Scalable OpenMP Programming

Dieter an Mey

Center for Computing and Communication
RWTH Aachen University, Germany

www.rz.rwth-aachen.de
anmey@rz.rwth-aachen.de

Overview

- Why OpenMP
- Short OpenMP Introduction
- OpenMP on NUMA Machines
- OpenMP on Clusters
- Conclusion

Overview

- Why OpenMP
- Short OpenMP Introduction
- OpenMP on NUMA Machines
- OpenMP on Clusters
- Conclusion

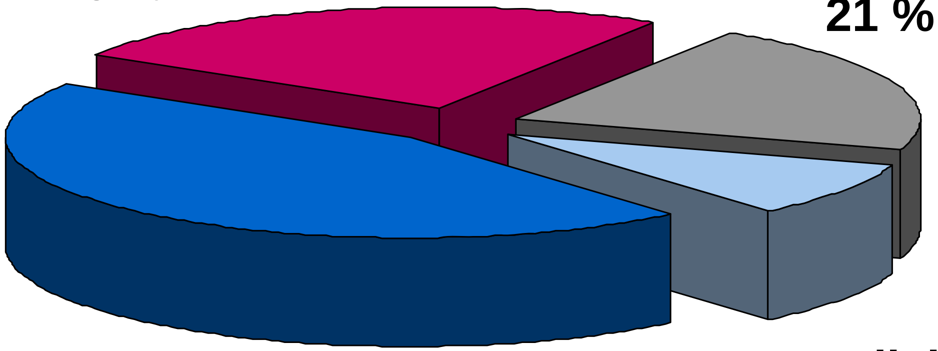
RWTH Aachen University: Key Figures WT 06/07

30180 students

428 professorships

3600 academic staff

260 institutes



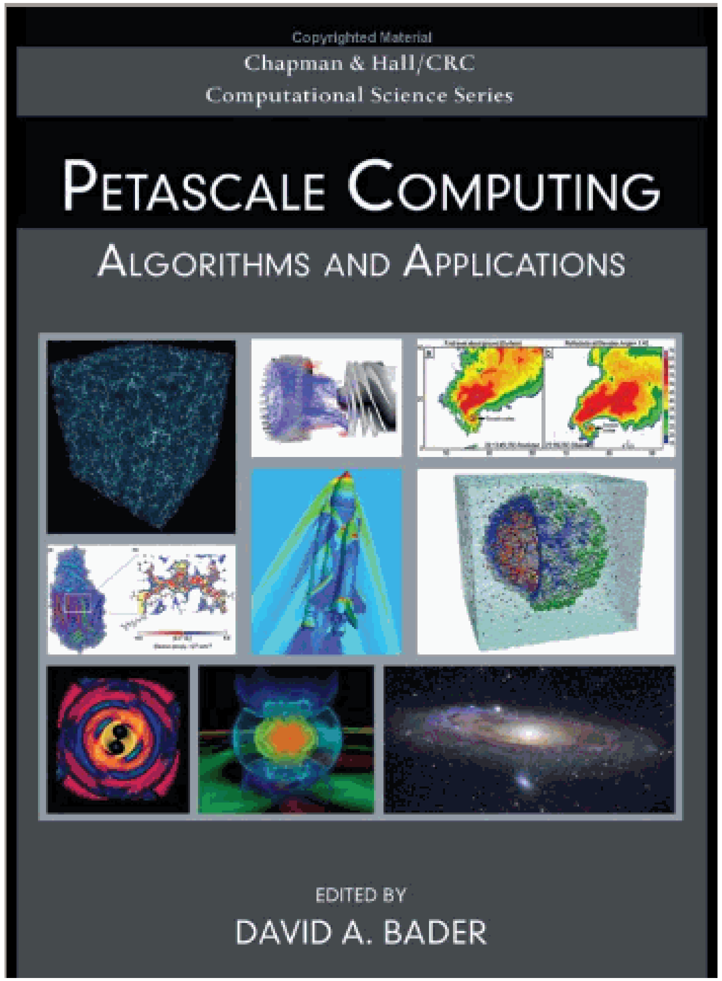
Why OpenMP?

- Large codes mainly in C++ and Fortran and some C
- Software lifetime measured in decades
- MPI is there to stay on clusters
  - Cannot always be applied easily – if at all
  - Scalability may be limited due to underlying problem (geometry etc.)
  - "MPI only" may not be appropriate for "many cores"
    => MPI + OpenMP (hybrid)
- OpenMP is the alternative and the supplement to MPI

- Scalability of OpenMP limited by current machinery
- So far scalability explored on
  - Sun Fire E25K (144 cores UltraSPARC IV)
  - Sun UltraSPARC T2 (64 threads in one "Niagara 2" chip)
  - Intel Cluster OpenMP
  - ScaleMP "Virtual SMP"

RWTH Center for Computing and Communication

Statistics from the "First Petascale Book"



Keyword	Hits	Remarks
MPI	612	since 1994
OpenMP	150	since 1997 with some 28 hits in our own chapter about OpenMP
threads	109	frequently in the context of OpenMP, 57 in our chapter about OpenMP
C++	87	since 1983
Fortran	69	since 1957
Chapel	49	with some 22 hits in Zima's chapter about Chapel
UPC	30	since 2001
Co-array Fortran	27	since 1998
hybrid MPI/OpenMP	~26	hard to count
C	~20	hard to count
HPF	11	since 1993
X10	9	
Fortress	6	
Java	5	since 1995
Titanium	3	
posix threads	2	1995, Linux since 2003

Petascale Computing: Algorithms and Applications (Chapman & Hall/Crc Comp. Sci. Ser.)
edited by David A. Bader , 2007, 528 pages, 24 contributions, 90 contributors

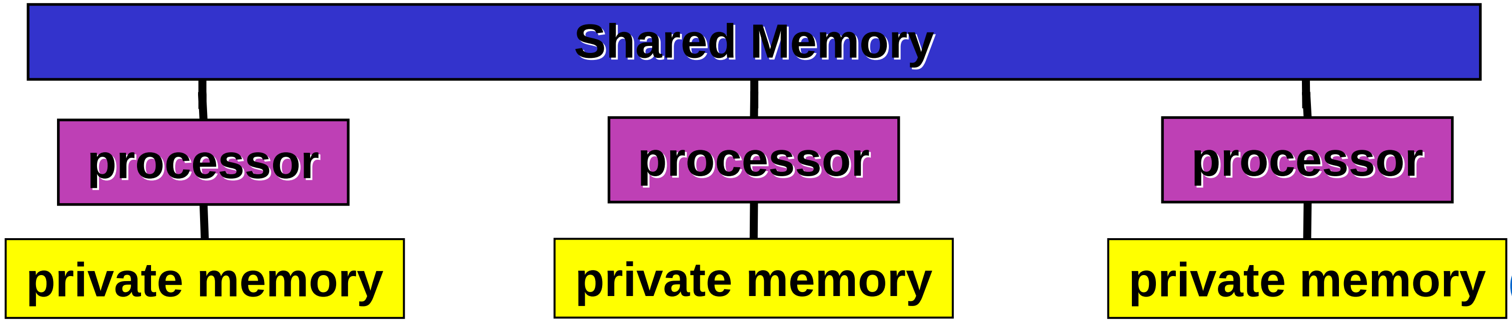
Overview

- Why OpenMP
- Short OpenMP Introduction
- OpenMP on NUMA Machines
- OpenMP on Clusters
- Conclusion

Memory Model of OpenMP

- OpenMP: Shared-Memory model
  - All threads share a common address space (shared memory)
  - Threads can have private data (explicit user control)

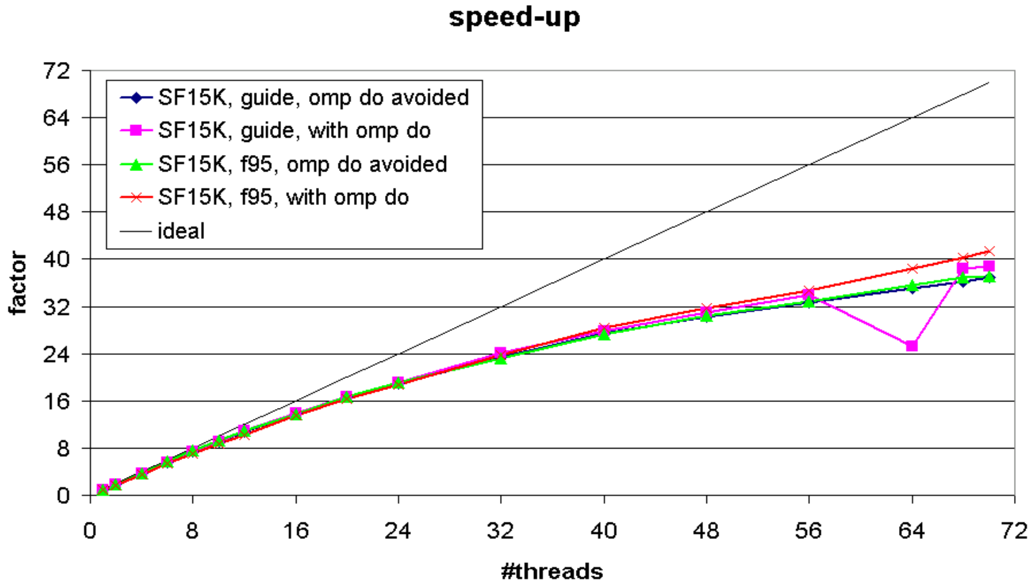
- Relaxed memory consistency
  - Temporary View ("*Caching*"):
    Memory consistency is guaranteed only after synchronization points, namely implicit and explicit `flush`es
    - Each OpenMP `barrier` includes a `flush`
    - Exit from worksharing constructs include barriers by default
    - Entry to and exit from `critical` regions include a `flush`
    - Entry to and exit from lock routines (OpenMP API) include a `flush`



Heat Flow Simulation with FEM - ThermoFlow60

Thomas Haarmann, Wolfgang Koschel, Jet Propulsion Laboratory, RWTH Aachen University

- simulation of the heat flow in a rocket combustion chamber
- Finite Element Method
- OpenMP Parallelelization
  - 30000 lines of Fortran
  - 200 OpenMP directives, 69 parallel loops,
  - 1 main parallel region, "*orphaning*"

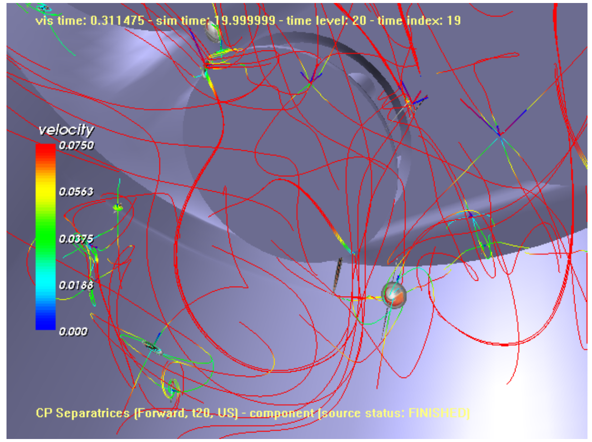


Speedup: ~40 with 68 threads

Nested OpenMP for Critical Point Computation

Samuel Sarholz, Andreas Gerndt, Computing and Communication Center, RWTH Aachen University

- **Analysis of complex and accurate fluid dynamics simulations**
- **Extraction of Critical Points for VR (Location with velocity = 0)**
- **25-100% efficiency with 128 threads on Sun Fire E25K (72 UltraSPARC IV dual core) depending on data set**




```
//  Loop over time levels
#pragma omp parallel for num_threads(nTimeThreads) schedule(dynamic,1)
for (curT=1; curT<=maxT; ++curT) {
//  Loop over Blocks
#pragma omp parallel for num_threads(nBlockThreads) schedule(dynamic,1)
for (curB=1; curB<=maxB; ++curB) {
//  Loop over Cells
#pragma omp parallel for num_threads(nCellThreads) schedule(guided)
for (curC=1; curC<=maxC; ++curC) {
FindCriticalPoints (curT, curB, curC); //  highly adaptive algorithm (bisectioning)
} } }                                  //  huge load imbalances
```

Overview

- Why OpenMP
- Short OpenMP Introduction
- OpenMP on NUMA Machines
- OpenMP on Clusters
- Conclusion

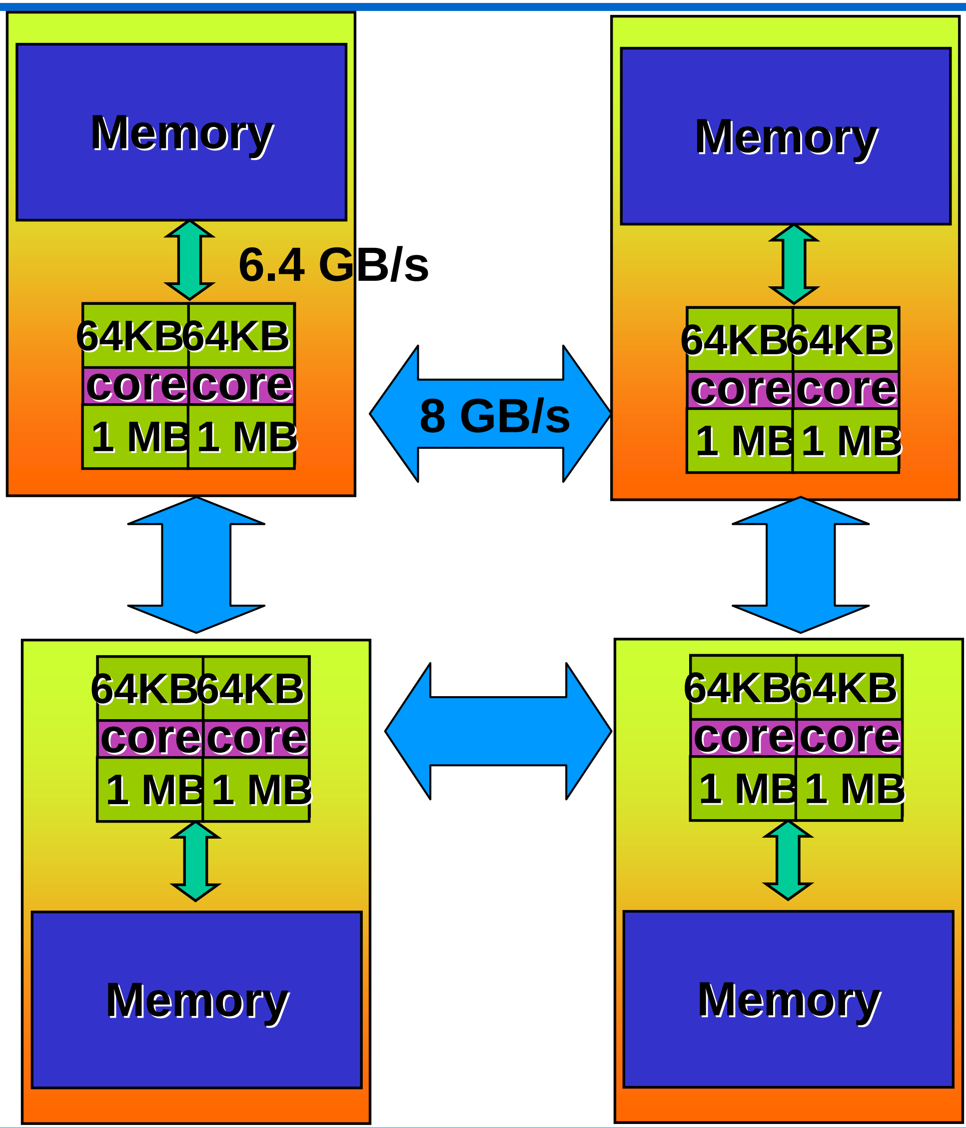
The Earth is Flat

OpenMP is Hardware agnostic
It has no notion of data locality

=>
The Affinity Problem**:**
How to maintaining or improve the nearness of threads and their most frequently used data

Or**:**
Where to run threads?
Where to place data?

Sun Fire V40z (w/ dualcore AMD Opteron Chip)

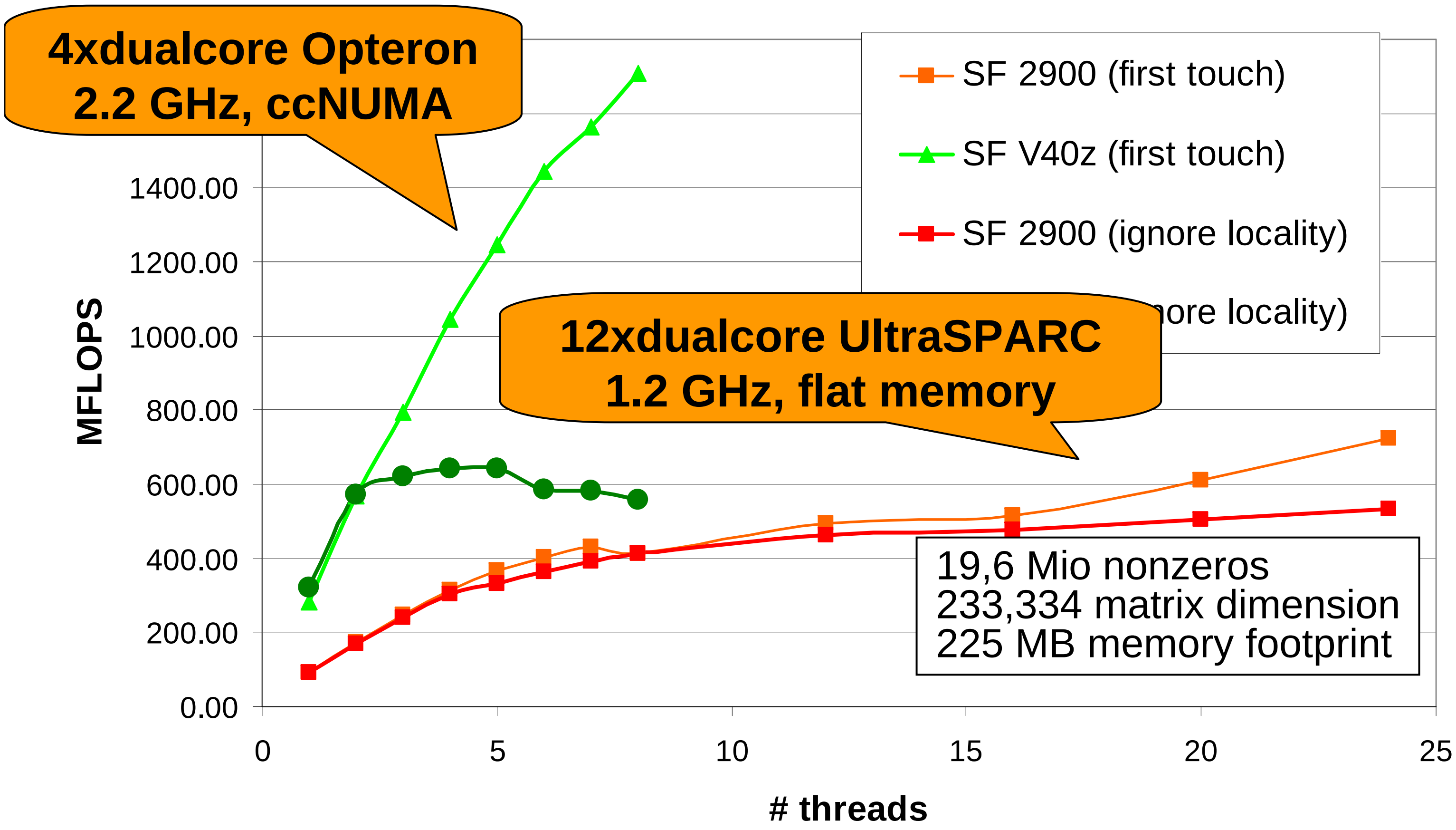


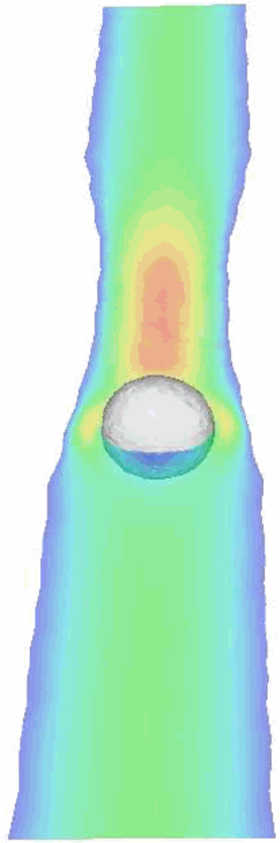
4 AMD Opteron 875 dual core processors 2.2 GHz

Cache-coherent HyperTransport Connections

Sparse-Matrix-Vector-Multiplication as part of the Navier Stokes Solver DROPS (C++)

C. Terboven (RZ,RWTH), A. Spiegel (RZ,RWTH), D. an Mey (RZ,RWTH), S. Gross(IGPM,RWTH), and V. Reichelt (IGPM,RWTH).





IWOMP 2005

Performance of a cc-Numa system is very sensitive to data placement.

RWTH Center for Computing and Communication

Thread-Data-Affinity (1 of 2)

- In an ideal world the operating system together with the OpenMP runtime system would handle affinity automatically.
- In simple situations things might work well:
  - Exclusive access to the compute node
  - Single level of parallelism
  - Static program behaviour concerning thread and data usage
  - Initialization of data by the same thread which later uses the data („first touch policy“)

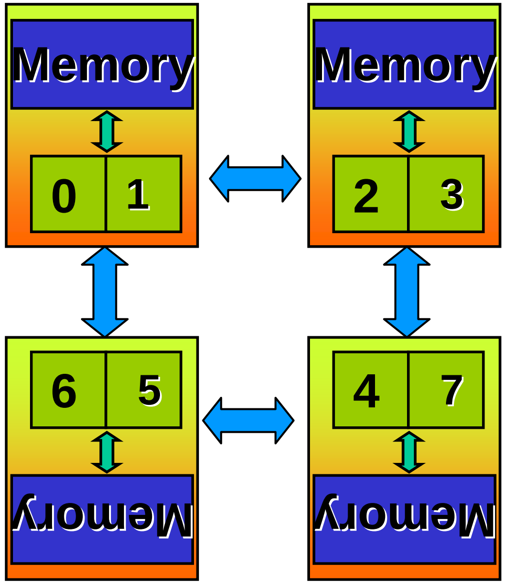
Thread-Data-Affinity (2 of 2)

- In more complicated situations, you may want to
  - Bind threads explicitely
    (How about multi user mode? Hybrid parallelization?)
  - Carefully initialize data
  - If necessarry and possible, migrate data (or threads)
    - Solaris MPO madvise() implements „next touch strategy“
    - Linux 2.6.18: move_pages()
      can be used to implement „next touch strategy“
      (RWTH: prototype by RZ, better solution by LfBS)
    - Windows: Migration is not yet supported)

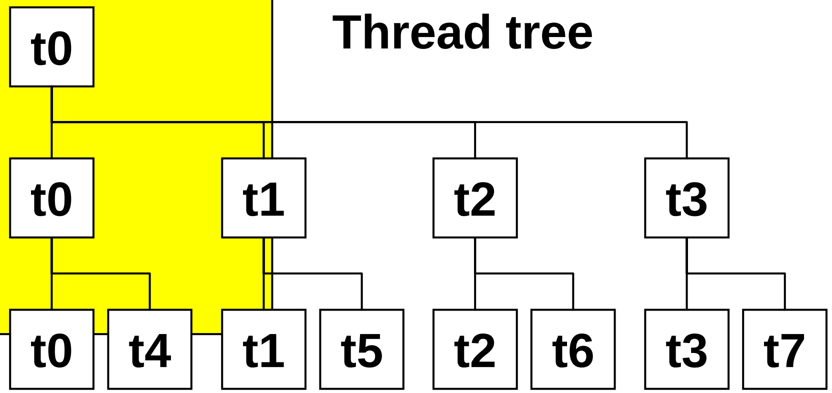
- Nested OpenMP is implemented with thread pool
  - Inner teams‘ threads loose affinity to their data
  - Sun Studio on Solaris: SUNW_MP_THR_AFFINITY=TRUE

OpenMP nested, here: 4x2 threads

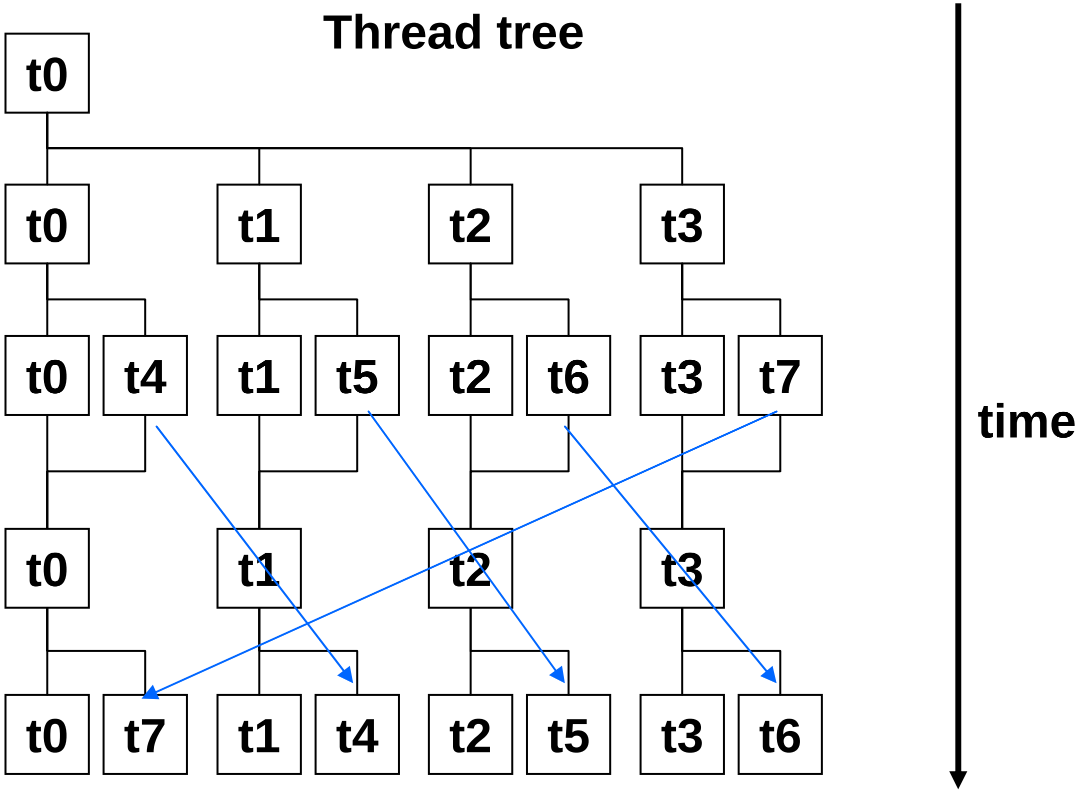
```
!$omp parallel private(me) num_threads(4)
      me = omp_get_thread_num()
      CALL stream(a(1,me),b(1,me),c(1,me))
!$omp end parallel
      ...
      subroutine stream (a,b,c)
      double precision a(*),b(*),c(*)
      ...
!$omp parallel do num_threads(2)
         do 50 j = 1,n
            c(j) = a(j)+x*b(j)
   50    continue
!$omp end parallel do
      ...
```

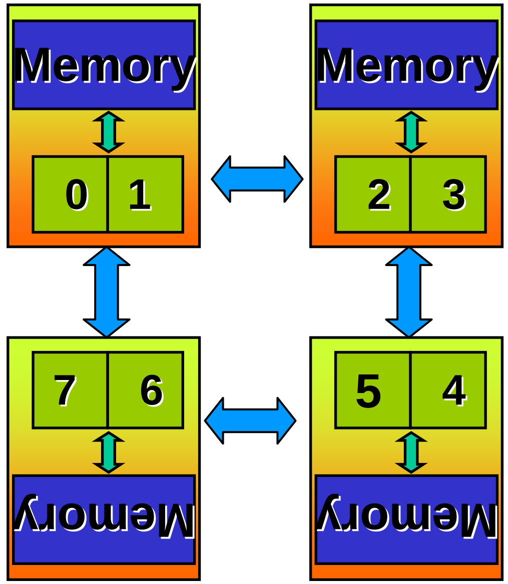


Thread tree



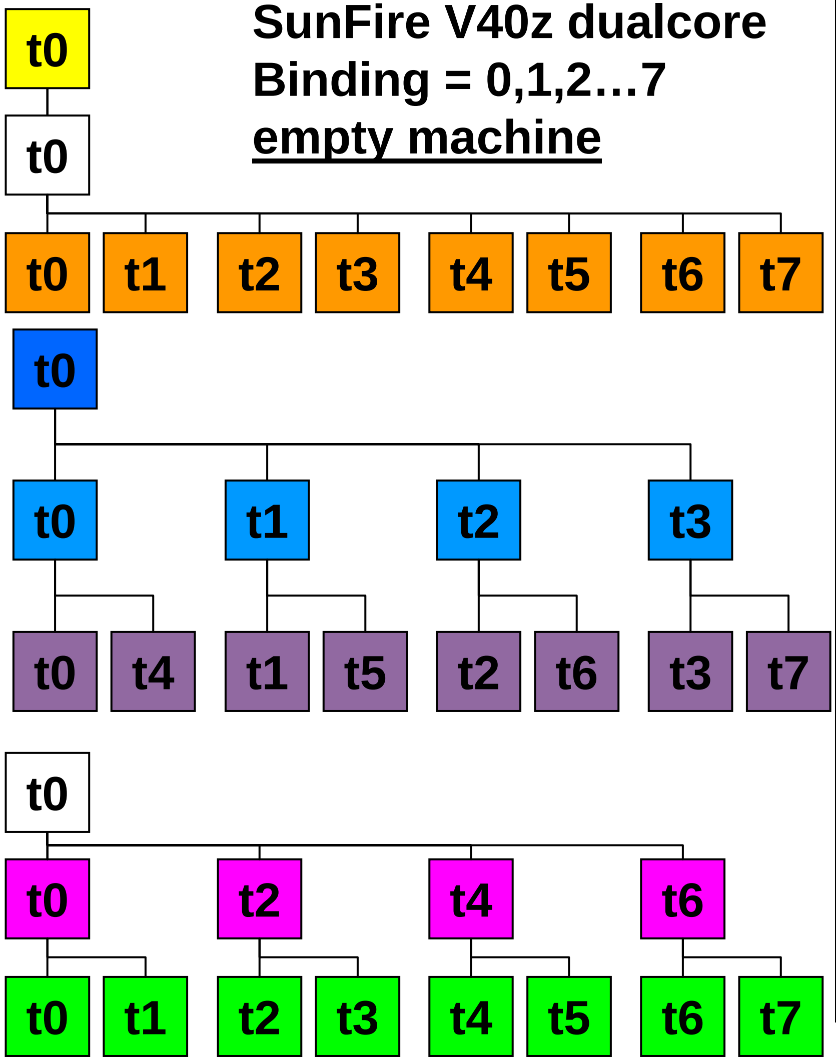
OpenMP nested





Typically OS threads are organized in a pool and may be allocated variably, thus loosing data affinity !

OpenMP nested

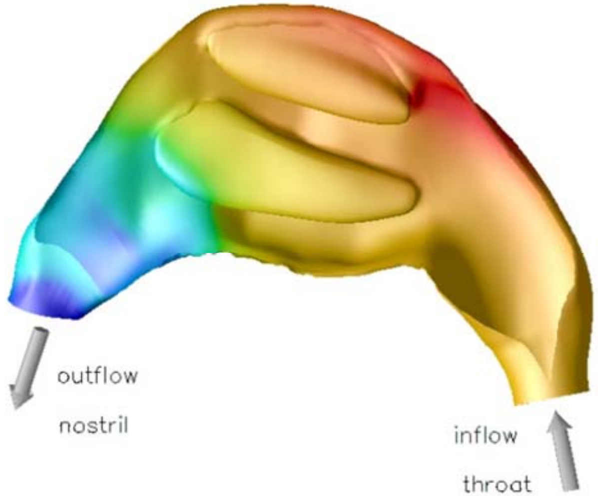


SunFire V40z dualcore
Binding = 0,1,2…7
empty machine

#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

Simulating the Flow through the Human Nose TFS on Solaris

S. Johnson (PSP), C. Ierotheou (PSP),
A. Spiegel (RZ,RWTH), D. an Mey (RZ,RWTH),
I. Hörschler (AIA, RWTH)



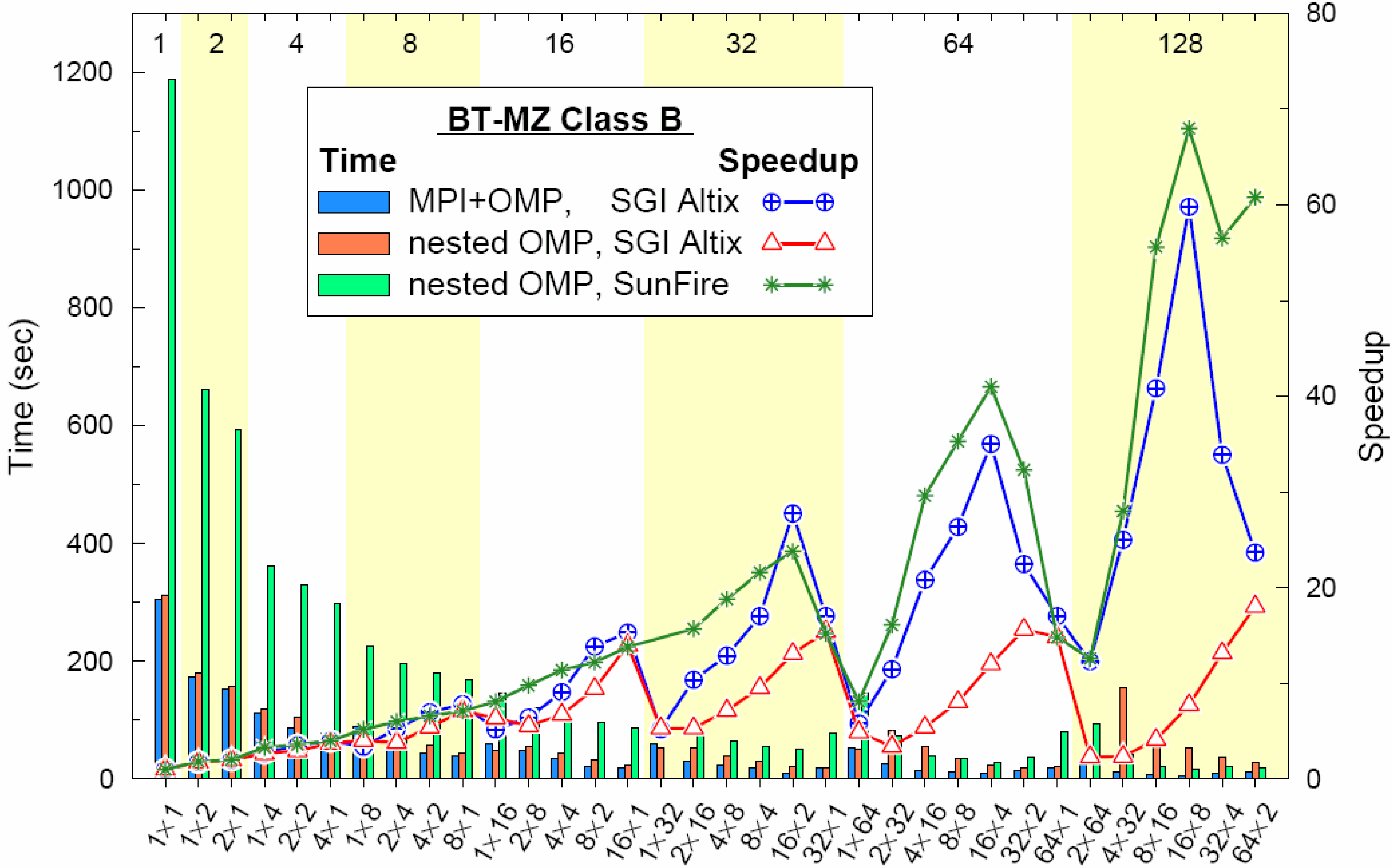
SUNW_MP_THR_AFFINITY=TRUE

Thread affinity + processor binding + data migration improved the performance by ~25 % on a Sun Fire E 25K

Before		Improved thread affinity		
#threads	Speed-up	#threads	Speed-up	Strategy (best effort)
64	**20**	64	**25**	thread balancing 2-11 threads per team, static schedule, 16 threads in outer team
121	**20**	128	**27**	block grouping, 16 threads in outer team

NPB Benchmark BT-MZ Class B

H. Jin (Nasa Ames), et. al.



Overview

- Why OpenMP
- Short OpenMP Introduction
- OpenMP on NUMA Machines
- OpenMP on Clusters
- Conclusion

OpenMP on Clusters

- Multiple Approaches (based on MPI, on DSM …)
  so far not very successful or uncomplete.
- Intel Cluster OpenMP on Commodity Infiniband Cluster
  - Based on TreadMarks (twin pages, sending diffs,…)
  - Integrated in commercial compiler (C++ and F95)
  - Profits from OpenMP's memory model
    (relaxed consistency, temporary view of shared data, consistency enforced at well defined synchronization points.)
  - Need to explicitely mark some shared variables (`sharable` directive)
- ScaleMP – Versatile SMP™ Architecture
  - Aggregation of multiple x86 boards into one larger system
  - Cache coherent connection through InfiniBand
  - Modified IB stack and BIOS, caching strategies
  - Single system image, virtual SMP machine
  - Aggregation of all I/O resources to the OS
- Affinity matters!

EPCC OpenMP Micro-Benchmarks

J. M. Bull. Measuring Synchronization and Scheduling Overheads in OpenMP. 1999.

	Tigerton	Opteron	CLOMP	ScaleMP (MEG)
PARALLEL FOR				
2 threads	**1.31**	**1.36**	**723.77**	**264.83**
16 threads	**5.01**	**7.17**	**4342.82**	**717.77**
BARRIER				
2 threads	**0.75**	**0.58**	**598.82**	**144.45**
16 threads	**2.55**	**2.64**	**4062.67**	**429.35**
REDUCTION				
2 threads	**1.56**	**2.05**	**932.18**	**298.06**
16 threads	**5.68**	**25.77**	**4686.00**	**801.91**

About three orders of magnitude

Overhead in microseconds [us].

Binding: 1 Thread/board for CLOMP and ScaleMP(MEG)
8 Threads/board for CLOMP and ScaleMP(MEG)

Stream Benchmark

# threads	Tigerton	Opteron (*)	CLOMP	ScaleMP (RWTH)
1	**2080.78**	**1882.24**	**3321.08**	**2674.13**
2	**4033.88**	**3665.35**	**6495.34**	**5330.22**
4	**7008.31**	**6674.57**	**10031.07**	**10439.76**
8	**7156.56**	**9629.90**	**10344.97**	**17478.77**
16	**7508.01**	**8787.33**	**10473.24**	**18666.49**

Higher Memory Bandwidth

Bandwidth in MB/s. Scattered Binding.

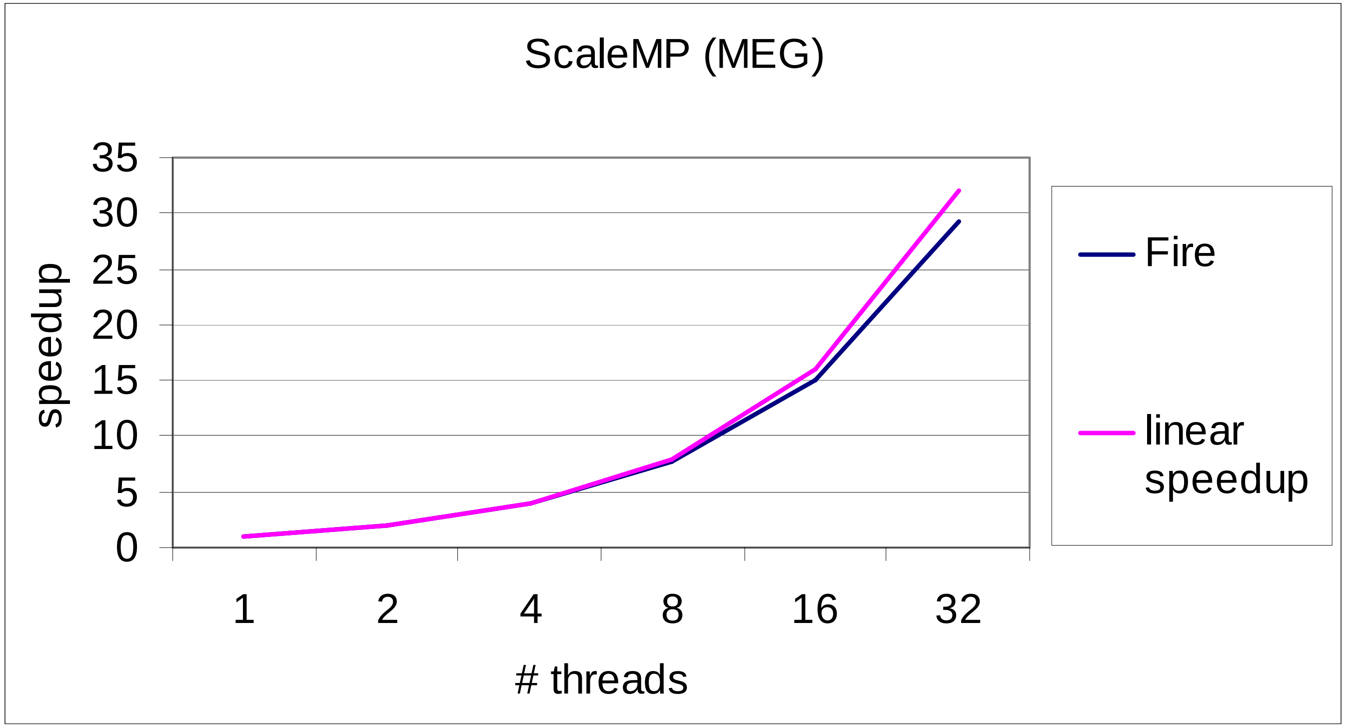
(*) We see better performance on our 4-socket Opteron machine running Solaris

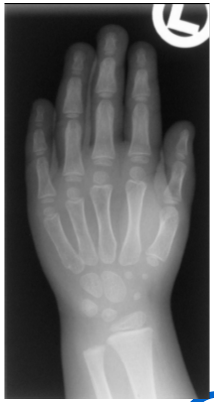
FIRE: Image Retrieval System Scales on ScaleMP (1 of 2)

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





RWTH Center for Computing and Communication

FIRE: Image Retrieval System Scales on ScaleMP (2 of 2)

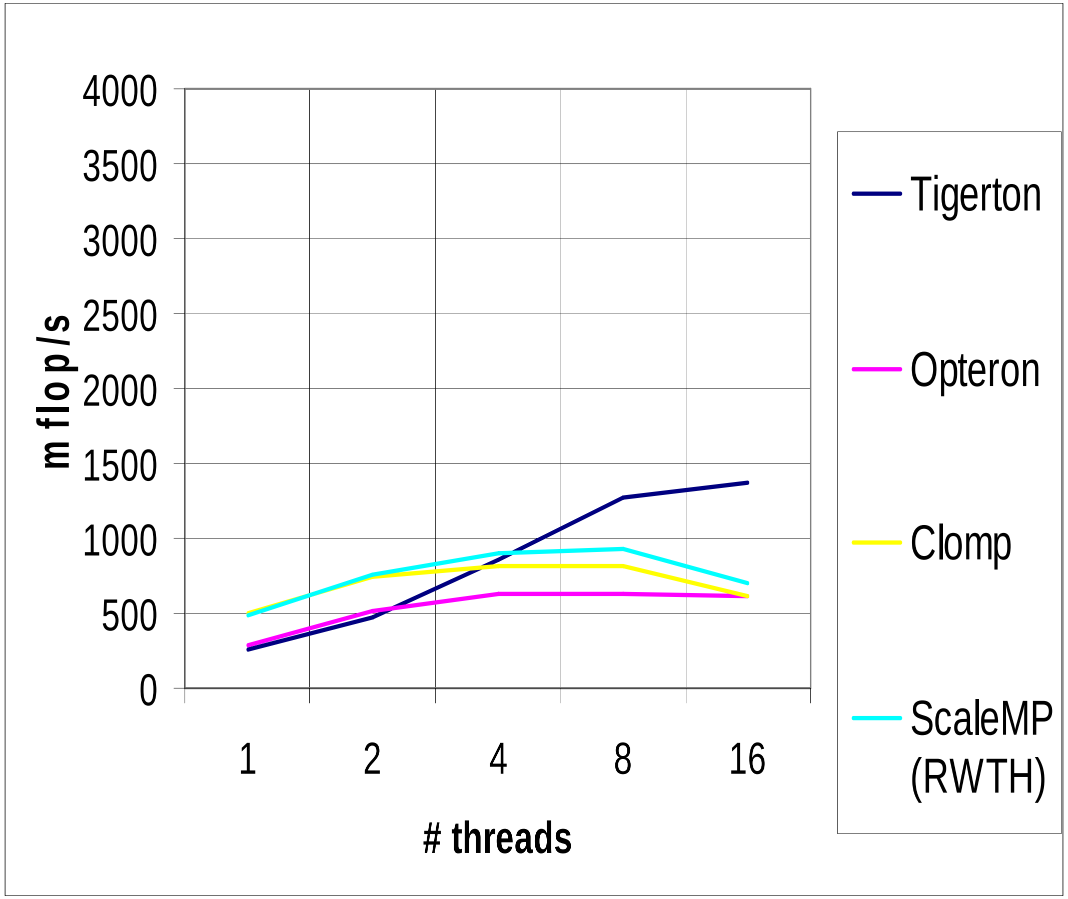
On the new 13 node system:
13 nodes with 2 Harpertown Processors at 2.5 GHz

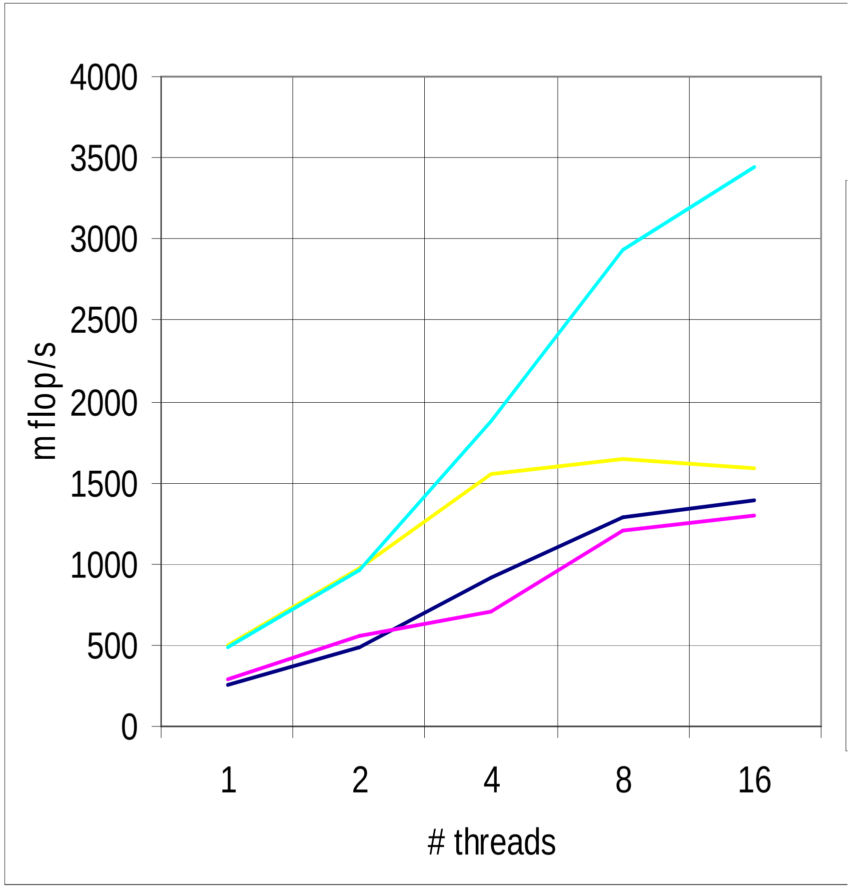
Speed-up				
#threads	outer level	inner level	nested best effort	#threads on inner x outer level
1	1,0	1,0	**1,0**	1 x 1
2	2,1	2,1	**2,1**	2 x 1
4	4,0	3,9	**4,1**	2 x 2
8	7,8	7,1	**8,0**	2 x 4
16	14,8	12,6	**15,6**	2 x 8
32	25,5		**29,9**	4 x 8
64	45,4		**53,2**	4 x 16
104			**67,1**	4 x 26

FIRE: Image Retrieval System on 144-core SF E25K Nested OpenMP improves scalability

Speedup of FIRE	Sun Fire E25K, 72 dual-core UltraSPARC-IV processors		
# Threads	Only outer level	Only inner level	Nested OpenMP
4	---	3.8	---
8	---	7.6	---
16	14.8	14.1	15.4
32	29.6	28.9	30.6
72	56.5	---	67.6
144	---	---	133.3

Sparse Matrix-Vector-Multiplication [Mflop/s] Apply Suitable Strategy!



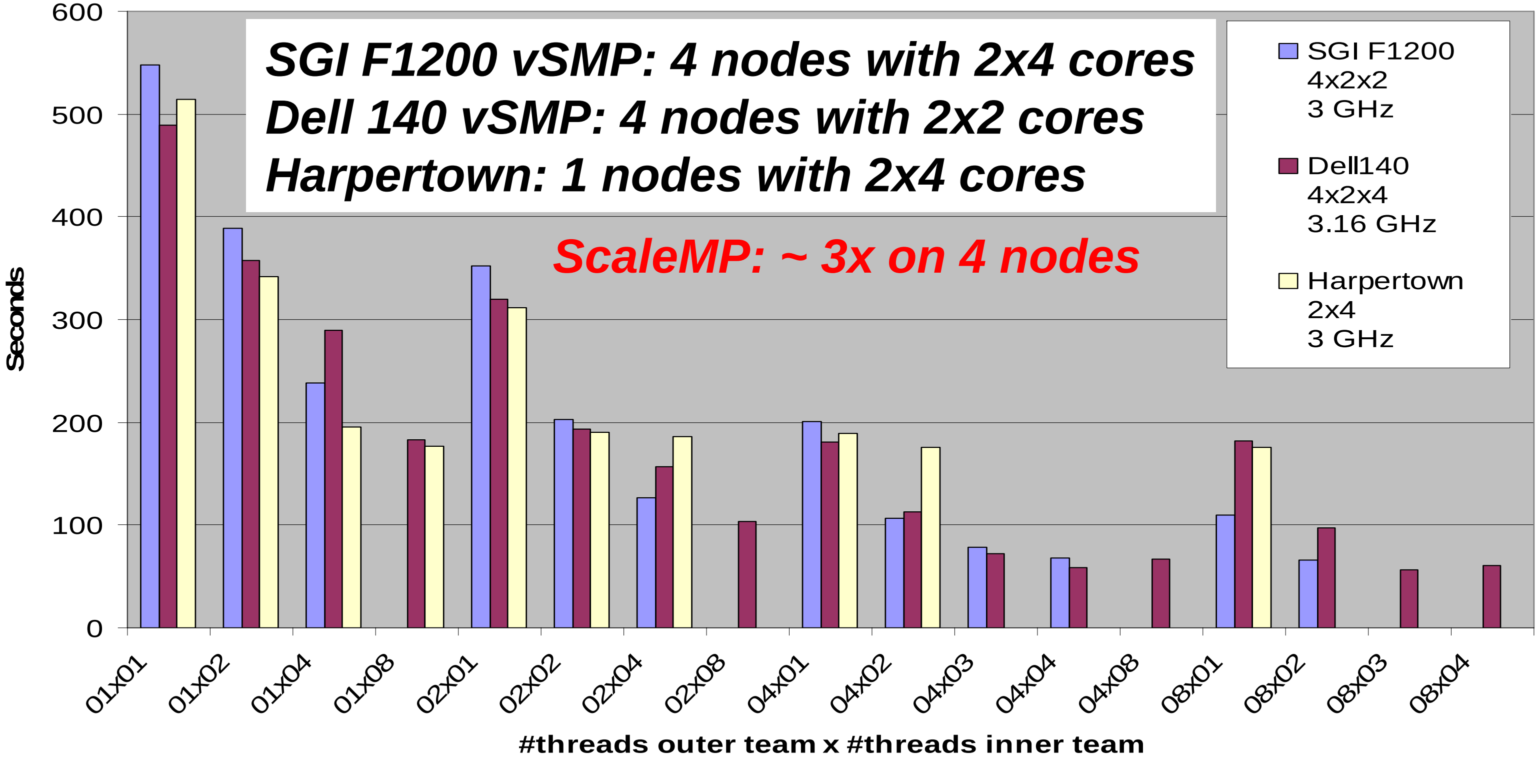


parallel loop over #rows,
dynamic loop sched.

#nonzeros
statically partitioned

SHEMAT on ScaleMP (1 of 2)

- Simulation of Coupled Flow, Heat Transfer and Transport Interaction
- BiCGStab Solver with ILU0 Preconditioner
- Nested Parallelization with OpenMP
- Explicite binding in all inner parallel regions



SHEMAT on ScaleMP (2 of 2)

Cranking up the probem size for the new 13 node system:
13 nodes with 2 Harpertown Processors at 2.5 GHz

#boards	1	2	4	8	10
best effort timing	8664,8	3357,9	2264,9	1281,8	981,6
#threads outer level	1	4	8	16	20
#threads inner level	4	1	2	2	2
#core used per board	4	2	4	4	4
speed-up on board level	**1,0**	**2,6**	**3,8**	**6,8**	**8,8**
speed-up versus 1 thread	1,4	3,6	5,3	9,4	12,2

Speed-up across the nodes good (threads on outer level don't interact much)
Speed-up withing the nodes bad (limited memory bandwidth)

Overview

- Why OpenMP
- Short OpenMP Introduction
- OpenMP on NUMA Machines
- OpenMP on Clusters
- Conclusion

Conclusion

- Scalable applications may need multiple levels of parallelization
- OpenMP suitable for a growing number of cores per node
- Combining MPI and OpenMP is getting more popular
- OpenMP on Clusters an alternative, if MPI is too hard to apply.

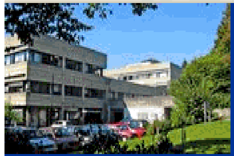
- Thread/Data Affinity is essential for OpenMP performance on ccNUMA machines and even more on Clusters
- OpenMP is hardware agnostic
- Needs control of thread and data placement
- Needs data migration, explicite and/or automatic for irregular, adaptive problems

RZ - Parallel Programming in Computational Engineering and Science (PPCES) March 2009 - HPC Tutorials - - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

http://www.rz.rwth-aachen.de/go/id/sms/?lang=en

Meistbesuchte Seiten Erste Schritte Aktuelle Nachrichten -... Marder Alarm http://openmp.org/tw... http://www.rz.rwth-a... RZ Aktuelles



RWTH Rechen- und Kommunikationszentrum

Search Help RZ Internal
A-Z Feedback RWTH

Students | Faculties and Staff | Projects and Cooperations | MATSE | Press

- News
- About Us
- Our services
- High Performance Computing
- Virtual Reality
- Events
- Help and Bugs

RZ > High Performance Computing > Help, support, Tutorials, Events > Tutorials and Events > Parallel Programming in Computational Engineering and Science (PPCES) March 2009 - HPC Tutorials -

Parallel Programming in Computational Engineering and Science (PPCES) March 2009 - HPC Tutorials -

PP ces

Monday, March 23 - Friday, March 27, 2009

Kindly supported by: intel Microsoft Sun microsystems

Date	Time	Location
Monday, March 23 Tuesday, March 24 Wednesday, March 25 Thursday, March 26 Friday, March 27	14:00 - 17:30 (*) 09:00 - 17:30 09:00 - 17:30 09:00 - 17:30 09:00 - 12:30	Center for Computing and Communication RWTH Aachen University Seffenter Weg 23 52074 Aachen

(*) We like to draw your attention to a presentation by **Horst Simon** (University of California Berkeley) on Monday morning on Future Directions in High Performance Computing 2009-2018 in the SuperC

Introduction	Related Events	Sponsors	Participants	Flyer & Poster	Agenda	Course Material
Questionnaire!!!	Links	Travel Information	Contact			

Introduction

Further information

Questionnaire:

We hope that you enjoyed the PPCES. Your feedback is welcome on the questonnaire you find here >>>

Fertig auto: en-US