Binding Nested OpenMP Programs on Hierarchical Memory Architectures

Dirk Schmidl, Christian Terboven,
Dieter an Mey, and Martin Bücker

{schmidl, terboven, anmey}@rz.rwth-aachen.de
buecker@sc.rwth-aachen.de

Agenda

- **Thread Binding**
  - special isues concerning nested OpenMP
  - complexity of manual binding
- **Our Approach to bind Threads for Nested OpenMP**
  - Strategy
  - Implementation
- **Performance Tests**
  - Kernel Benchmarks
  - Production Code: SHEMAT-Suite

Advantages of Thread Binding

- "first touch" data placement only makes sense when threads are not moved during the program run
- faster communication and synchronization through shared caches
- reproducible program performance

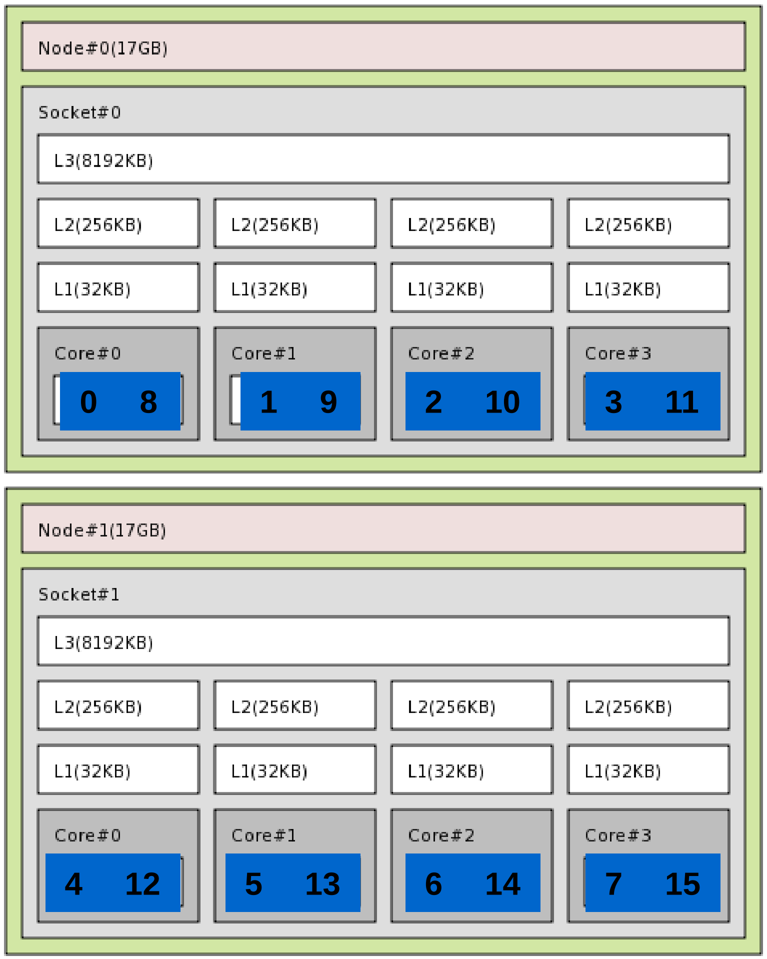
Thread Binding and OpenMP

1. **Compiler dependent Environment Variables**
   (KMP_AFFINITY, SUNW_MP_PROCBIND, GOMP_CPU_AFFINITY,…)
   - not uniform
   - nesting is not well supported

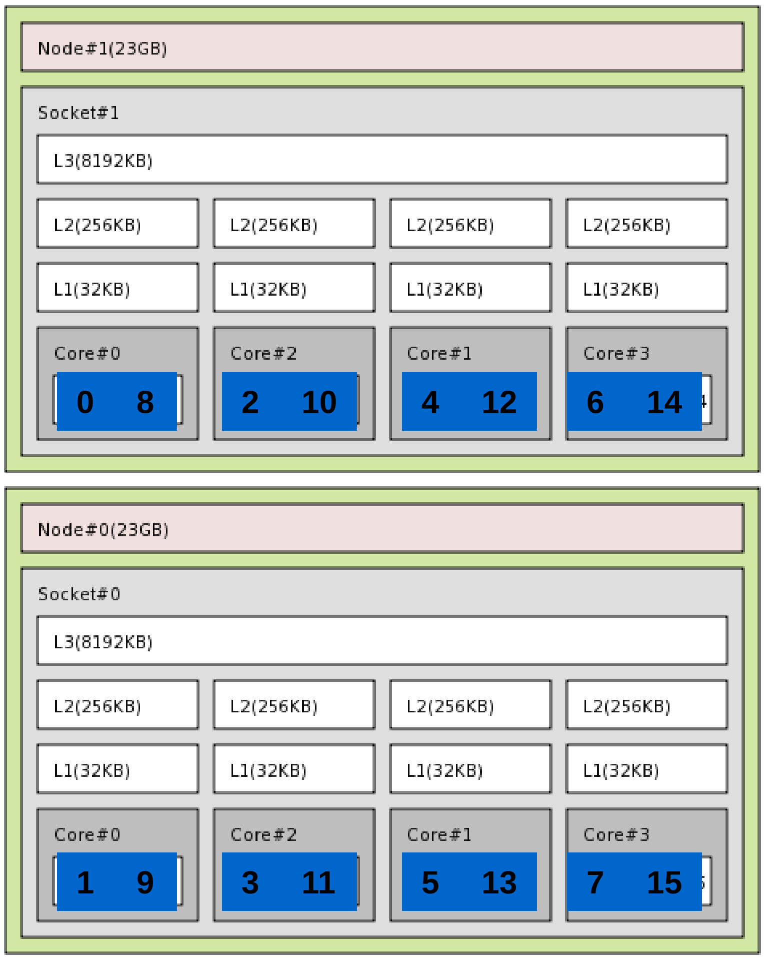
2. **Manual Binding through API Calls**
   (sched_setaffinity,…)
   - only binding of system threads possible
   - Hardware knowledge is needed for best binding

Numbering of cores on Nehalem Processor

2-socket system from Sun



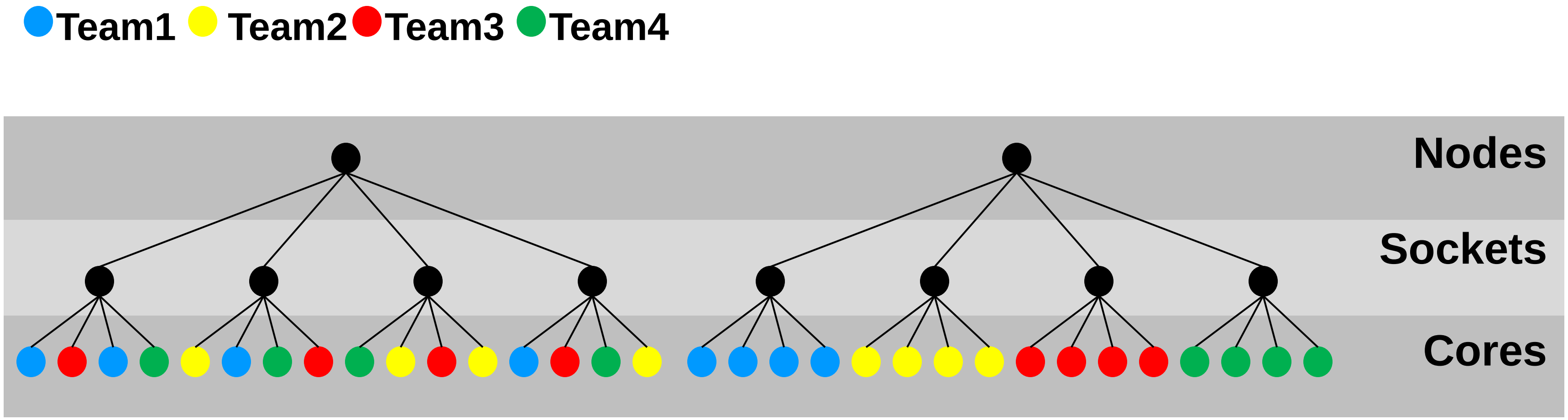
2-socket system from HP



Nested OpenMP

- most compilers use a “thread pool”, so not always the same system threads are taken out of this pool

- more synchronization and data sharing within the inner teams
=> higher importance where to place the threads of an inner team



Thread Binding Approach

Goals:

- easy to use
- no detailed hardware knowledge needed
- user interaction possible in an understandable way
- support for nested OpenMP

Thread Binding Approach

Solution:

- **user provides simple Binding Strategies (scatter, compact, subscatter, subcompact)**
  - environment variable :

    OMP_NESTING_TEAM_SIZE=4,scatter,2,subcompact
  - function call: omp_set_nesting_info("4,scatter,2,subcompact");

- **hardware information and mapping of threads to the hardware is done automatically**

- **affinity mask of the process is taken into account**

Binding Strategies

<u>compact</u>: bind threads of the team close together

- if possible the team can use shared caches and is connected to the same local memory

<u>scatter</u>: spread threads far away from each other

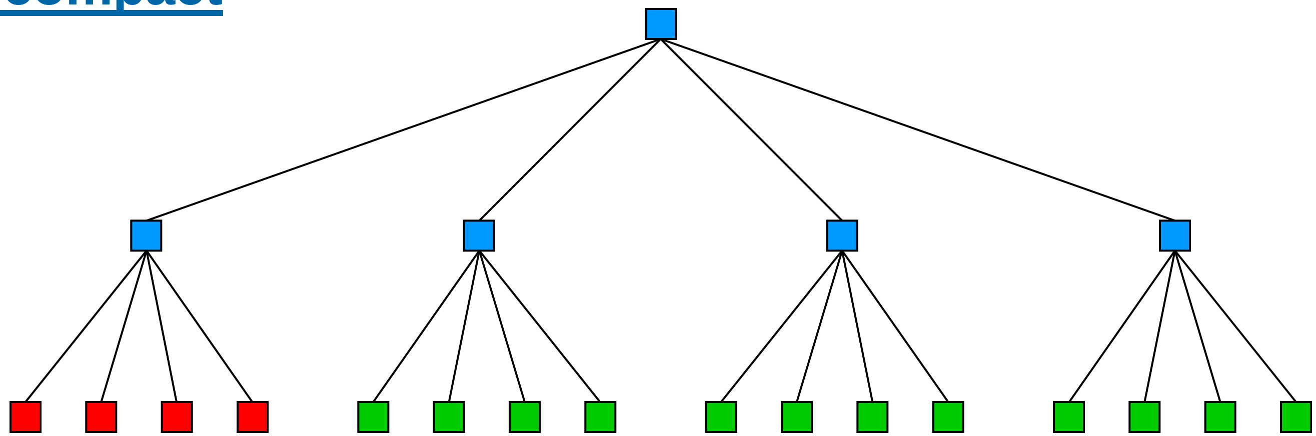
- maximizes memory bandwidth by using as many NUMA nodes as possible

<u>subcompact/subscatter</u>: run close to the master thread of the team, e.g. on the same board or socket and use the scatter or compact strategy on the board or socket

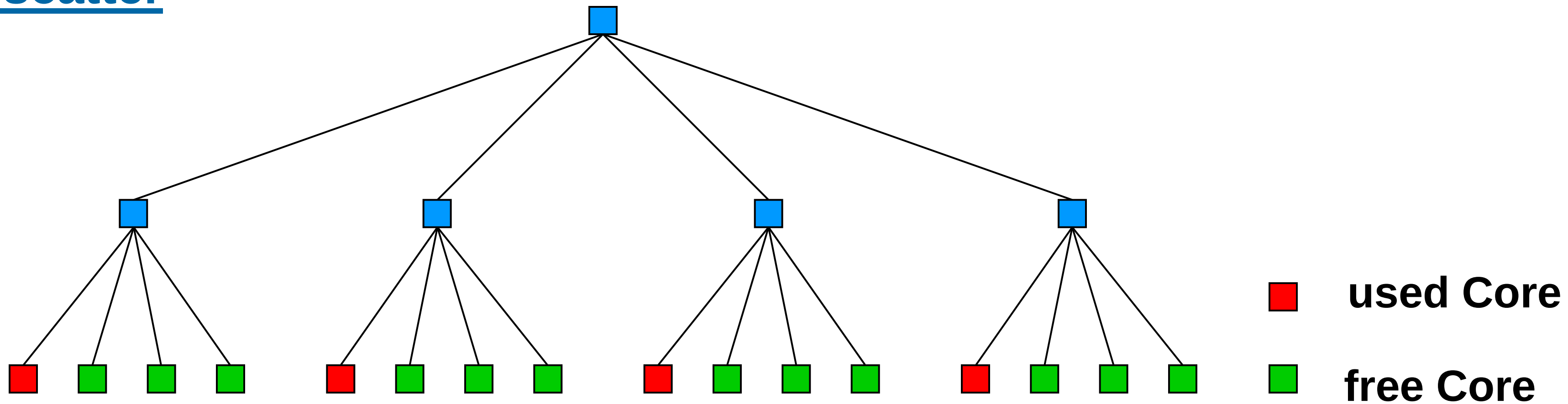
- data initialized by the master can still be found in the local memory

Binding Strategies

4,compact

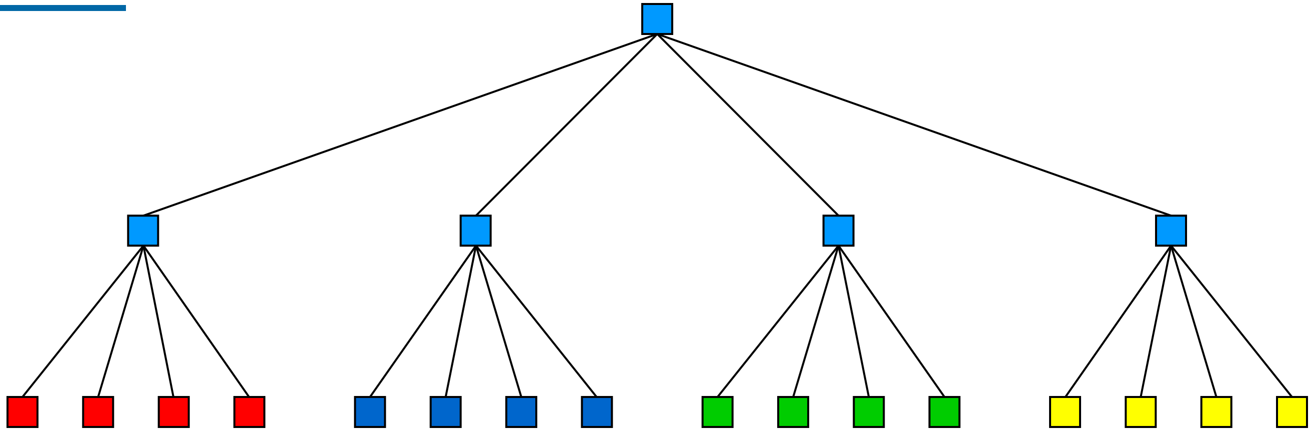


4,scatter

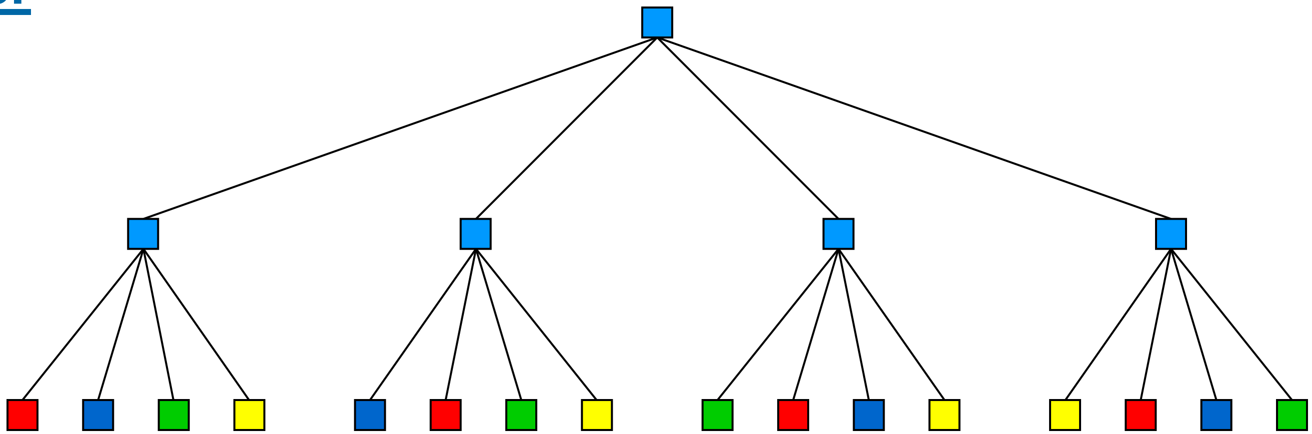


Binding Strategies

4,scatter,4,subscatter



4,scatter,4,scatter



Binding-Library

1. Automatically detect hardware information from the system
2. Read environment variables and map OpenMP threads to cores respecting the specified strategies
3. Binding needs to be done every time a team is forked since it is not clear which system threads are used
   - instrument the code by OPARI
   - provide a library which binds threads in **`pomp_parallel_begin`**() function using the computed mapping
4. Update the mapping after omp_set_nesting_info()

Used Hardware

1. **Tigerton (Fujitsu-Siemens RX600):**
    1. 4 x Intel Xeon X7350 @ 2,93 GHz
    2. 1 x 64 GB RAM
2. **Barcelona (IBM eServer LS42):**
    1. 4 x AMD Opteron 8356 @2,3 GHz
    2. 4 x 8 = 32 GB RAM
3. **ScaleMP**
    1. 13 board connected via Infiniband
    2. each 2 x Intel Xeon E5420 @ 2,5 GHz
    3. cache coherency by virtualization software (vSMP)
    4. 13 x 16 = 208 GB RAM

       ~38 GB reserved for vSMP = 170 GB available

SMP

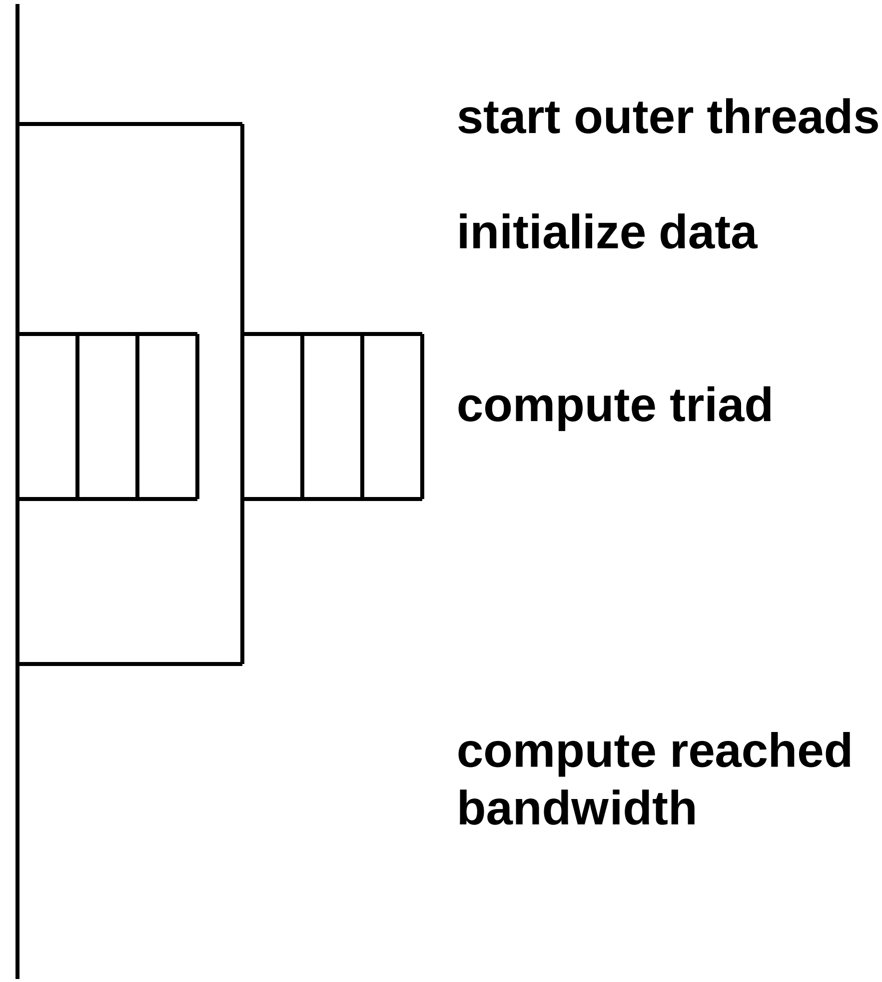
cc-NUMA

cc-NUMA with high NUMA ratio

“Nested Stream”

- modification of the STREAM benchmark

- start different teams
- each inner team computes STREAM benchmark
- separate data arrays for every inner team
- compute totally reached memory bandwidth



“Nested Stream”

threads	1x1	1x4	4x1	4x4	6x1	6x4	13x1	13x4
Barcelona								
unbound	4.4	4.9	15.0	10.7				
bound	4.4	7.6	15.8	13.1				
Tigerton								
Unbound	2.3	6.0	4.8	8.7				
Bound	2.3	3.0	8.2	8.5				
ScaleMP								
Unbound	3.8	10.7	11.2	1.7	9.0	1.6	3.4	2.4
bound	3.8	5.9	14.4	18.8	20.4	15.8	43.0	27.8

Memory bandwidth in GB/s of the nested Stream benchmark.
Y,scatter,Z,subscatter strategy used for YxZ Threads

“Nested Stream”

threads	1x1	1x4	4x1	4x4	6x1	6x4	13x1	13x4
Barcelona								
unbound	4.4	4.9	15.0	10.7				
bound	4.4	7.6	15.8	13.1				
Tigerton								
Unbound	2.3	6.0	4.8	8.7				
Bound	2.3	3.0	8.2	8.5				
ScaleMP								
Unbound	3.8	10.7	11.2	1.7	9.0	1.6	3.4	2.4
bound	3.8	5.9	14.4	18.8	20.4	15.8	43.0	27.8

Memory bandwidth in GB/s of the nested Stream benchmark.
Y,scatter,Z,subscatter strategy used for YxZ Threads

“Nested Stream”

threads	1x1	1x4	4x1	4x4	6x1	6x4	13x1	13x4
Barcelona								
unbound	4.4	4.9	15.0	10.7				
bound	4.4	7.6	15.8	13.1				
Tigerton								
Unbound	2.3	6.0	4.8	8.7				
Bound	2.3	3.0	8.2	8.5				
ScaleMP								
Unbound	3.8	10.7	11.2	1.7	9.0	1.6	3.4	2.4
bound	3.8	5.9	14.4	18.8	20.4	15.8	43.0	27.8

Memory bandwidth in GB/s of the nested Stream benchmark.
Y,scatter,Z,subscatter strategy used for YxZ Threads

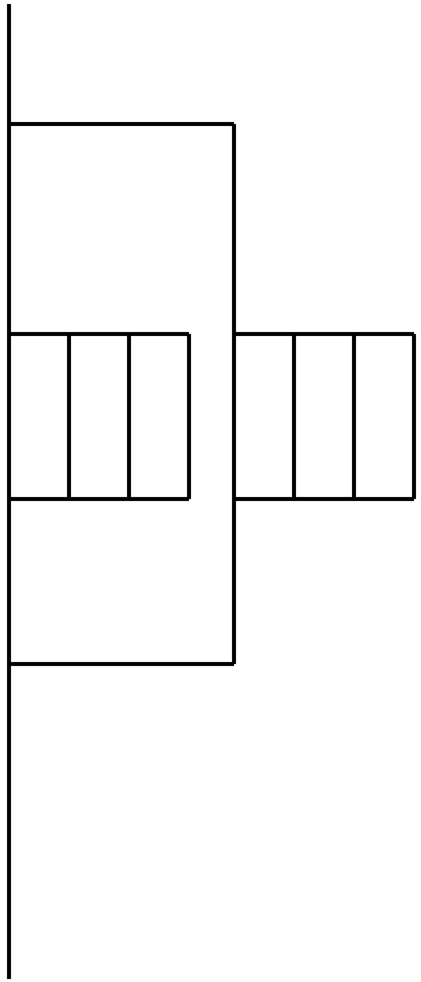
“Nested Stream”

threads	1x1	1x4	4x1	4x4	6x1	6x4	13x1	13x4
Barcelona								
unbound	4.4	4.9	15.0	10.7				
bound	4.4	7.6	15.8	13.1				
Tigerton								
Unbound	2.3	6.0	4.8	8.7				
Bound	2.3	3.0	8.2	8.5				
ScaleMP								
Unbound	3.8	10.7	11.2	1.7	9.0	1.6	3.4	2.4
bound	3.8	5.9	14.4	18.8	20.4	15.8	43.0	27.8

Memory bandwidth in GB/s of the nested Stream benchmark.
Y,scatter,Z,subscatter strategy used for YxZ Threads

"Nested EPCC synchbench"

- modification of EPCC microbenchmarks
- start different teams
- each inner team uses synchronization constructs
- compute average synchronization overhead



“Nested EPCC synchbench”

	parallel	barrier	reduction	parallel	barrier	reduction
Barcelona		**1x2**			**4x4**	
unbound	19.25	18.12	19.80	117.90	100.27	119.35
bound	23.53	20.97	24.14	70.05	69.26	69.29
Tigerton		**1x2**			**4x4**	
unbound	23.88	20.84	24.17	74.15	54.90	77.00
bound	9.48	7.35	9.77	58.96	34.75	58.24
ScaleMP		**1x2**			**2x8**	
unbound	63.53	65.00	42.74	2655.09	2197.42	2642.03
bound	34.11	33.47	41.99	425.63	323.71	444.77

overhead of nested OpenMP constructs in microseconds
Y,scatter,Z,subcompact strategy used for YxZ Threads

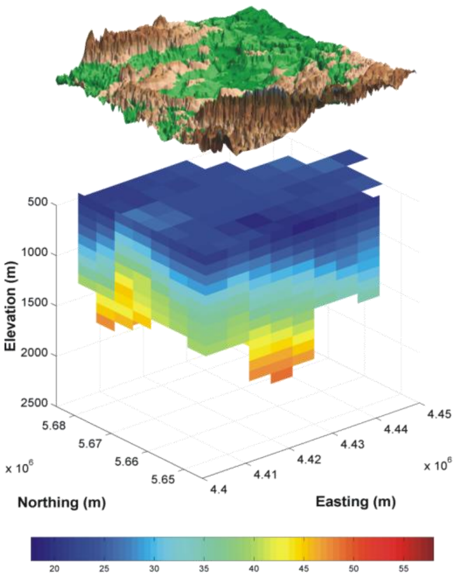
“Nested EPCC synchbench”

	parallel	barrier	reduction	parallel	barrier	reduction
Barcelona		**1x2**			**4x4**	
unbound	19.25	18.12	19.80	117.90	100.27	119.35
bound	23.53	20.97	24.14	70.05	69.26	69.29
Tigerton		**1x2**			**4x4**	
unbound	23.88	20.84	24.17	74.15	54.90	77.00
bound	9.48	7.35	9.77	58.96	34.75	58.24
ScaleMP		**1x2**			**2x8**	
unbound	63.53	65.00	42.74	2655.09	2197.42	2642.03
bound	34.11	33.47	41.99	425.63	323.71	444.77

- Barcelona: 1.7X
- Tigerton: 1.3X
- ScaleMP: 6X

overhead of nested OpenMP constructs in microseconds
Y,scatter,Z,subcompact strategy used for YxZ Threads

Geothermal Simulation Package to simulate groundwater flow, heat transport, and the transport of reactive solutes in porous media at high temperatures (3D)



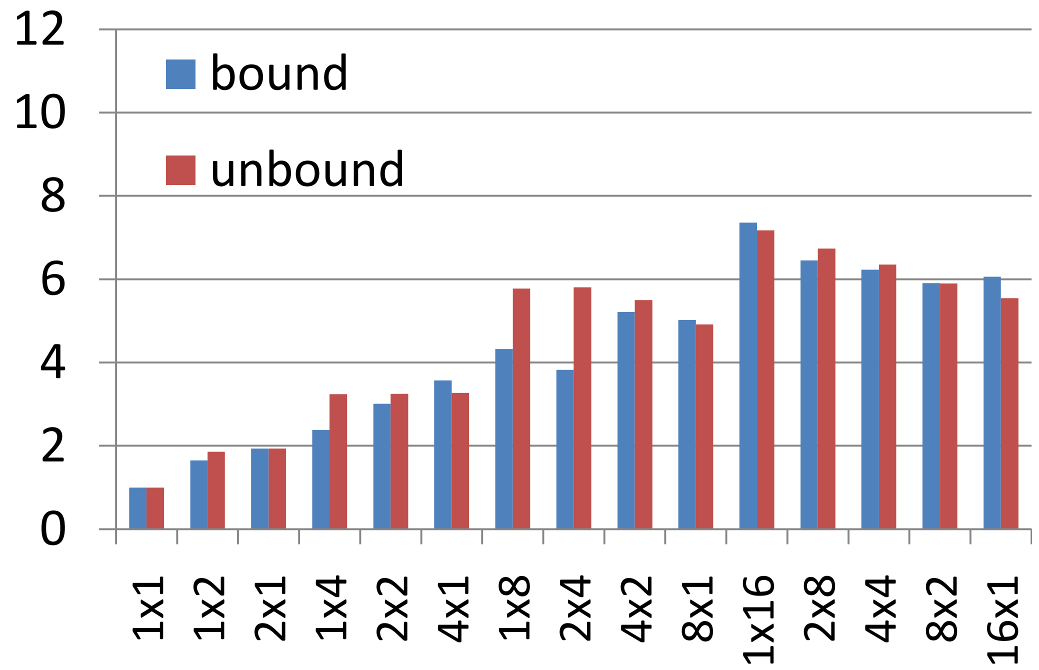
Institute for Applied Geophysics of RWTH Aachen University

Written in Fortran, two levels of parallelism:

- Independent Computations of the Directional Derivatives (Maximum is 16 for the used Dataset)
- Setup and Solving of linear equation systems

SHEMAT-Suite on Tigerton

- small differences
- sometimes unbound strategy is advantageous
- for larger thread numbers only very small differences
- binding in the best case brings only 2,3 %

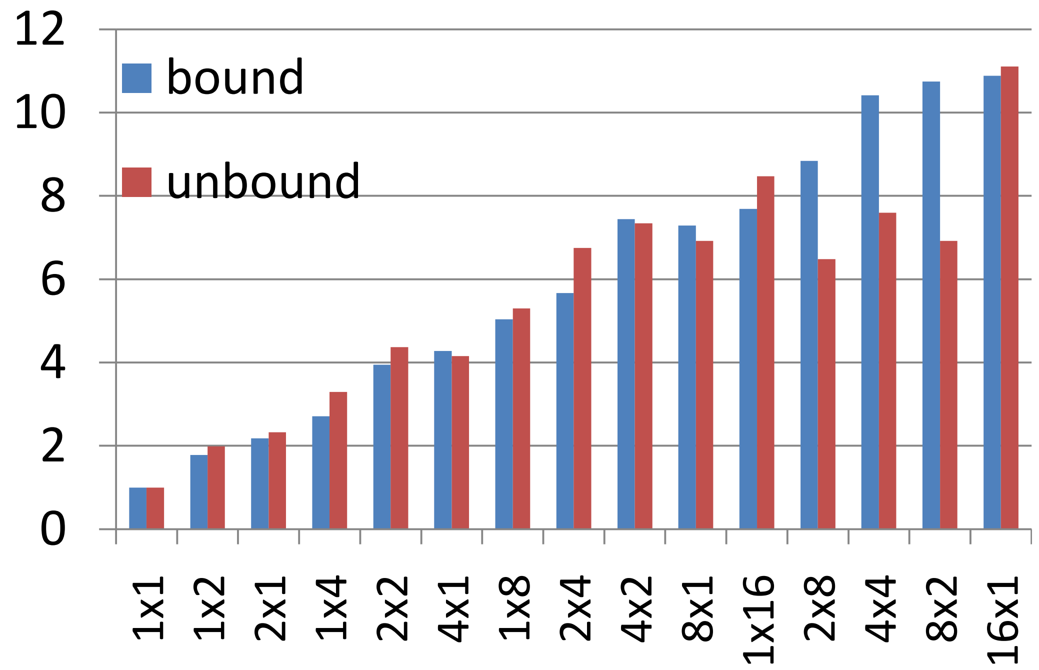


speedup for different thread numbers on Tigerton

Y,scatter,Z,subscatter strategy used for YxZ Threads

SHEMAT-Suite on Barcelona

- larger differences
- for larger thread numbers noticeable differences in nested cases
- e.g. for 8x2 the improvement is 55%

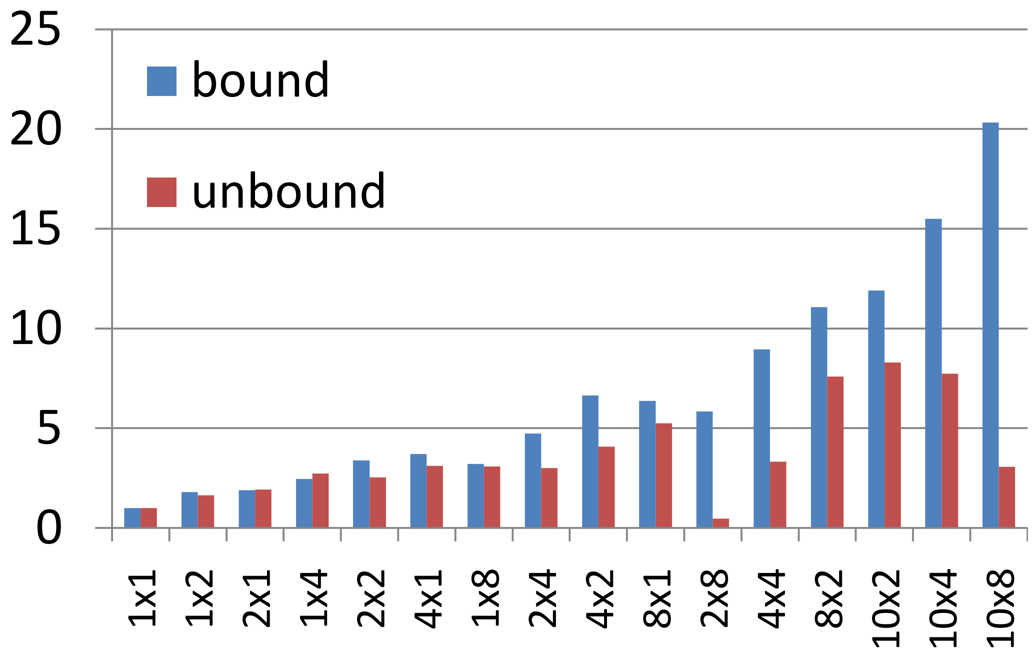


speedup for different thread numbers on Barcelona

Y,scatter,Z,subscatter strategy used for YxZ Threads

SHEMAT-Suite on ScaleMP

- very larger differences when multiple boards are used
- binding in the best case brings 2.5X performance improvement
- speedup is about 2 times higher than on the Barcelona and 3 times higher than on the Tigerton



speedup for different thread numbers on ScaleMP

Y,scatter,Z,subscatter strategy used for YxZ Threads

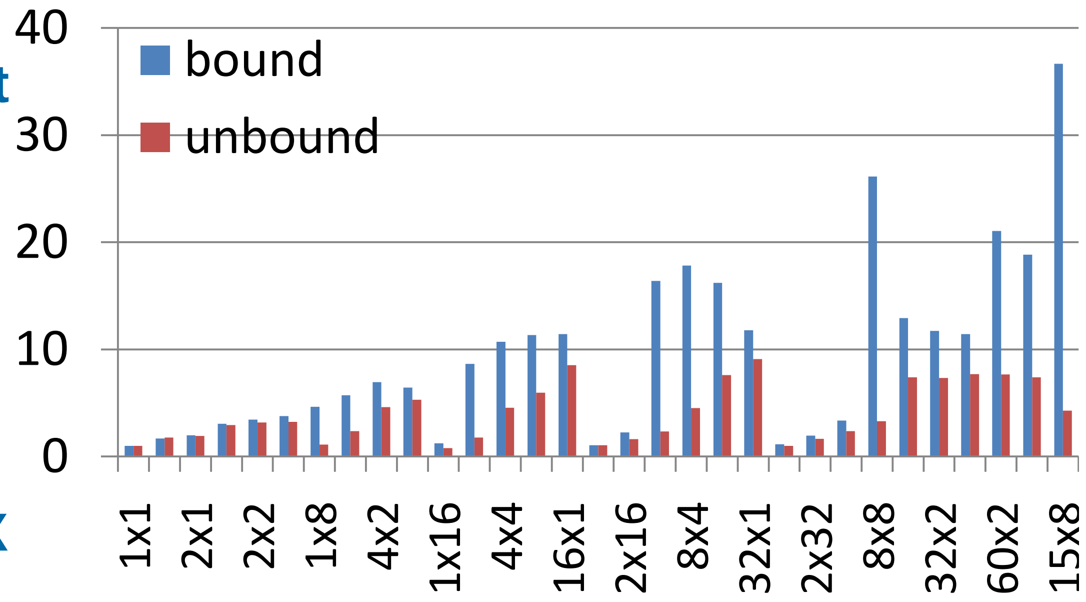
Recent Results: SHEMAT-Suite on new ScaleMP box at Bielefeld

ScaleMP – Bielefeld:

1. 15 (16) board each 2 x Intel Xeon E5550 @ 2,67 GHz (Nehalem)
2. 320 GB RAM - ~32 GB reserved for vSMP = 288 GB available

Shemat – ScaleMP Bielefeld

- speedup of about 8 without binding
- speedup of up to 38 when binding is used
- comparing best effort in both cases: improvement 4X



speedup for different thread numbers on ScaleMP - Bielefeld

Y,scatter,Z,subscatter strategy used for YxZ Threads

Summary

- **Binding is Important especially for Nested OpenMP Programs**

- **On SMP's the advantage is low, but on cc-NUMA architectures it is really an advantage**

- **Hardware Details are confusing and should be hidden from the User**

Thank you for

your attention!

Questions?