Place: where threads can execute

Affinity Policy: where threads go
Why Worry?
A Generic cc-NUMA Architecture

Main Issue:
How To Distribute The Data?
Where Does Your Data Go?

The OS decides on the placement of data

A common default is to use the “First Touch” policy
Example First Touch Placement/1

```
for (i=0; i<100; i++)
a[i] = 0;
```

**First Touch**

*All array elements are in the memory of the processor executing this thread*
#pragma omp parallel for num_threads(2)

```c
for (i=0; i<100; i++)
a[i] = 0;
```

**First Touch**

*Both memories now have “their own half” of the array*
Thread Affinity – Machine Model

Cache Coherent System
Interconnect

socket 0

memory

core 0

hw thread 0

hw thread “m”

socket “k”

memory

core 0

hw thread 0

hw thread “m”

core “n”

hw thread 0

hw thread “m”

core “n”

hw thread 0

hw thread “m”
Why Thread Affinity Support?

For good performance and scalability it is key that

Threads are close to the data they need most often

Benefits

Reduces remote memory references

Maximizes bandwidth and reduces latency
Basic OpenMP Philosophy

Data is wherever it may be

Threads are moved to the data they need most
Two Key Concepts

The Place List

The Thread Affinity Policy
The Place List
A place consists of a (set of) numbers

Each number represents a scheduling unit

That is, something a thread can run on

For example, a hardware thread
Single element place

Multi element place
Example Multicore System

<table>
<thead>
<tr>
<th>Component</th>
<th>Count</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sockets</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Cores/socket</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Threads/core</td>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>
Example System Architecture

Cache Coherent System Interconnect

Memory

Core 0

- hw thread 0
- hw thread 1
- hw thread 2
- hw thread 3

Core 1

- hw thread 0
- hw thread 1
- hw thread 2
- hw thread 3

Socket 0

0
1
2
3
4
5
6
7

Socket 1

8
9
10
11
12
13
14
15
Places – Example

{0,1,2,3} identifies the threads in socket 0, core 0
Places – Example

\{0,1,2,3\} identifies the threads in socket 0, core 0

Convenient interval notation: \{0 : 4 : 1\}

start : count : stride
The order of the numbers within a place does not matter

\{0,1,2,3\} is the same as \{3,2,1,0\}

Assumption
No preference regarding memory access time
The Place List - Definition

The Place List consists of a comma separated list of places

For example: \{0,1,2,3\}, \{8,9,10,11\}

The Order Of The Places In The List DOES Matter

\{0,1,2,3\}, \{8,9,10,11\} \neq \{8,9,10,11\}, \{0,1,2,3\}
The Place List – How To Set It

Environment variable OMP_PLACES is used to define the place list

Example: OMP_PLACES="{0,1,2,3}, {8,9,10,11}"

The interval notation is very convenient

Example: OMP_PLACES="{0:4:1}, {8:4:1}"
The Place List – Abstract Names

Three abstract names are always available:

- **sockets**
- **cores**
- **threads**

Example: `OMP_PLACES=cores`

Example: `OMP_PLACES="cores(4)"

Note: Implementation can add names
Example – OMP_PLACES

**OMP_PLACES=cores**

*Equivalent To*

**OMP_PLACES=“{0,1,2,3},{4,5,6,7},{8,9,10,11},{12,13,14,15}”**

*Equivalent To*

**OMP_PLACES=“{0:4:1},{4:4:1},{8:4:1},{12:4:1}”**

*Equivalent To*

**OMP_PLACES = “{0:4:1}:4:4”**
The Affinity Policy
Affinity Policy: Which Place(s) To Use

- Balcony Place
- Hamburger Place
- Field Place
- Cheap Place

Illega Place: 4, 5
Cheap Place: 1
Field Place: 0
Balcony Place: 3
Hamburger Place: 6, 7, 8, 9, 10, 11
## Thread Affinity Policies

The Affinity Policy defines which places to use

This is defined in a symbolic way:

*master, close, or spread*

Each parallel region has such an affinity policy

Thread binding to a place is implied
OpenMP

Booth
–

Let’s Stay Close

Ruud van der Pas

32

Master Policy

Master Thread
### Setting The Thread Affinity Policy

- **Defined through OMP_PROC_BIND**

  Example: `OMP_PROC_BIND="spread,close"`

- **Can also use the “proc_bind” clause**

  Applied to the current parallel region
Recap - Places And Affinity

The Place List defines what is available

(fixed for the duration of the program)

The Affinity Policy defines thread placement

(can be set for each parallel region)
## Example – Spread Policy

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OMP_PLACES=cores</code></td>
<td>(this means 4 places in the place list)</td>
</tr>
<tr>
<td><code>OMP_PROC_BIND=spread</code></td>
<td></td>
</tr>
<tr>
<td><code>OMP_NUM_THREADS=4</code></td>
<td></td>
</tr>
</tbody>
</table>

**Result:** One OpenMP thread per place
Example Spread Affinity Policy
Thank You And ..... Stay Tuned!

ruud.vanderpas@oracle.com