

OpenMP on cc-NUMA Architectures

Christian Terboven, Dieter an Mey, Dirk Schmidl
{terboven, anmey, schmidl}@rz.rwth-aachen.de

Center for Computing and Communication
RWTH Aachen University

Agenda

- Introduction and Motivation
- Explicit Thread and Data Placement
- Issues with Multi-level Parallelism
- Summary and Outlook

2

Center for

Computing and
Communication

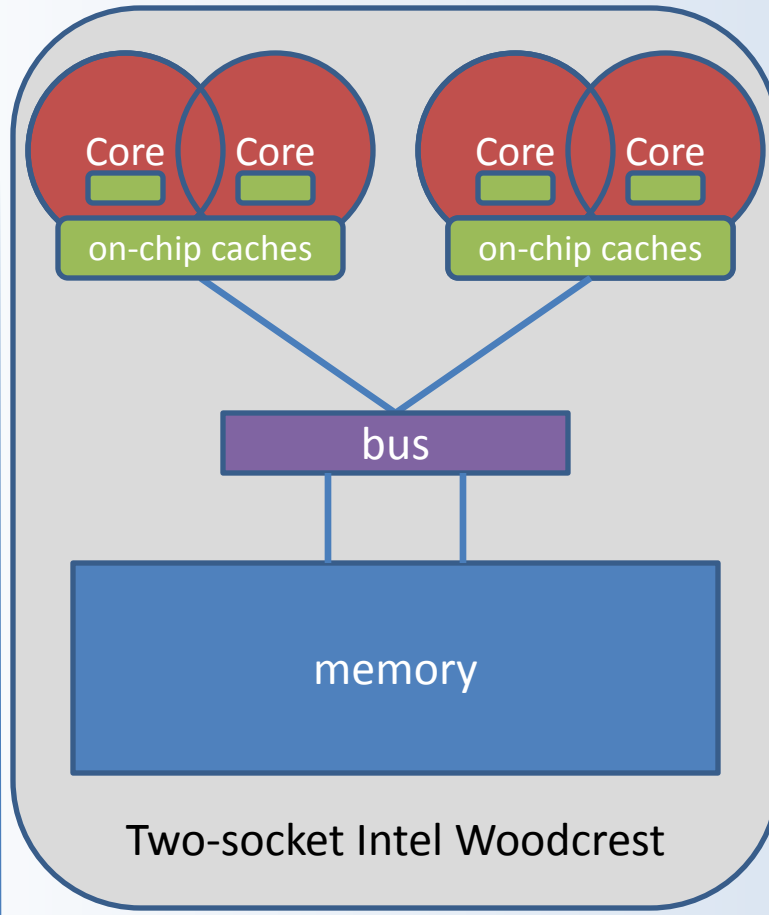
Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

The OpenMP Memory Model

- One Shared Memory, Private Memory per thread → equal distance from each core to the whole memory assumed



3

Center for

Computing and
Communication

Introduction

Data and Thread
Placement

Multi-level
Parallelism

Summary

Current Computer Architectures

- cc-NUMA
 - It is hard (= expensive) to build large SMPs offering a flat memory architecture with low latency and high bandwidth
 - Large Shared-Memory systems have physically distributed memories connected by a cache-coherent interconnect
 - It is profitable (= important) to keep threads close to data
- CMP: chip multi-processing
 - Recent processors contain multiple cores sharing the path to the memory, different cache layouts possible
- CMT/SMT: chip multi-threading, simultaneous multi-thr.
 - Some cores are capable of executing multiple instructions streams (= HW threads) (quasi) simultaneously

4

Center for

Computing and

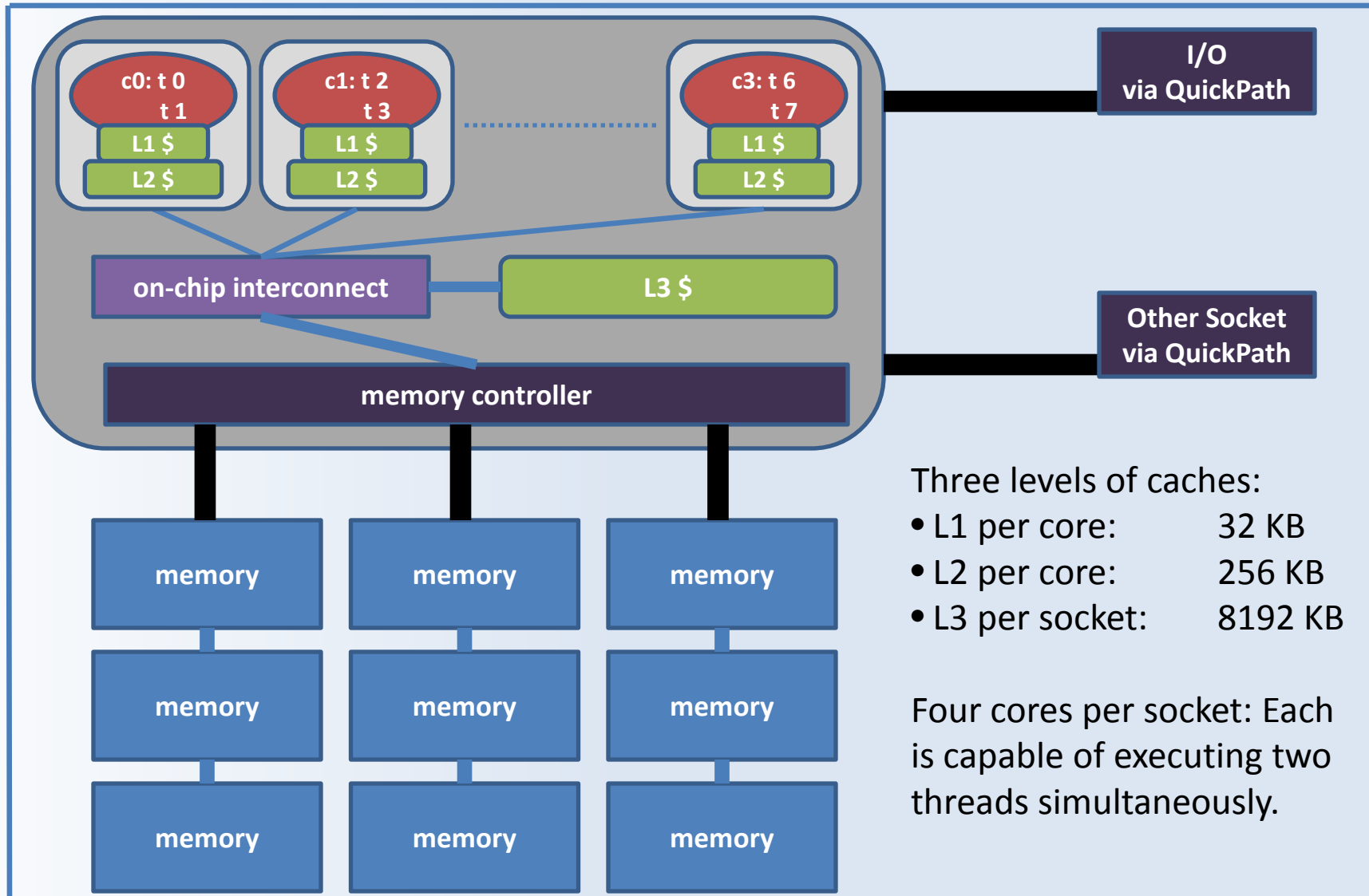
Communication

Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

Intel's Nehalem-EP (Zoomed-In)



Three levels of caches:

- L1 per core: 32 KB
- L2 per core: 256 KB
- L3 per socket: 8192 KB

Four cores per socket: Each is capable of executing two threads simultaneously.

5

Center for

Computing and
Communication

Introduction

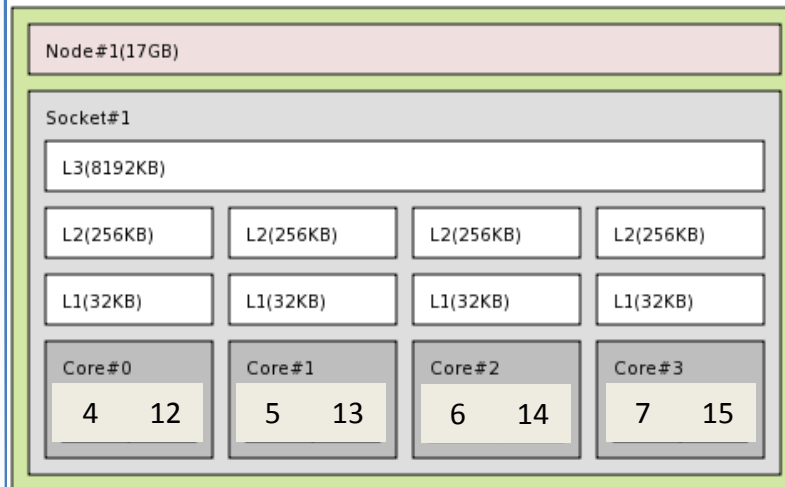
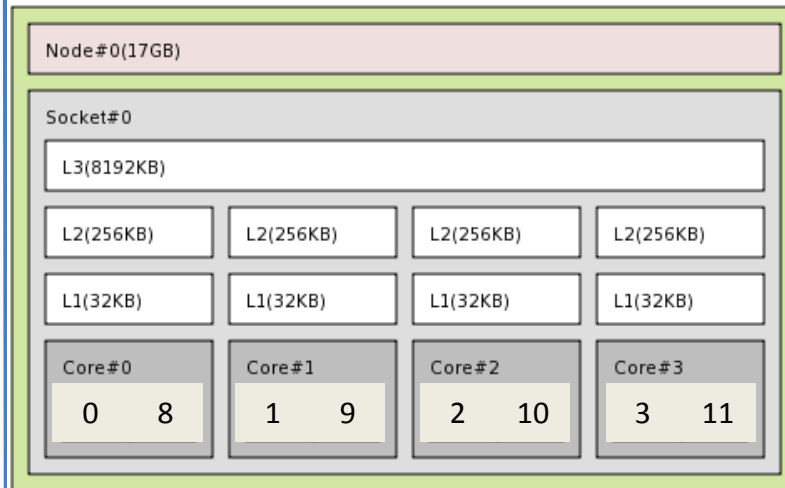
Data and Thread
Placement

Multi-level
Parallelism

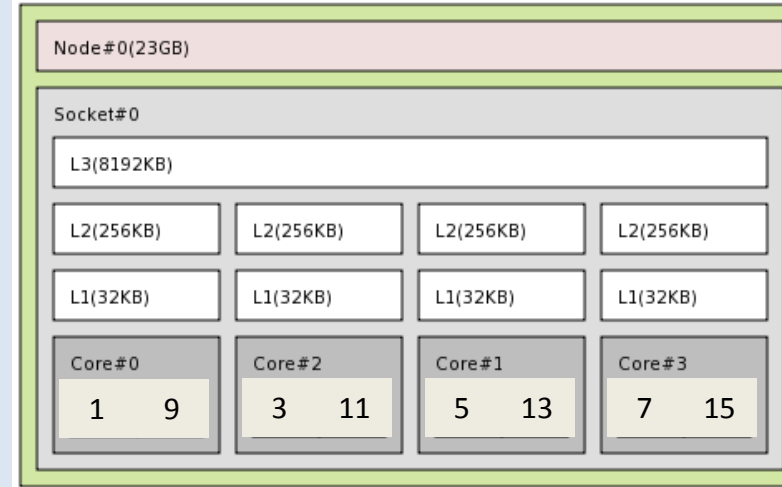
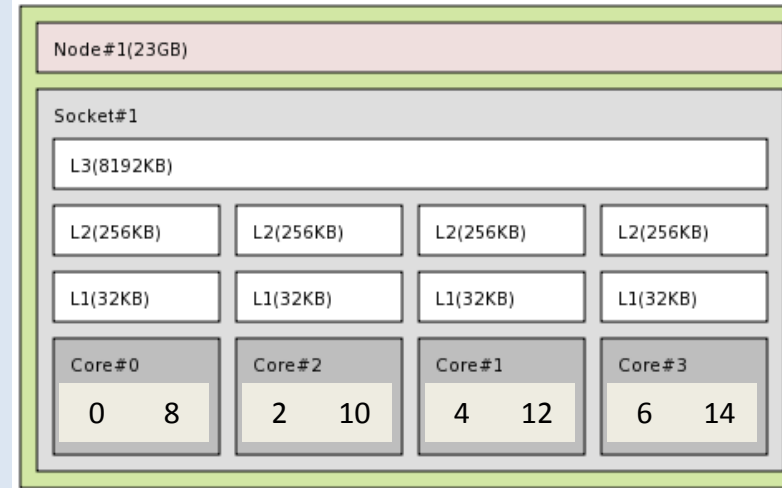
Summary

Intel's Nehalem-EP (Zoomed-Out: Sun vs. HP)

2-socket system from Sun



2-socket system from HP



6

Center for

Computing and
Communication

Introduction

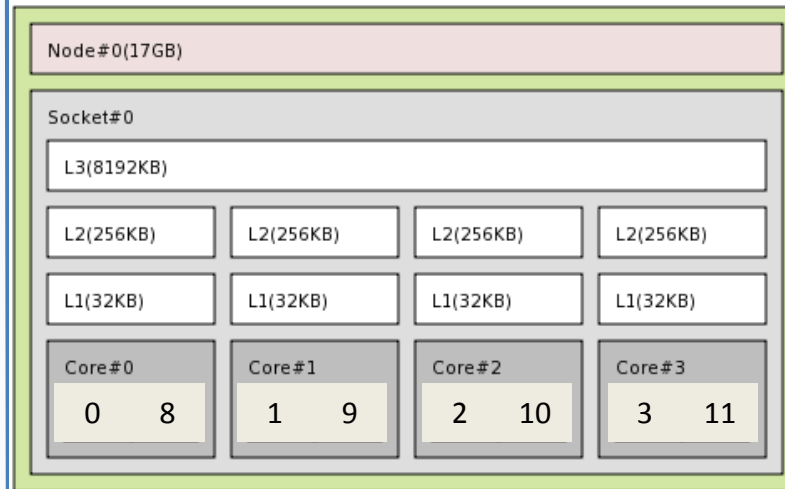
Data and Thread
Placement

Multi-level
Parallelism

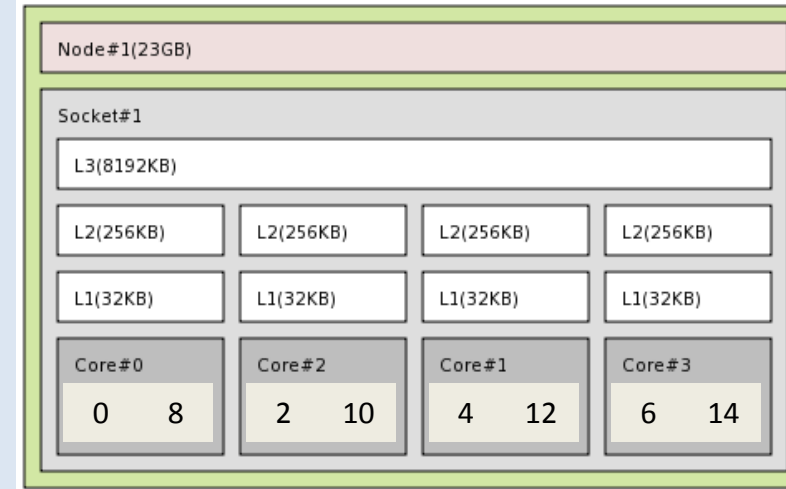
Summary

Intel's Nehalem-EP (Zoomed-Out: Sun vs. HP)

2-socket system from Sun



2-socket system from HP



- Systems exhibit slight differences that may have a huge impact on the performance
- Do not hard-code architecture details in your app, instead follow a strategy-based approach where possible
- Intel Compilers: `KMP_AFFINITY` environment variable controls how threads are placed, i.e. *compact* versus *scatter*

7

enter for

Computing and
Communication

Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

Agenda

- Introduction and Motivation
- Explicit Thread and Data Placement
- Issues with Multi-level Parallelism
- Summary and Outlook

8

Center for

Computing and

Communication

Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

User Control of Affinity (1/2)

- Current operating systems employ the first-touch strategy: Data is placed closest to the thread that touches it first.
- The OS may do a reasonable job,
 - if the machine is not overloaded,
 - and the first-touch policy has been carefully taken into account,
 - and the program does not change the memory access pattern.
- There are options for additional user control:
 - Explicit binding of threads to processors:
 - By environment variables, i.e. `KMP_AFFINITY`
 - By user commands
 - Linux: `taskset, numactl`
 - Windows: `start /affinity`

9

enter for

Computing and

Communication

Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

User Control of Affinity (2/2)

- There are options for additional user control:
 - Explicit binding of threads to processors (cont'd):
 - By system calls
 - Linux: `sched_setaffinity()`
 - Windows: `SetThreadAffinityMask()`
 - Explicit numa-aware memory allocation:
 - By carefully touching data by the thread which later uses it
 - By changing the default memory allocation strategy
 - Linux: `numactl` command
 - Windows: `VirtualAllocExNuma()` (limited functionality)
 - By explicit migration of memory pages
 - Linux: `move_pages()`
 - Windows: no option

10

Enter for

Computing and

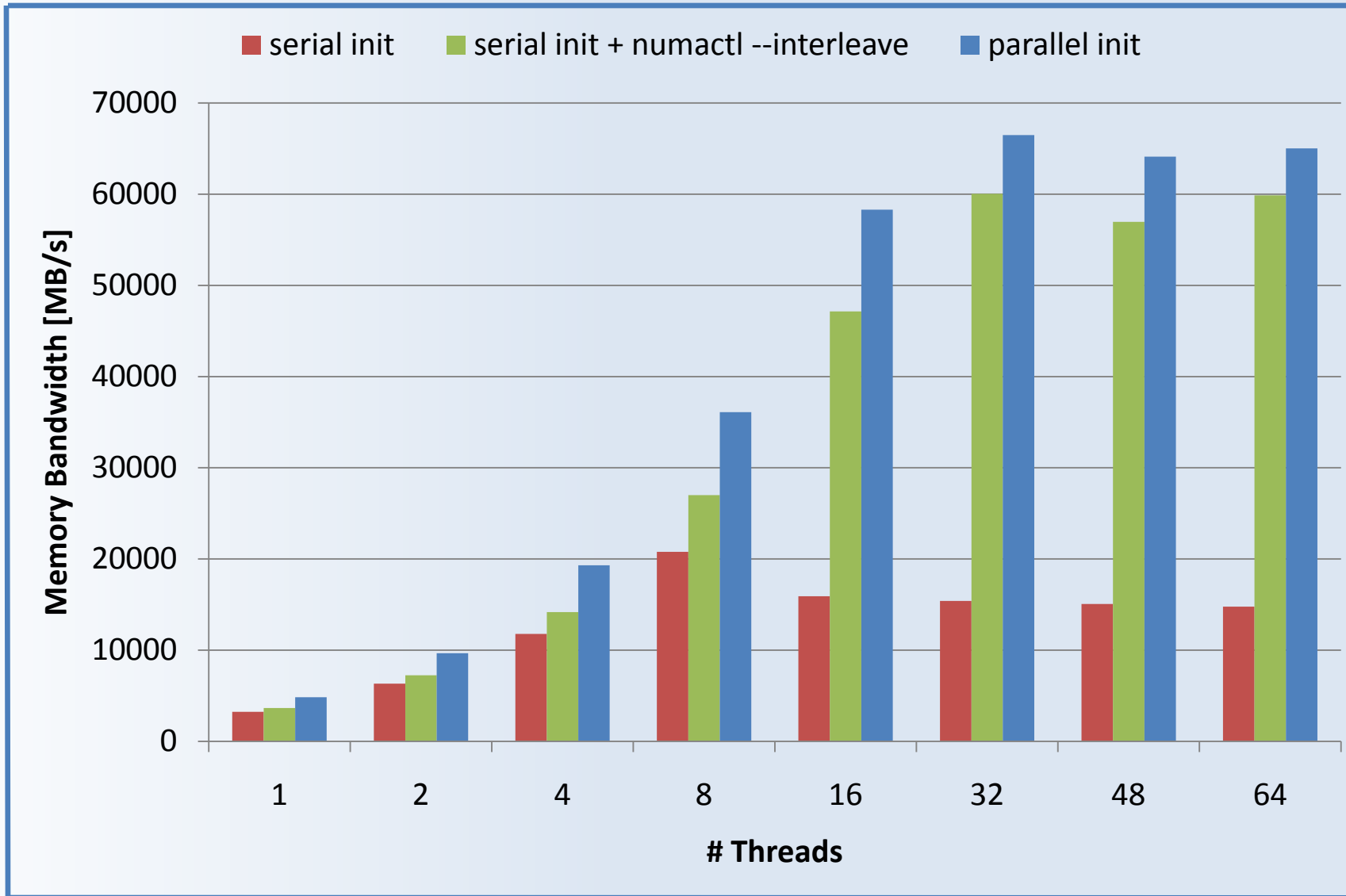
Communication

Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

STREAM-Triad on Nehalem-EX (1/2)



11

enter for

Computing and
Communication

Introduction

Data and Thread
Placement

Multi-level
Parallelism

Summary

STREAM-Triad on Nehalem-EX (2/2)

- Missing the parallel initialization can spoil all the fun ...

```
//allocation of arrays
double *a, *b, *c;
a, b, c = (double*) malloc(N*sizeof(double));
//parallel initialization: data allocated where used
→ # pragma omp parallel for ←
for (i=0;i<N;i++) a[i]=...=0.0;
// triad-ing with optimal memory placement
#pragma omp parallel for
for(i=0;i<N;i++) a[i]=b[i]+scalar*c[i];
```

- If data is setup in serial, but the computation is parallel, the data to thread affinity may be very bad
 - Either take care explicitly by binding + first-touch parallel init
 - Or apply random / round robin data placement
- Support by current multi-threading paradigms is missing!

12

enter for

Computing and

Communication

Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

A first Summary

- Everything under control?
- In principle Yes, but only if
 - threads can be bound explicitly,
 - data can be placed well by first-touch, or can be migrated,
 - you focus on a specific platform (= os + arch) → no portability
- What if the data access pattern changes over time?
- What if you use more than one level of parallelism?

13

Center for

Computing and

Communication

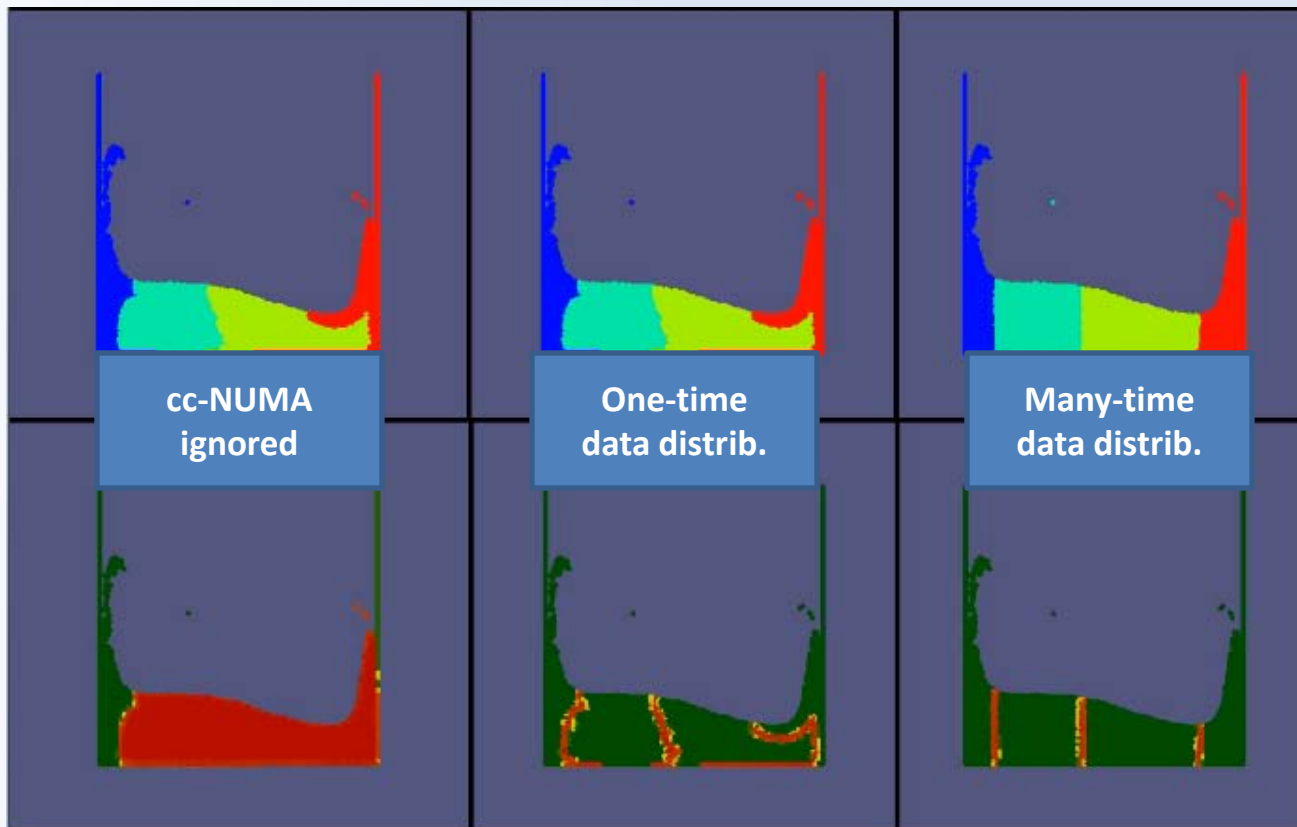
Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

SPH

- SPH = Smoothed Particle Hydrodynamics, fluid simulation
- Water is represented as a set of (many) particles
- Domain / Grid changes over time ...



Every color represents one thread being executed on one socket

Green = local access
Red = remote access

Agenda

- Introduction and Motivation
- Explicit Thread and Data Placement
- **Issues with Multi-level Parallelism**
- Summary and Outlook

15

Center for

Computing and

Communication

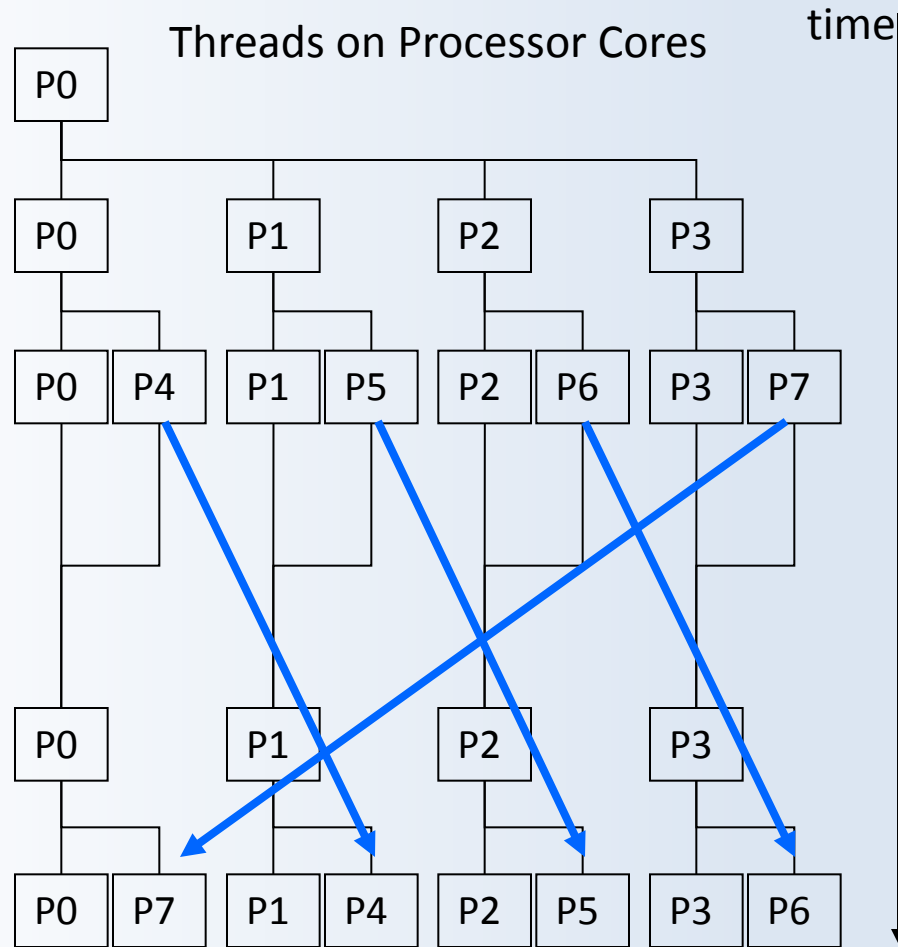
Introduction

Data and Thread
Placement**Multi-level
Parallelism**

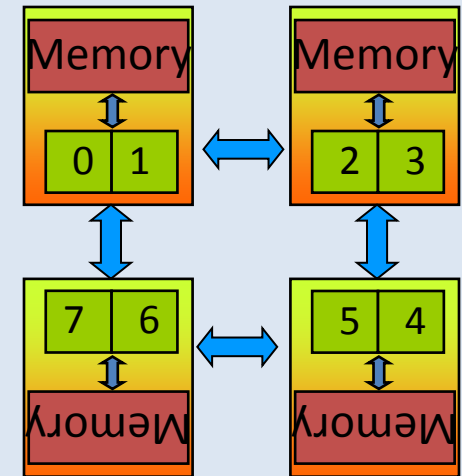
Summary

OpenMP Nested (1/3)

- The OpenMP runtime organizes the (OS) threads in a pool.



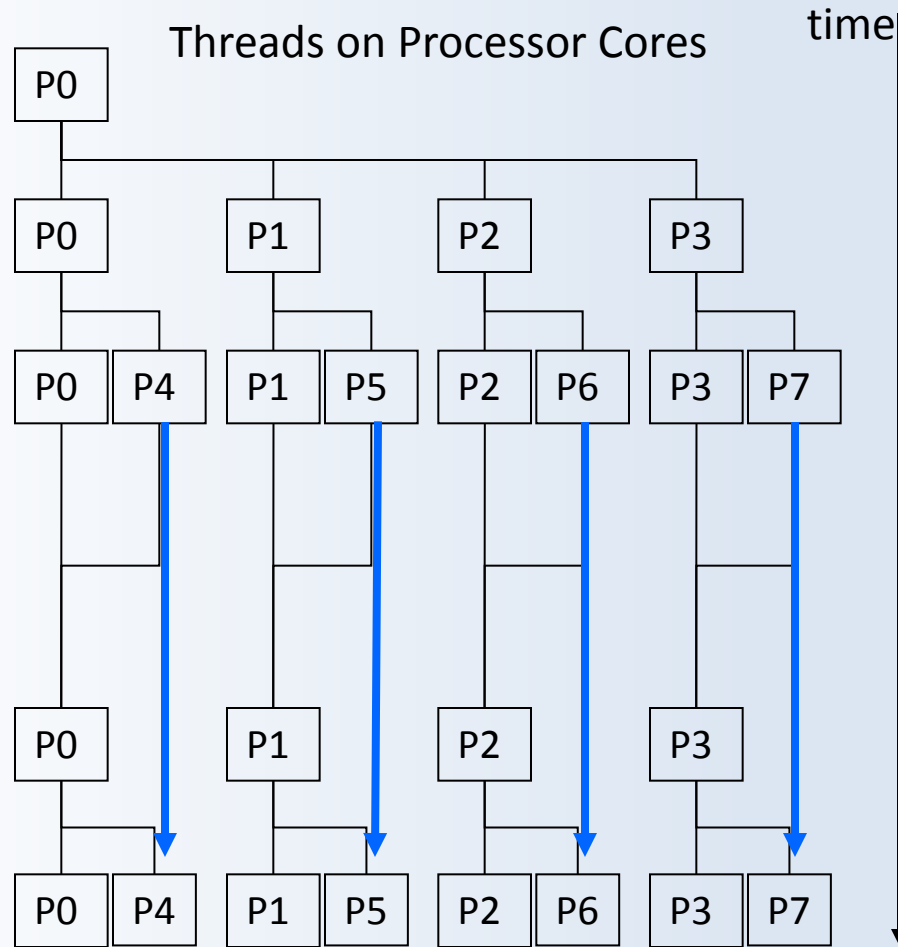
16



enter for

OpenMP Nested (2/3)

- o Experimental extension implemented in Sun Studio compilers



`SUNW_MP_PROCBIND=true`

Binds threads to cores in the order in which they were created (similar features are available for almost all OpenMP compilers)

`SUNW_MP_THR_AFFINITY=true`

Maintains thread assignments (on the cost of possible resource waste)

17

enter for

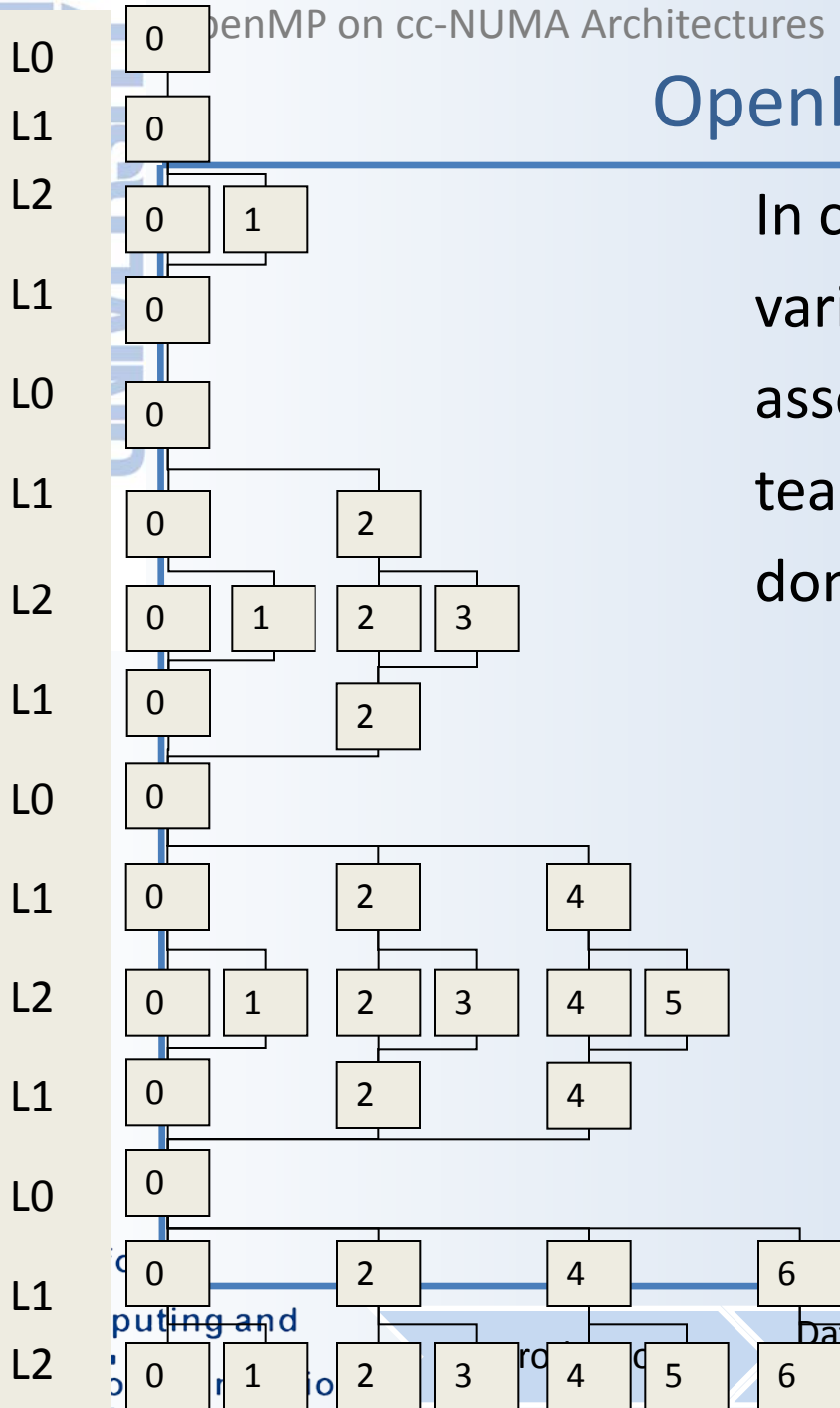
OpenMP Nested (3/3)

In combination with both environment variables a short loop takes care of associating the threads of the inner teams to cores in the same locality domain.

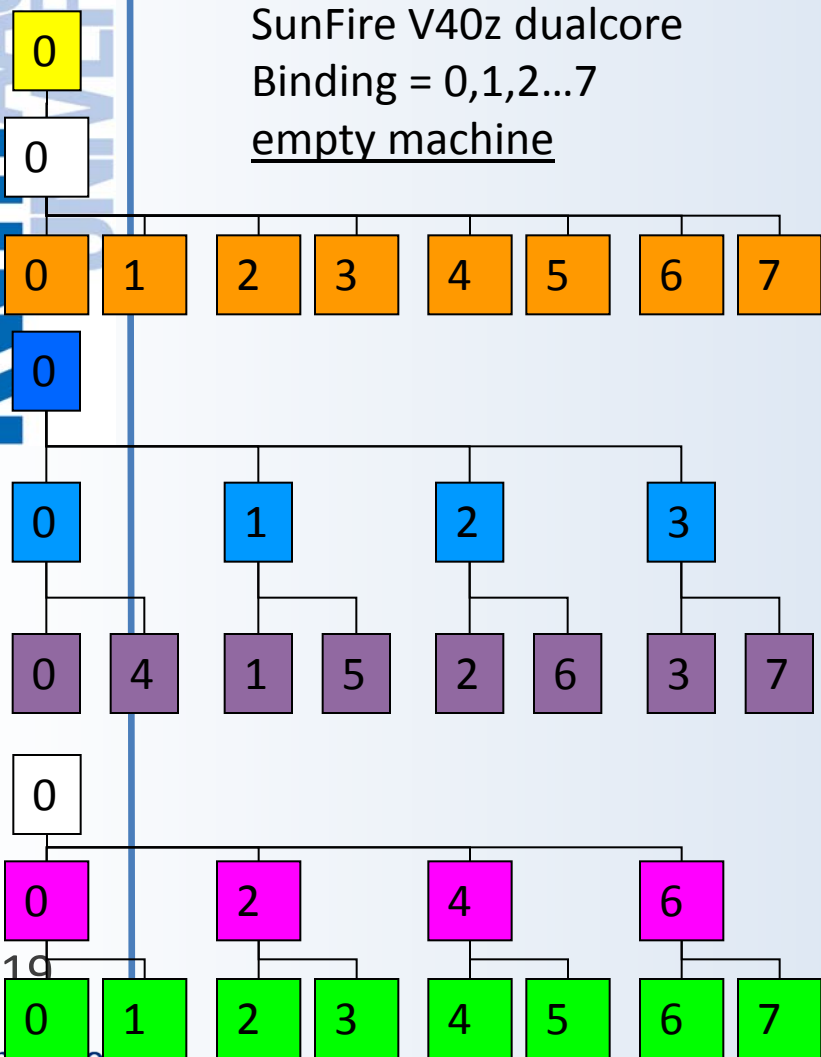
```

do i = 1, 4
!$omp parallel num_threads(i)
!$omp   parallel num_threads(2)
        continue
!$omp   end parallel
!$omp end parallel

```



Nested STREAM-Triad



#thrds	First touch	Affinity	min MB/s	max MB/s
1 x 8	Initial thread	n.a.	2525	2534
1 x 8	All inner threads	n.a.	11786	11869
4 x 2	Initial thread	no	4x629	4x631
4 x 2	Initial thread	yes	4x628	4x631
4 x 2	Inner master	no	4x1312	4x1332
4 x 2	Inner master	yes	4x1329	4x1334
4 x 2	Inner master	Yes+sort	4x2881	4x2943
4 x 2	All inner threads	no	4x2640	4x2948
4 x 2	All inner threads	yes	4x2893	4x2950
4 x 2	All inner threads	Yes + sort	4x2922	4x2934

ScaleMP: Overview

- These issues become more severe on larger machines...
- The ScaleMP vSMP software employs virtualization techniques to run a single system image (Linux) on a set of nodes connected via InfiniBand.
- The machine at RWTH Aachen University:
 - 13 boards connected via InfiniBand
 - 13x 2x Intel Xeon E5420 (Harpertown, 2.5 GHz)
 - 13x 16 GB of main memory
 - 38 GB used for management and automatic page migration
 - 170 GB available for applications

20

enter for

Computing and

Communication

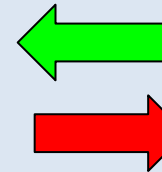
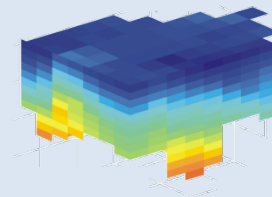
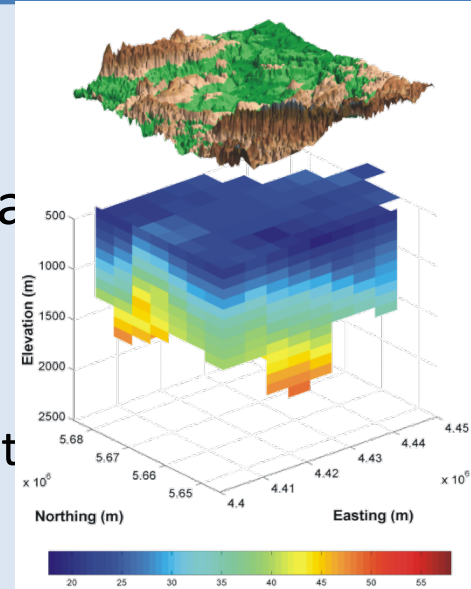
Introduction

Data and Thread
PlacementMulti-level
Parallelism

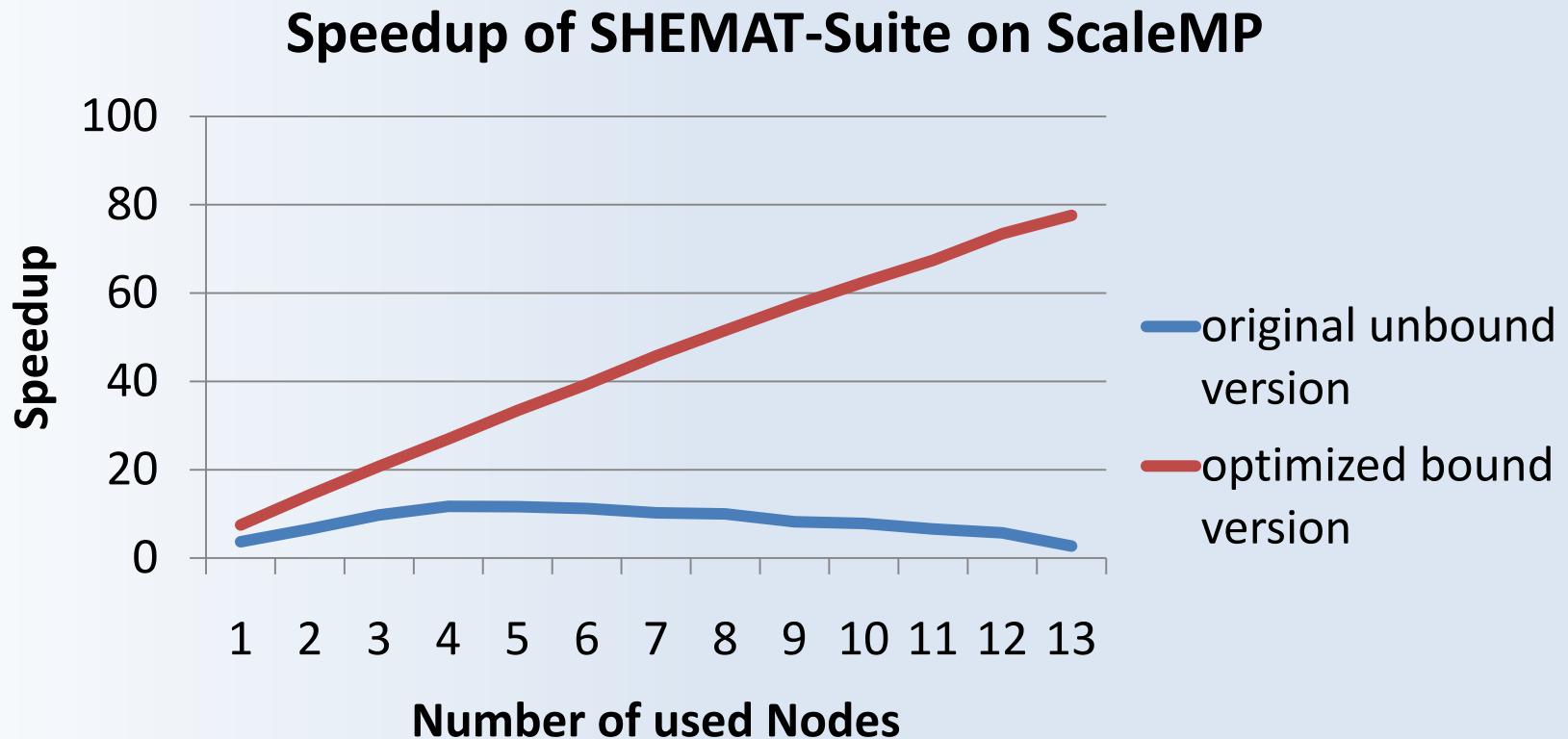
Summary

Application Case Study: SHEMAT-Suite

- Geothermal Simulation Package to simulate groundwater flow, heat transport, and the transport of reactive solutes in porous media at high temperatures (3D)
 - Forward simulation
 - ← • 3D finite-differences solver, Coupled transient equations for groundwater flow, Compute state variables from rock properties
 - Inverse computation
 - • Parameter estimation
- Written in Fortran, two levels of parallelism
 - Independent Computations of the Directional Derivatives
 - Setup and Solving of linear equation systems



ScaleMP



Optimizations:

1. Binding threads to let inner teams run on one board.
2. Avoiding unnecessary temporal data usage.

22

Center for

Computing and

Communication

Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

Agenda

- Introduction and Motivation
- Explicit Thread and Data Placement
- Issues with Multi-level Parallelism
- Summary and Outlook

23

Center for

Computing and

Communication

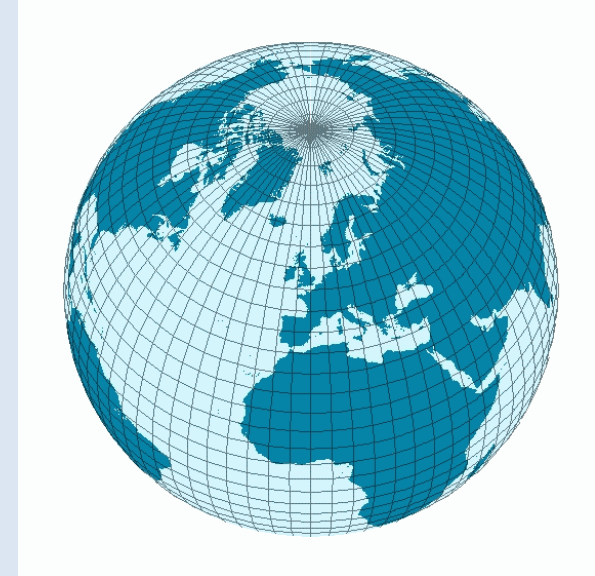
Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

The final Summary

- Affinity support is essential for OpenMP (or any other Shared-Memory paradigm) on cc-NUMA machines!
- Using OpenMP 3.0 on Linux it is possible to solve most issues
- Windows offers less functionality
- Open issues:
 - Better support for Nested Parallelism
 - Interaction with Resource Management Systems
 - Make the application aware of it's intended part of the machine
 - Provide guidance for the (OpenMP) runtime



Making OpenMP 3.1 / 4.0 ready for cc-NUMA

- Our favorite proposal:
 1. Do not describe the HW architecture within OpenMP
 2. Provide means to describe the structure of a (nested) OpenMP program as a series of thread trees
 3. Provide means to pass this information along with some guidance in form of strategies to the runtime
 4. Allow the runtime to take care of the mapping of threads to cores / the HW (analogous to the MPI topology concept)
 5. Guarantee threadprivate persistence as long as the tree remains constant
 6. Provide means for first-touch / random placement and migration to enable portability

25

Center for

Computing and

Communication

Introduction

Data and Thread
PlacementMulti-level
Parallelism

Summary

The End

Thank you for
your attention!

26

Backup: Compact vs. Scatter on Nehalem

