RWTH AACHEN UNIVERSITY

Tasking and OpenMP Success Stories

Christian Terboven <terboven@rz.rwth-aachen.de>

23.03.2011 / Aachen, Germany

Stand: 21.03.2011

Version 2.3

Agenda

- OpenMP: Tasking
- Example: Fibonacci
- A Big SMP: our ScaleMP system(s)
- Two short (!) Success Stories

OpenMP: Tasking

How to parallelize a While-loop?

- **How would you parallelize this code?**

```
typedef list<double> dList;
dList myList;
/* fill myList with tons of items */

dList::iterator it = myList.begin();
while (it != myList.end())
{
     *it = processListItem(*it);
     it++;
}
```

- **One possibility: Create a fixed-sized array containing all list items and a parallel loop running over this array**
  Concept: Inspector / Executor

How to parallelize a While-loop!

RWTH AACHEN UNIVERSITY

- **Or: Use Tasking in OpenMP 3.0**

```
#pragma omp parallel
{
#pragma omp single
{
   dList::iterator it = myList.begin();
   while (it != myList.end())
   {
#pragma omp task
      {
        *it = processListItem(*it);
      }
        it++;
   }
}
}
```

- **All while-loop iterations are independent from each other!**

The task directive

C/C++

```
#pragma omp task [clause [[,] clause] ... ]
... structured block ...
```

- **Each encountering thread creates a new Task**
  - Code and data is being packaged up
  - Tasks can be nested
    - Into another Task directive
    - Into a Worksharing construct
- **Data scoping clauses:**
    - `shared`(*list*)
    - `private`(*list*)
    - `firstprivate`(*list*)
    - `default`(*shared* | *none*)

Task synchronization (1/2)

- **At OpenMP `barrier` (implicit or explicit)**
  - All tasks created by any thread of the current *Team* are guaranteed to be completed at barrier exit

- **Task barrier: `taskwait`**
  - Encountering Task suspends until child tasks are complete
    - Only direct childs, not descendants!

C/C++

```
#pragma omp taskwait
```

Task synchronization (2/2)

- **Simple example of Task synchronization in OpenMP 3.0:**

```
#pragma omp parallel num_threads(np)
{
#pragma omp task
   function_A();
#pragma omp barrier
#pragma omp single
   {
#pragma omp task
      function_B();
   }
}
```

np Tasks created here, one for each thread

All Tasks guaranteed to be completed here

1 Task created here

B-Task guaranteed to be completed here

Tasks in OpenMP: Data Scoping

- **Some rules from *Parallel Regions* apply:**
  - Static and Global variables are shared
  - Automatic Storage (local) variables are private
- **If no `default` clause is given:**
  - Orphaned Task variables are `firstprivate` by default!
  - Non-Orphaned Task variables inherit the `shared` attribute!

  → Variables are `firstprivate` unless `shared` in the enclosing context
- **So far no verification tool is available to check Tasking programs for correctness!**

Example: Fibonacci

Recursive approach to compute Fibonacci

```
int main(int argc,
         char* argv[])
{
   [...]
   fib(input);
   [...]
}
```

```
int fib(int n)   {
   if (n < 2) return n;
   int x = fib(n - 1);
   int y = fib(n - 2);
   return x+y;
}
```

- **On the following slides we will discuss three approaches to parallelize this recursive code with Tasking.**

First version parallelized with Tasking (omp-v1)

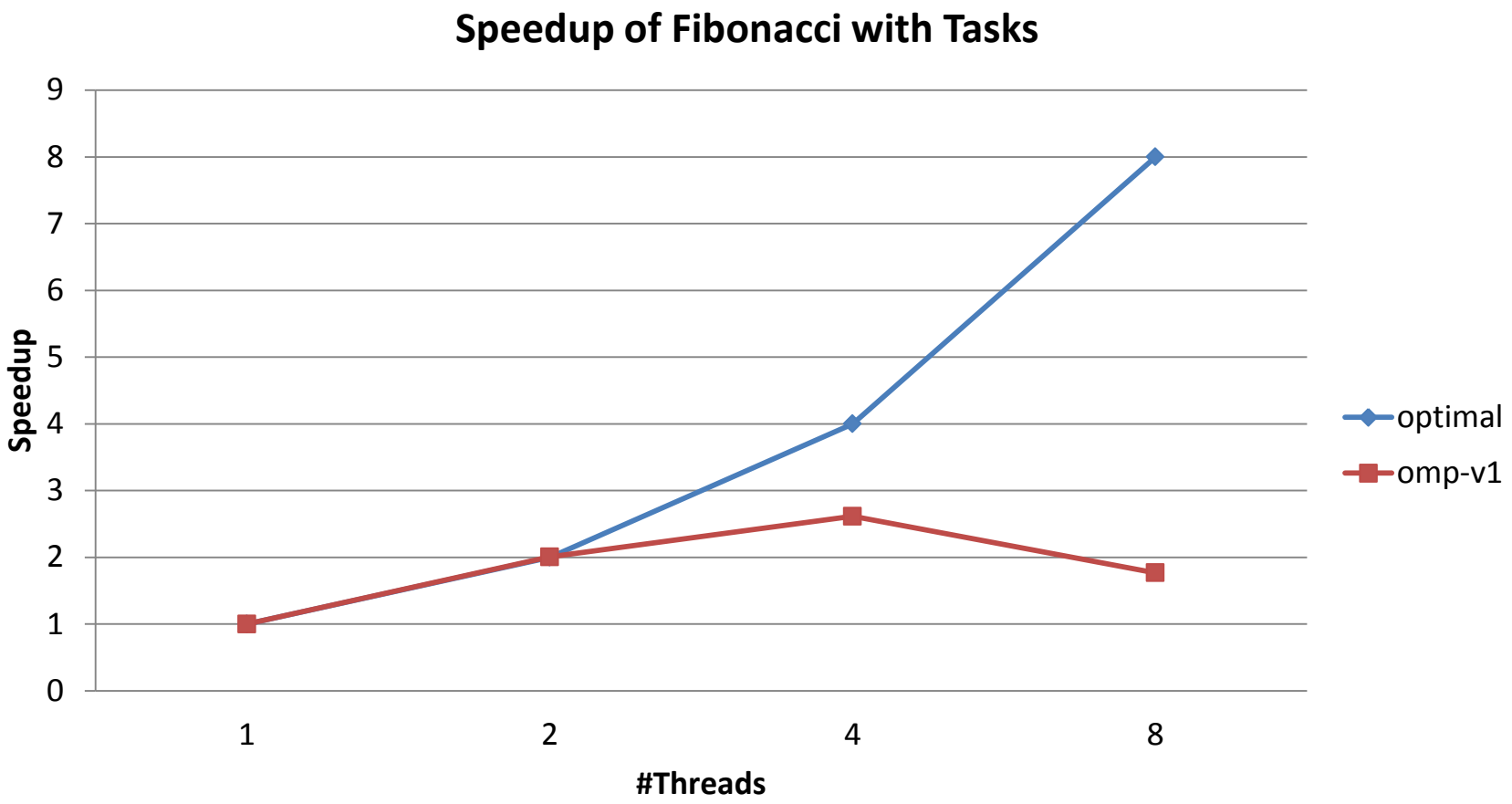
```
int main(int argc,
         char* argv[])
{
   [...]
#pragma omp parallel
{
#pragma omp single
{
   fib(input);
}
}
   [...]
}
```

```
int fib(int n)   {
   if (n < 2) return n;
int x, y;
#pragma omp task shared(x)
{
   x = fib(n - 1);
}
#pragma omp task shared(y)
{
   y = fib(n - 2);
}
#pragma omp taskwait
   return x+y;
}
```

- Only one Task / Thread enters `fib()` from `main()`, it is responsable for creating the two initial work tasks
- Taskwait is required, as otherwise `x` and `y` would be lost

Scalability measurements (1/3)

- Overhead of task creation prevents better scalability!



Improved parallelization with Tasking (omp-v2)

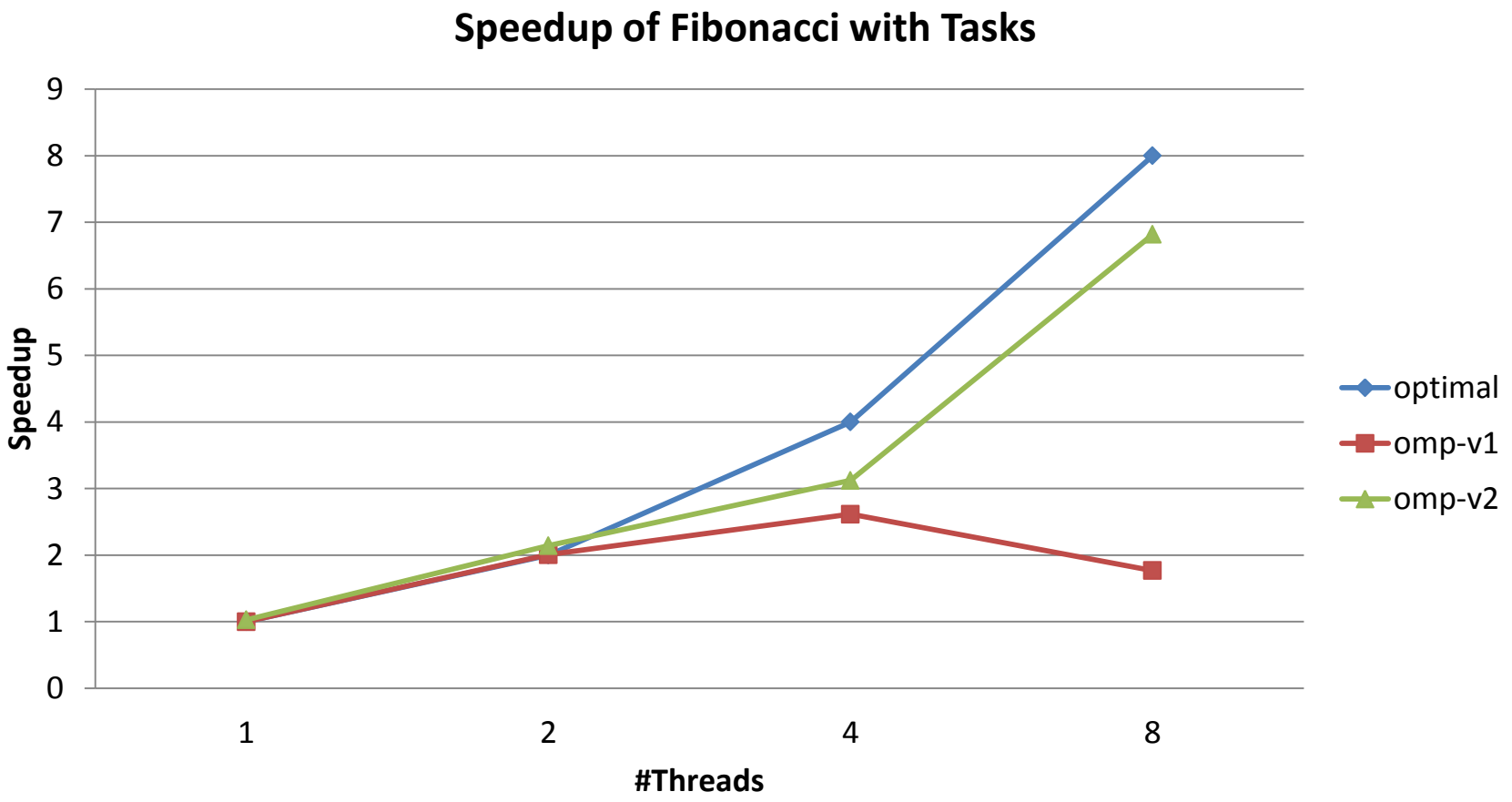
- **Improvement: Don‘t create yet another task once a certain (small enough) `n` is reached**

```
int main(int argc,
         char* argv[])
{
   [...]
#pragma omp parallel
{
#pragma omp single
{
   fib(input);
}
}
   [...]
}
```

```
int fib(int n)   {
   if (n < 2) return n;
int x, y;
#pragma omp task shared(x) \
   if(n > 30)
{
   x = fib(n - 1);
}
#pragma omp task shared(y) \
   if(n > 30)
{
   y = fib(n - 2);
}
#pragma omp taskwait
   return x+y;
}
```

Scalability measurements (2/3)

- Speedup is ok, but we still have some overhead when running with 4 or 8 threads



Improved parallelization with Tasking (omp-v3)

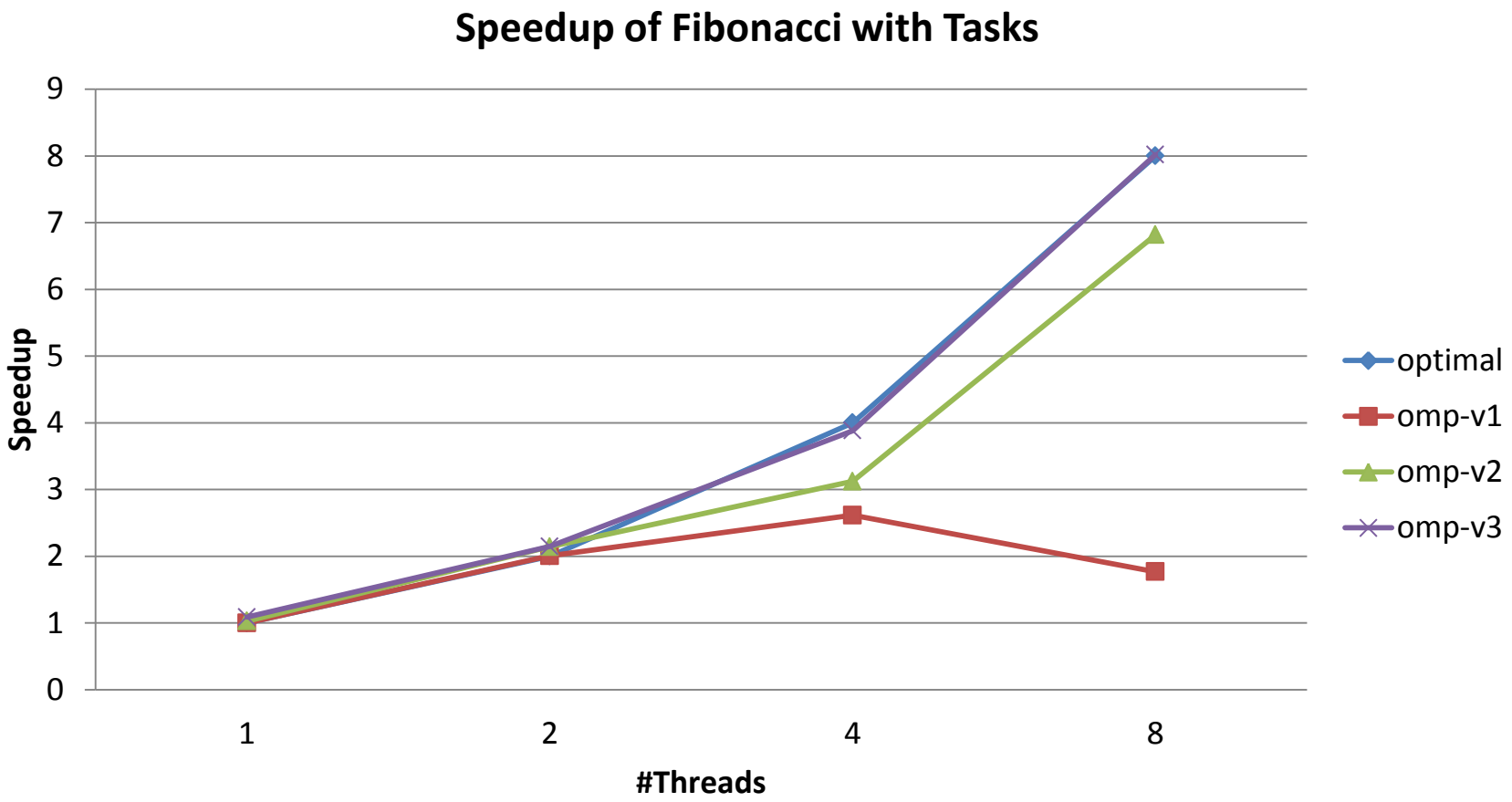
- **Improvement: Skip the OpenMP overhead once a certain `n` is reached (no issue w/ production compilers)**

```
int main(int argc,
         char* argv[])
{
   [...]
#pragma omp parallel
{
#pragma omp single
{
   fib(input);
}
}
   [...]
}
```

```
int fib(int n)   {
   if (n < 2) return n;
   if (n <= 30)
      return serfib(n);
int x, y;
#pragma omp task shared(x)
{
   x = fib(n - 1);
}
#pragma omp task shared(y)
{
   y = fib(n - 2);
}
#pragma omp taskwait
   return x+y;
}
```

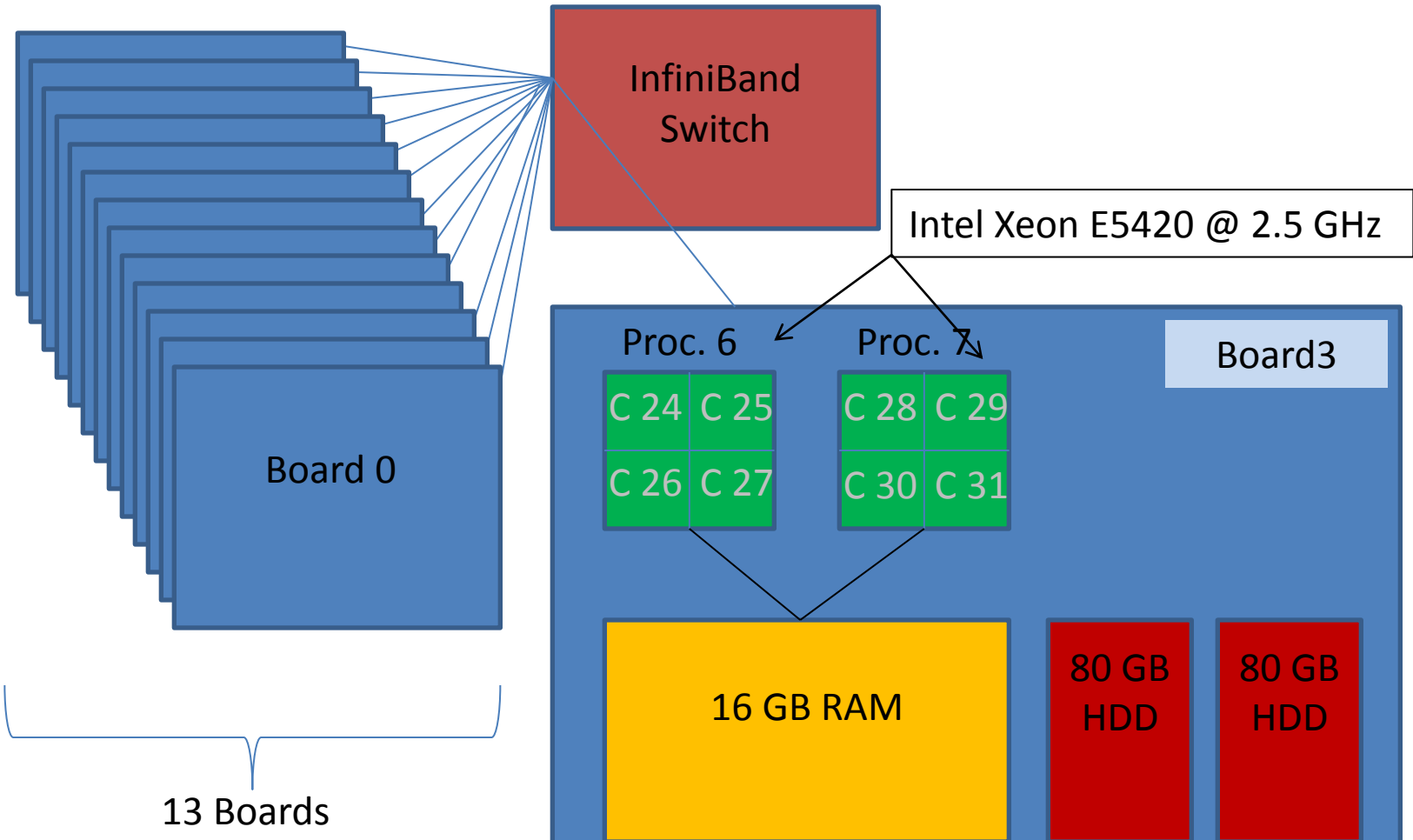
Scalability measurements (3/3)

- Everything ok now ☺



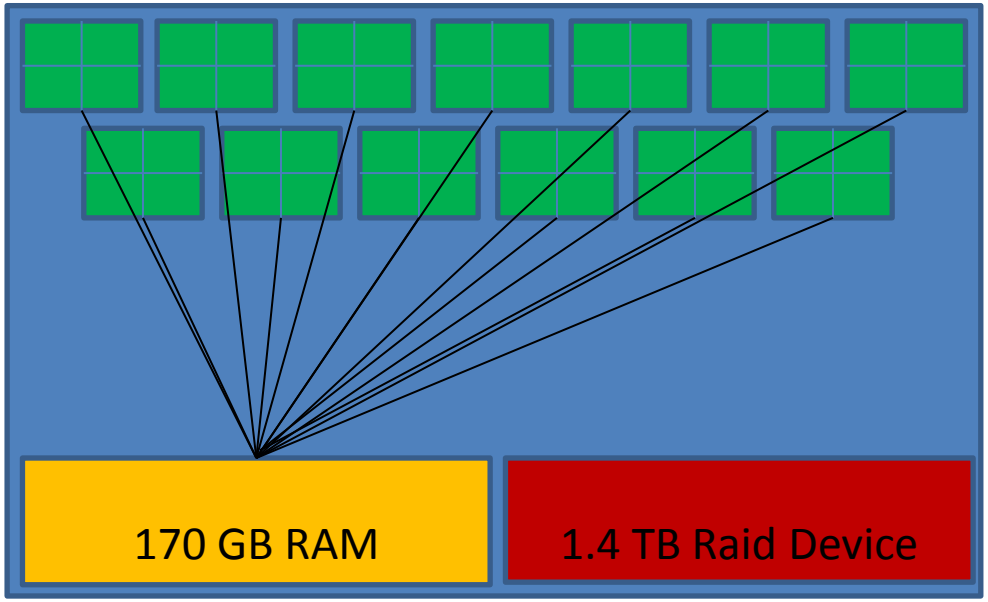
A Big SMP: ScaleMP system(s)

- In total 104 cores,170 GB of memory, 1.4 TB of disc space



ScaleMP: Single System Image

- **vSMP: Cache-coherent aggregation of physical resources via the InfiniBand network (employing virtualization techniques)**
  - The OS view (Single System Image):

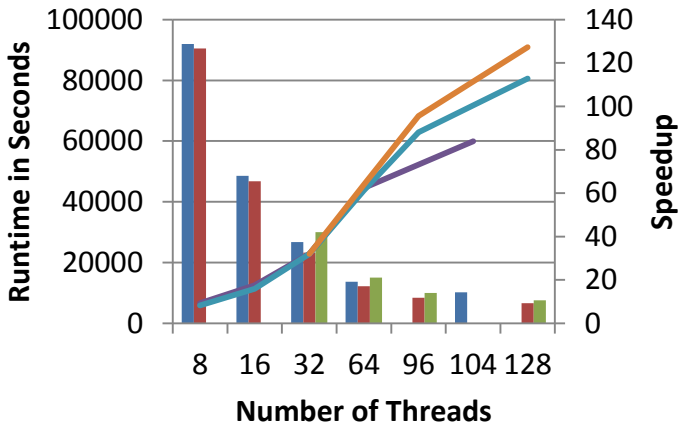


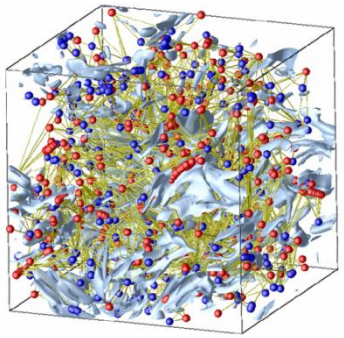
- **Shared-Memory parallelization on a cluster of x86-based boards**
  - Cheaper and more flexible than hardware-based solutions
  - Strong cc-NUMA characteristics

Two short Success Stories

Turbulent Flows Analysis of DNS Output

- **Grand challenge problem**
  - Understanding geometric structures of fine scale turbulences
  - Direct Numerical Simulation (DNS) on 2048^3 Cube took 4 months on BlueGene in Jülich w/ 8000 cores
- **Analysis of DNS output data problematic**
  - Not suited for MPI
  - Need for >300 GB of main memory
  - Estimated serial runtime > 1 week
- **OpenMP parallelization with NUMA optimization delivered ~100x speed-up**
  - On virtual SMP-Cluster (Software from ScaleMP)
  - On SGI Altix at LRZ Munich
- **Well-versed HPC Consultant necessary**
  - Green IT: NUMA Optimization





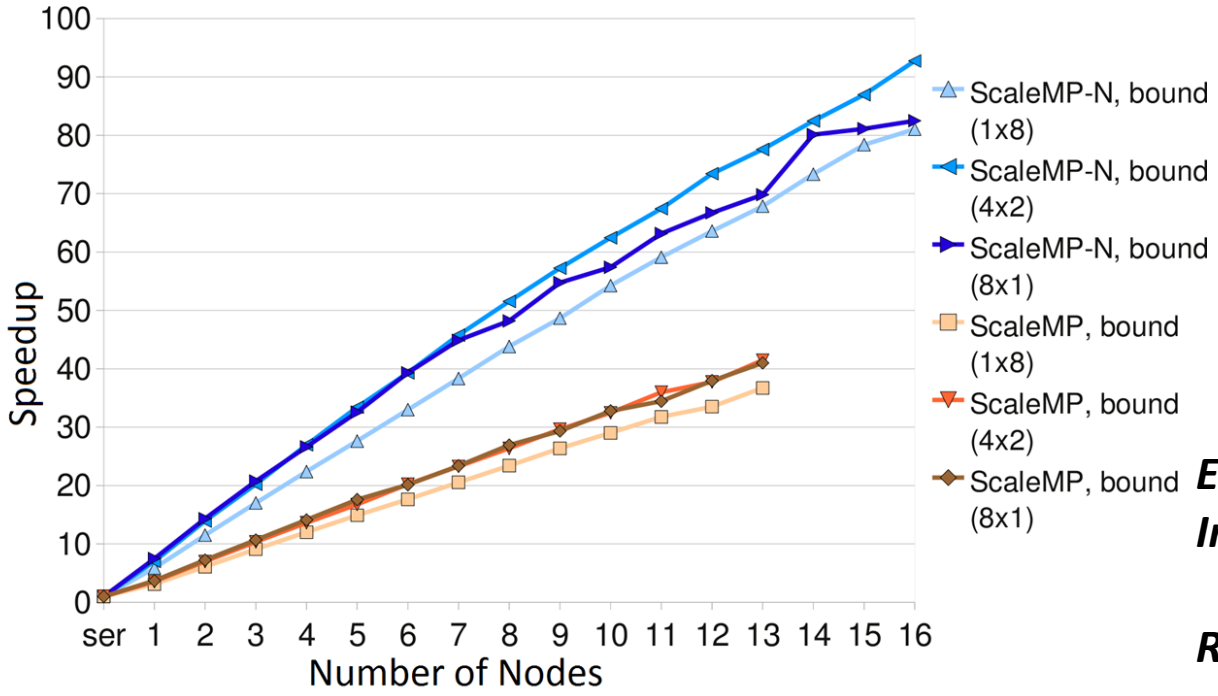
Institute for Combustion Technology,
RWTH Aachen University

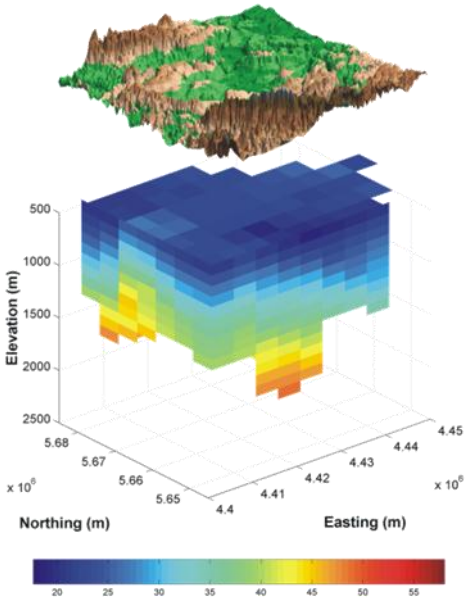
Geothermal Simulation of CO_2 Storage

SHEMAT-Suite

- Simulating Groundwater flow, heat transfer and transport of reactive solutes
- 10x speed-up with 2nd level of OpenMP

  on virtual SMP-Cluster (Software from ScaleMP)





E.ON Energy Research Center
Inst. of Appl. Geophysics and Geothermal Energy,
RWTH Aachen University

The End

Thank you for your attention.