

How to scale Nested OpenMP Applications on the ScaleMP vSMP Architecture

Dirk Schmidl, Christian Terboven, Andreas Wolf,
Dieter an Mey, Christian Bischof

IEEE Cluster 2010 / Heraklion
September 21, 2010

- ▶ **ScaleMP Overview**
- ▶ **Kernel Benchmarks**
 - ▶ Page Access Benchmark
 - ▶ STREAM
 - ▶ SMXV
 - ▶ Memory Allocation Tests
 - ▶ EPCC Microbenchmarks
- ▶ **Real World Applications**
 - ▶ Flexible Image Retrieval Engine (FIRE)
 - ▶ Simulator for Heat and Mass Transport (SHEMAT-Suite)
- ▶ **Conclusions**

1. 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

2. Tigerton (Fujitsu-Siemens RX600): (reference system)

1. 4 x Intel Xeon X7350 @ 2,93 GHz
2. 1 x 64 GB RAM

Benchmarks to measure page access time:

1. read a remote page
2. write a remote page
3. write a local page

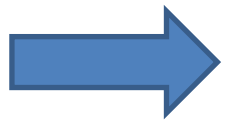
	ScaleMP	Tigerton
read_from_other	43,37	2,97
write_from_other	40,44	2,13
write_self	2,34	1,64

results in microseconds

- remote accesses 15 - 20 X more expensive on ScaleMP
- local accesses not affected by vSMP Software

default:

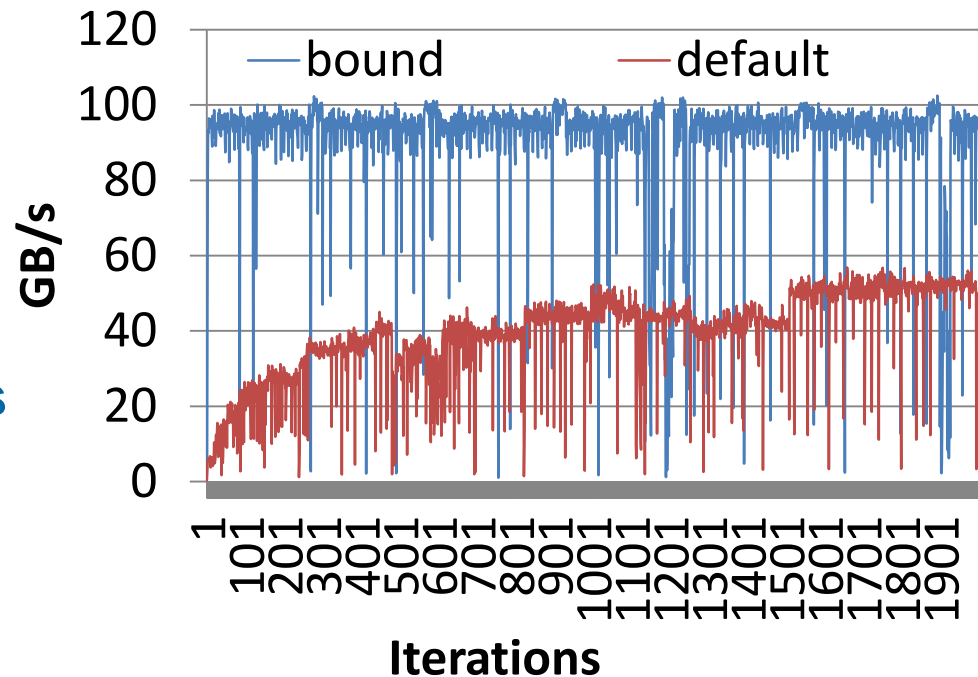
- ▶ memory bandwidth not constant over time
- ▶ threads are slowly migrated to other boards
- ▶ reached maximum ~45 GB/s



apply fix thread placement

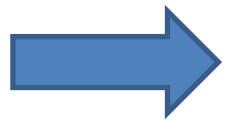
bound:

- ▶ reached bandwidth ~95 GB/s
- ▶ maximum reached from beginning



default:

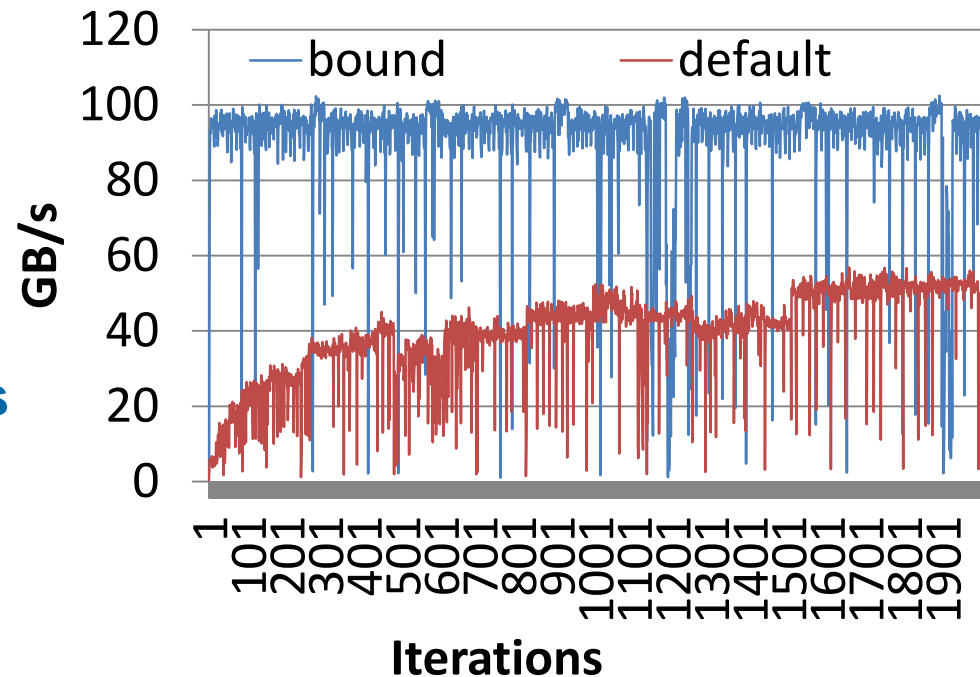
- ▶ memory bandwidth not constant over time
- ▶ threads are slowly migrated to other boards
- ▶ reached maximum ~45 GB/s



apply fix thread placement

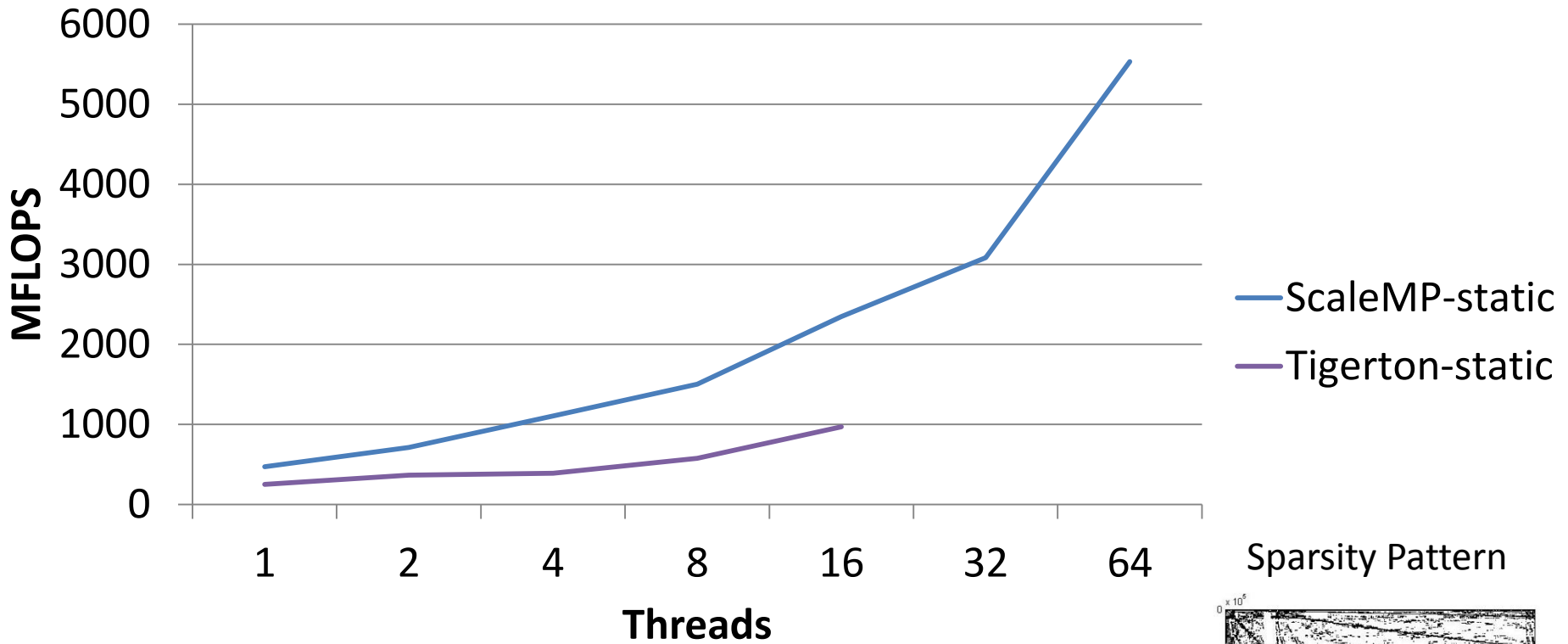
bound:

- ▶ reached bandwidth ~95 GB/s
- ▶ maximum reached from beginning



Conclusion:

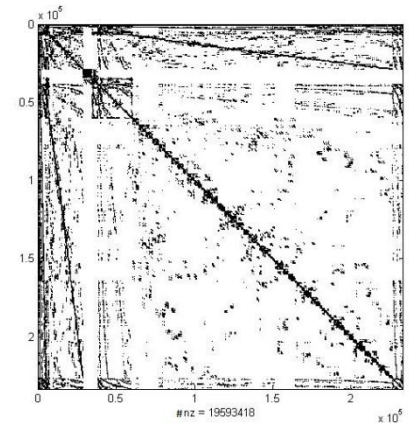
- high bandwidth possible for aligned memory accesses
- thread placement is important on ScaleMP

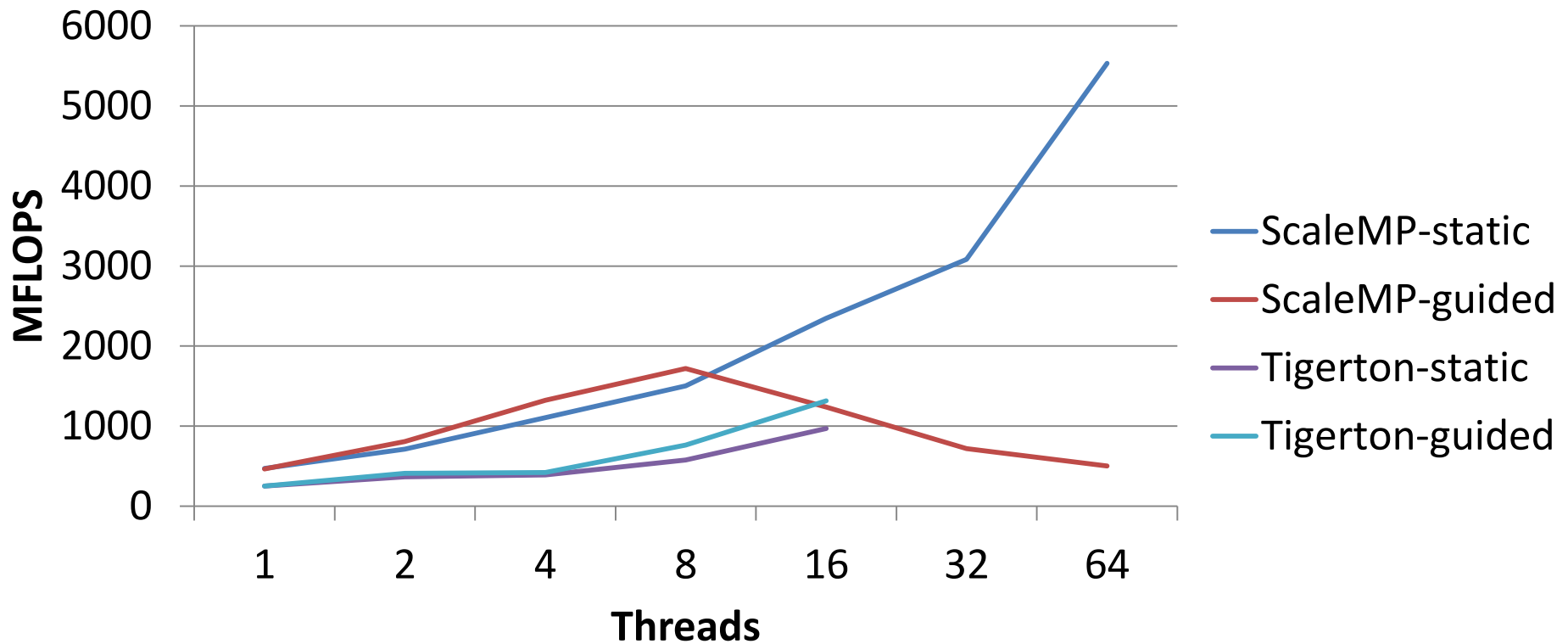


static schedule:

- high performance compared to SMP system
- load imbalance not addressed

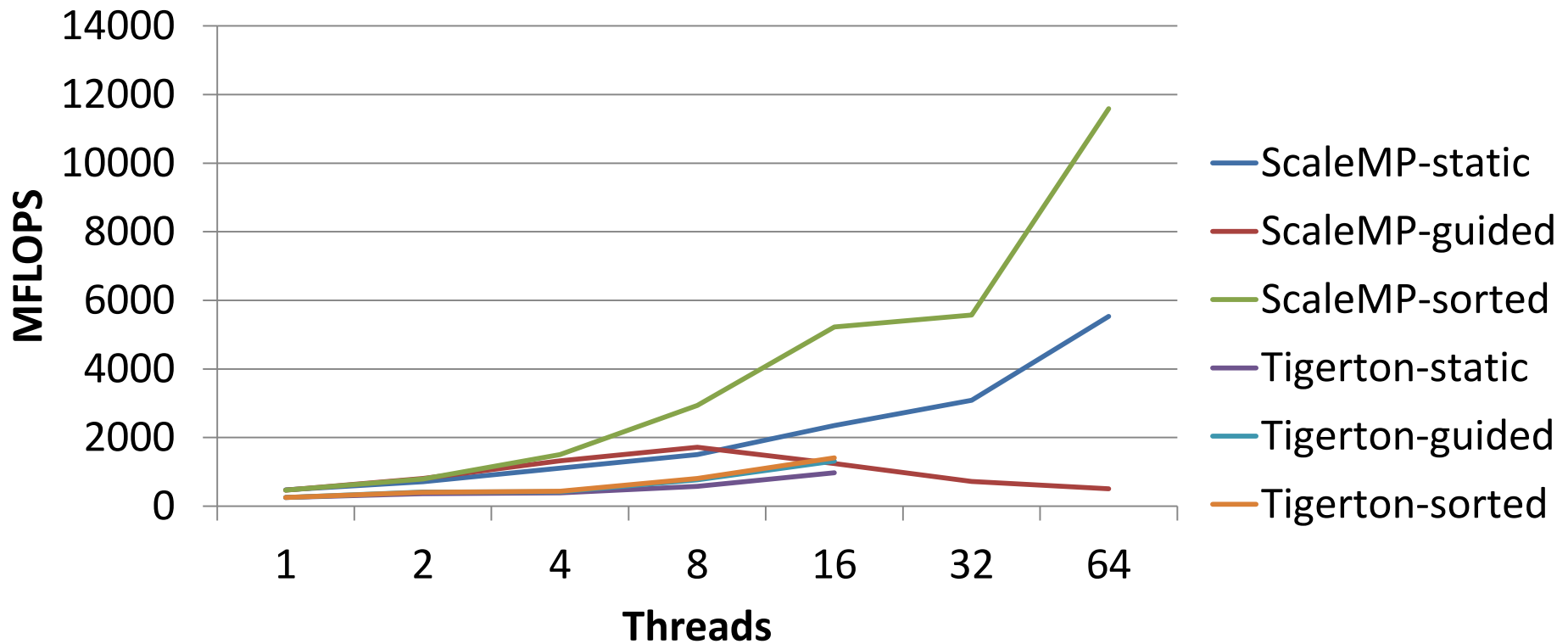
Sparsity Pattern





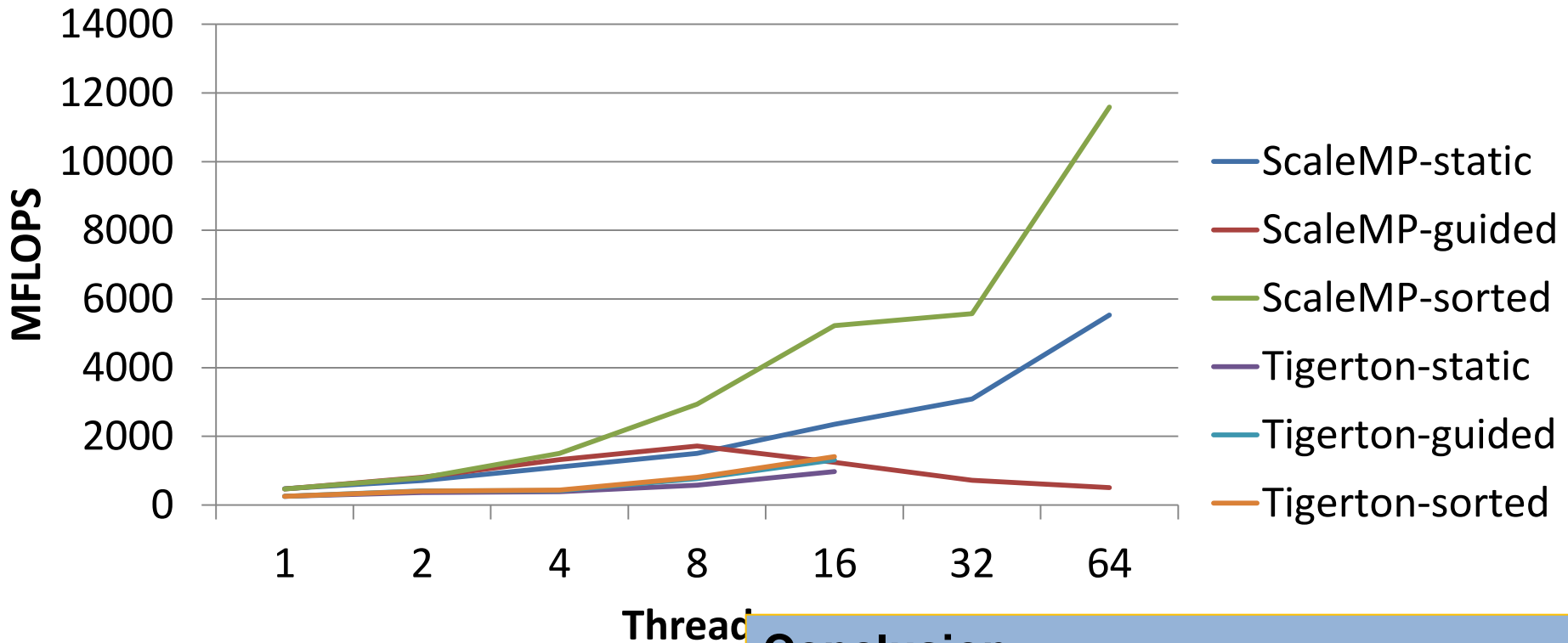
guided schedule:

- performance improvement on the SMP system
- bad performance on ScaleMP
- guided schedule introduces high number of remote accesses



sorted schedule:

- nearly no changes on the SMP
- good performance on ScaleMP +10X



sorted schedule:

- nearly no changes on the SMP
- good performance on ScaleMP

Conclusion:

- high bandwidth also for sparse matrixes possible
- dynamic access patterns are a problem
- precomputed pattern needed

- ▶ **time (in sec.) to allocate and initialize a 15 GB array in parallel**

Default Pages (4 KB):

- ▶ **allocation takes much longer on ScaleMP**
- ▶ **more Threads introduce larger overhead**

Huge Pages (2 MB):

- ▶ **nearly no benefit for one thread on ScaleMP**
- ▶ **scales better, but still not very well**

	default pages	huge pages
Tigerton		
1 Thread	34,79	27,01
8 Threads	16,6	20,34
16 Threads	26,9	15,87
ScaleMP		
1 Thread	211,13	196,29
8 Threads	423,4	187,45
16 Threads	449,01	179,25
104 Threads	432,47	121,76

- ▶ **time (in sec.) to allocate and initialize a 15 GB array in parallel**

Default Pages (4 KB):

- ▶ **allocation takes much longer on ScaleMP**
- ▶ **more Threads introduce larger overhead**

Huge Pages (2 MB):

- ▶ **nearly no benefit for one thread on ScaleMP**
- ▶ **scales better, but still not very well**

	default pages	huge pages
Tigerton		
1 Thread	34,79	27,01
8 Threads	16,6	20,34
16 Threads	26,9	15,87
ScaleMP		
1 Thread	211,13	196,29
8 Threads	423,4	187,45
16 Threads	449,01	179,25
104 Threads	432,47	121,76

Conclusion:

- **memory allocation takes longer on ScaleMP**
- **codes with a lot of allocation and deallocation of memory will most likely run bad on these systems**

ScaleMP	parallel	for	reduction
2 Threads on 1 boards	1,89	0,62	2,07
2 Threads on 2 boards	297,06	279,91	362,76
16 Threads on 2 boards	768,28	571,10	923,30
104 Threads on 13 boards	1453,30	1198,02	3185,69
Tigerton			
2 Threads	2,14	0,89	2,32
8 Threads	4,10	1,58	4,40
16 Threads	6,76	2,25	7,41

overhead in microseconds

- ▶ **~100 – 200 X more synchronization time when multiple boards are used**

ScaleMP	parallel	for	reduction
2 Threads on 1 boards	1,89	0,62	2,07
2 Threads on 2 boards	297,06	279,91	362,76
16 Threads on 2 boards	768,28	571,10	923,30
104 Threads on 13 boards	1453,30	1198,02	3185,69
Tigerton			
2 Threads	2,14	0,89	2,32
8 Threads	4,10	1,58	4,40
16 Threads	6,76	2,25	7,41

overhead in microseconds

- ▶ **~100 – 200 X more synchronization time when multiple boards are used**

Conclusion:

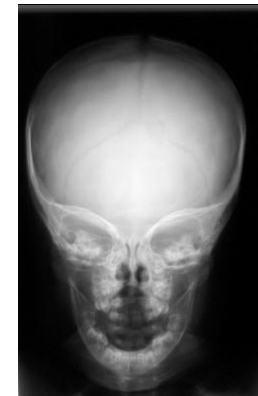
- use as few synchronizations as possible
- if synchronizations are your performance bottleneck, then your code will most likely not run well on ScaleMP



FIRE = Flexible Image Retrieval Engine

- **Compare the performance of common features on different databases**
- **Analysis of correlation of different features**

*Thomas Deselaers and Daniel Keysers,
RWTH I6: Chair for Human Language
Technology and Pattern Recognition*



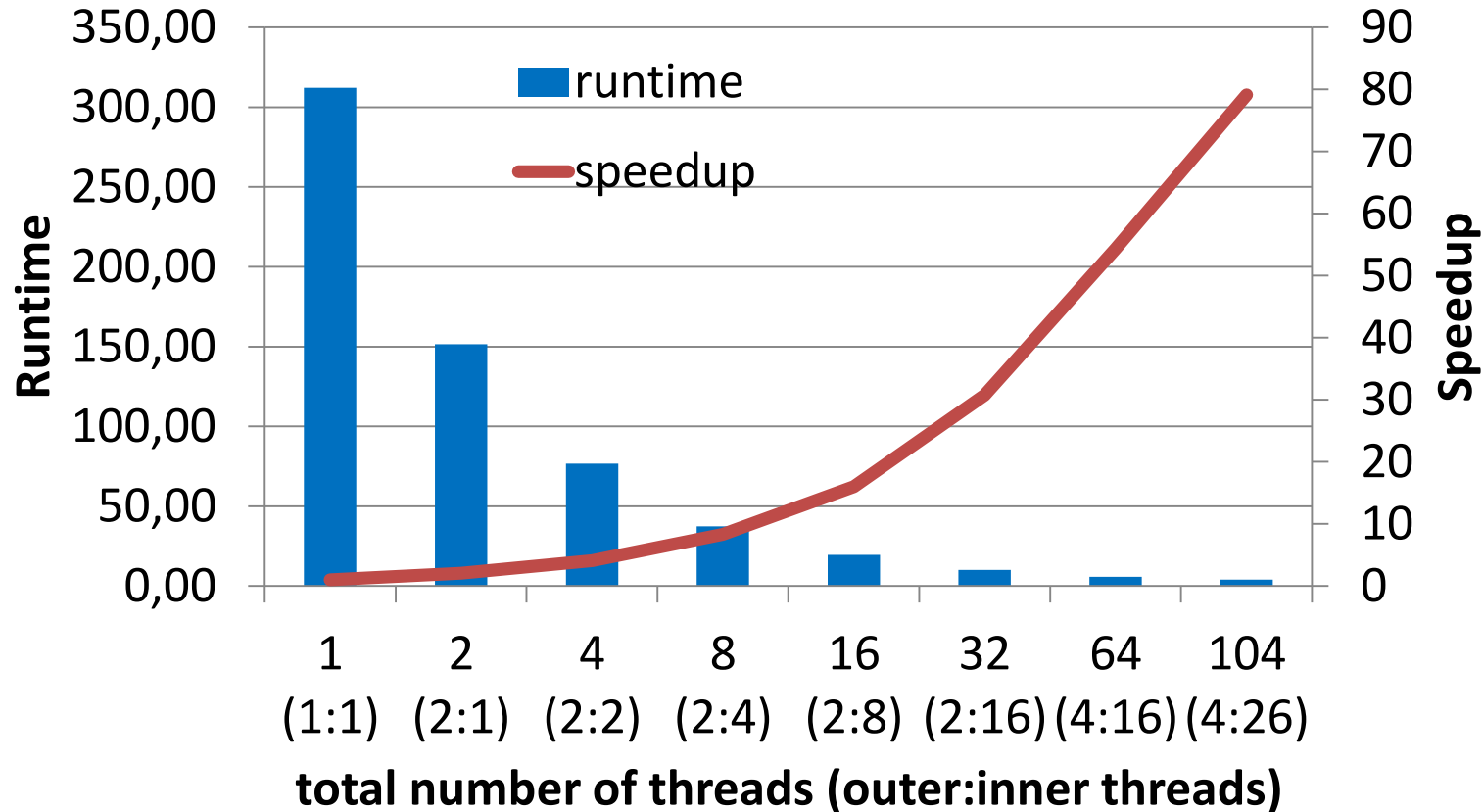
$$D(Q, X) := \sum_{m=1}^M w_m \cdot d_m(Q_m, X_m)$$

- **Q: query image, X: set of database images**
- **Q_m, X_m: m-th feature of Q and X**
- **d_m: distance measure, w_m: weighting coefficient**
- **Return the k images with lowest distance to query image**

- **Well-suited for Shared-Memory parallelization:
Data Mining in a large image database!**

- **Two levels of parallelism:**
 - **Process multiple query images in parallel**
 - **Process database comparison for one query image in parallel**

- $|Q| = 18$
- $|X| = 1000$



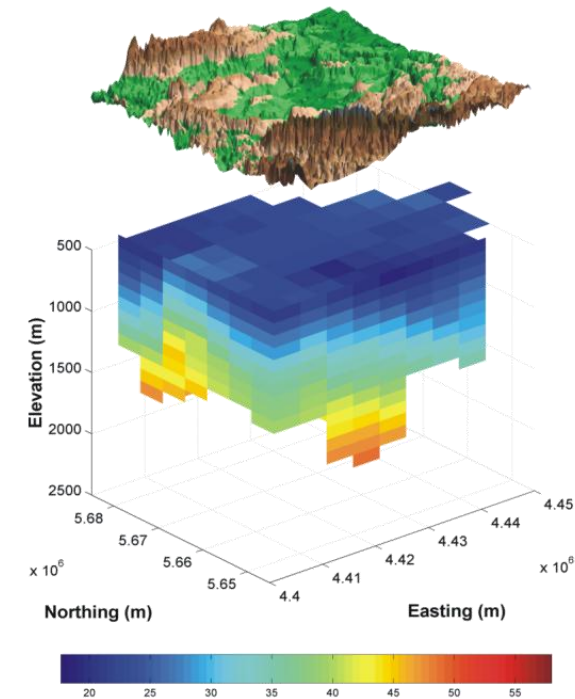
- speedup of ~80 on 104 Cores
- small load imbalance due to small number of query Images

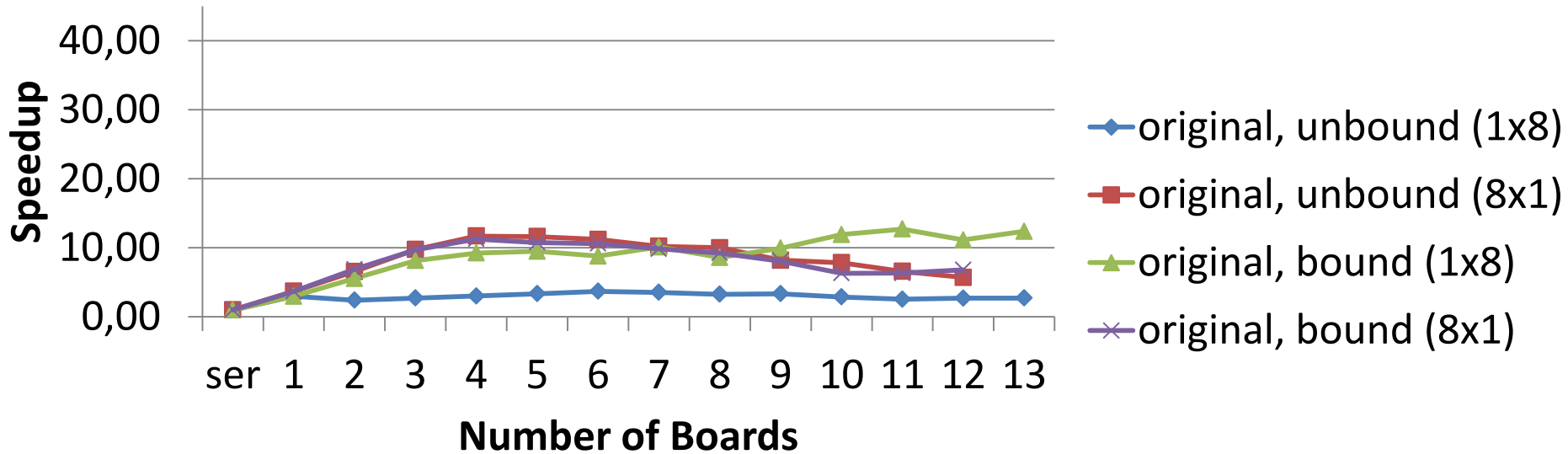
Geothermal Simulation Package to simulate groundwater flow, heat transport, and the transport of reactive solutes in porous media at high temperatures (3D)

Applied Geophysics and Geothermal Energy, E.ON Energy Research Center

Written in Fortran, two levels of parallelism:

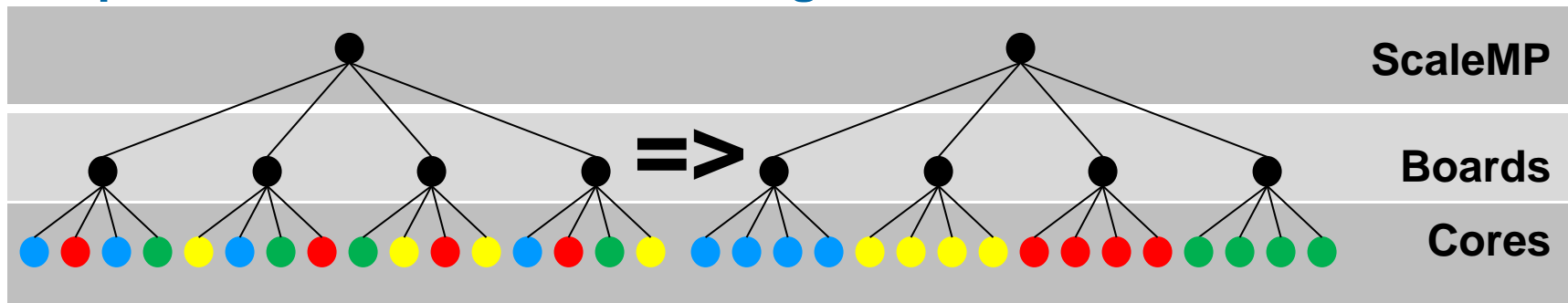
- ▶ **Independent Computations of the Directional Derivatives using AD**
- ▶ **Setup and Solving of linear equation systems**



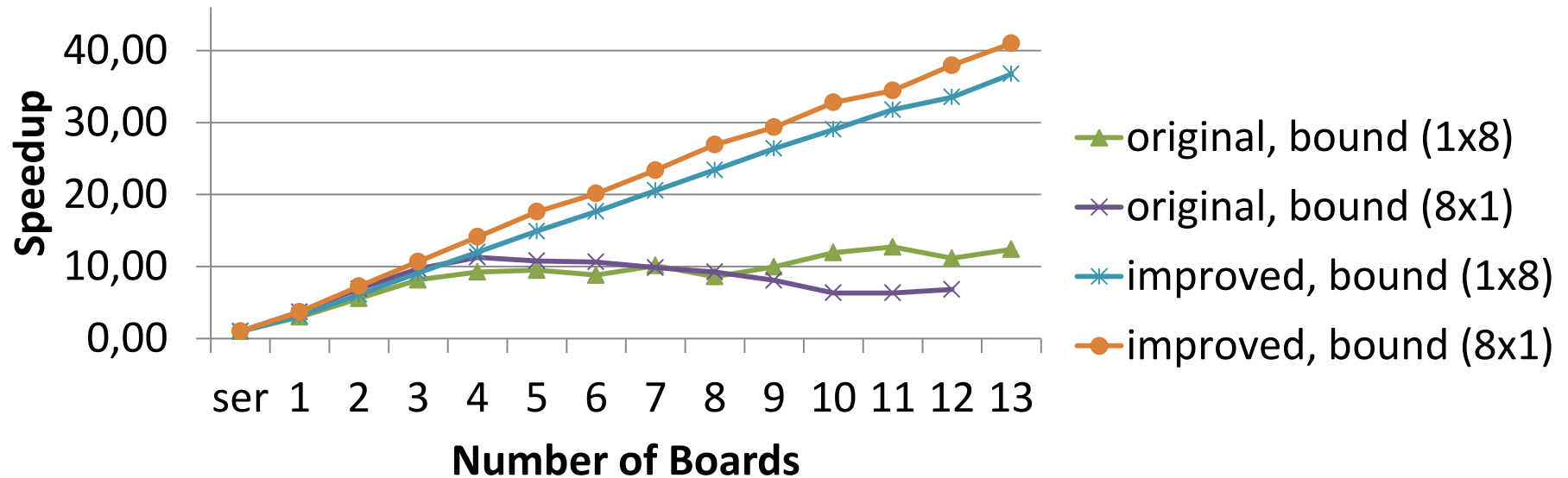


- Binding nested OpenMP programs was a problem, since there is no way to specify the mapping of inner teams to cores

=> implement manual thread binding



Simulator for Heat and Mass Transport (SHEMAT-Suite)



- **creation of private arrays turned out to be the performance bottleneck**

=> changed data management to use slices of a larger global array

Estimated speedup

number of directional derivatives, 104 Cores used:



First Phase: 

8 Threads working per board

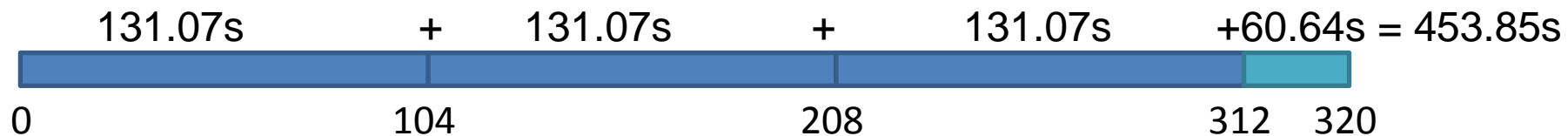
Best result for 8x1 on one board is 5242.67s \cong 131.07 s/8 derivatives

Second Phase: 

1 Threads working per board

Best result for 1x1 on one board is 19404.3s \cong 60.64 s/derivative

Estimated speedup:



First Phase:

8 Threads working per board

Best result for 8x1 on one board is 5242.67s \triangleq 131.07 s/derivative

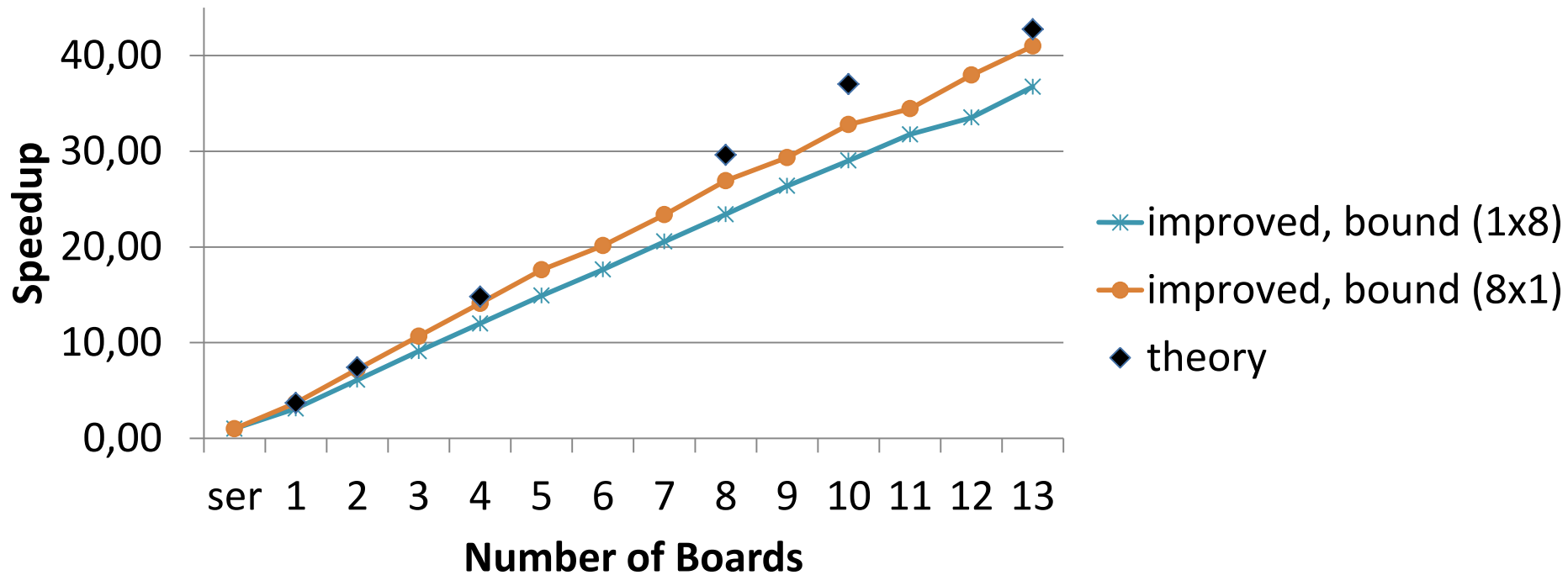
Second Phase:

1 Threads working per board

Best result for 1x1 on one board is 19404.3s \triangleq 60.64 s/derivative

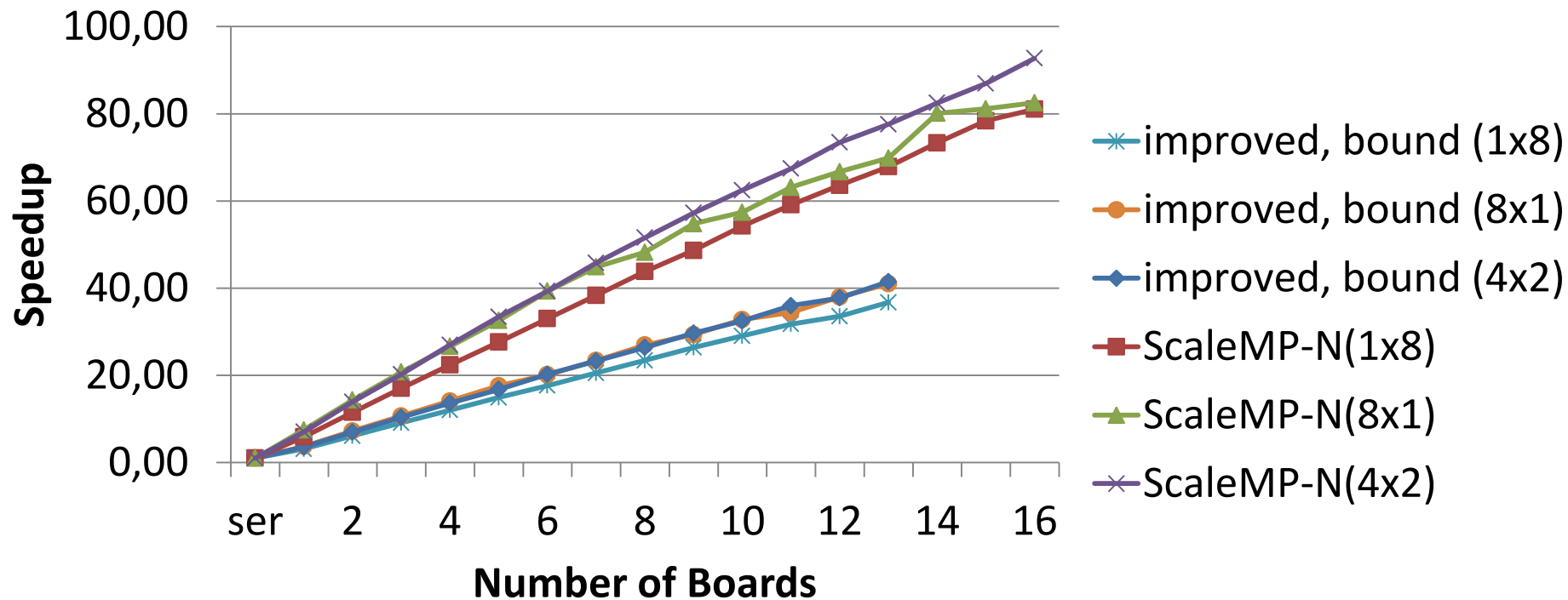
Speedup \triangleq 42.76

Simulator for Heat and Mass Transport (SHEMAT-Suite)



Best Speedup: 41.48

Estimated theoretical Speedup: 42.76



ScaleMP-N:

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

- ▶ high remote memory access (20X) and synchronization (150X) time
- ▶ high memory bandwidth for dense (~95 GB/s) and sparse matrices (10X)
- ⇒ distinct cc-NUMA behavior

- ▶ FIRE achieved speedup of ~80
- ▶ SHEMAT-Suite achieved speedup of 41.5 (theoretical maximum 42.7)
- ⇒ Real user applications can profit from the ScaleMP architecture.

Provided that your application ...

... takes care about thread placement...

... takes care about data placement...

... avoids too many synchronizations...

... reduces the number of memory allocations ...

... than the ScaleMP architecture can deliver a very good performance and a large amount of shared memory at a low price point.

**Thank you for
your attention!**

Questions?