OpenMP Programming on ScaleMP

Dirk Schmidl
schmidl@rz.rwth-aachen.de

MPI vs. OpenMP

RWTH AACHEN UNIVERSITY

MPI

- distributed address space
- explicit message passing
- typically code redesign

- e.g. jacobi example parallelized with 35 modified lines

=> harder to program

OpenMP

- shared address space
- read or write to shared variables
- incremental parallelization

- e.g. jacobi example parallelized with 3 modified lines

=> easier for fist parallelization

Cluster vs. SMP

Clusters:

- many nodes
- network interconnect
- distributed address space
- programmable with MPI

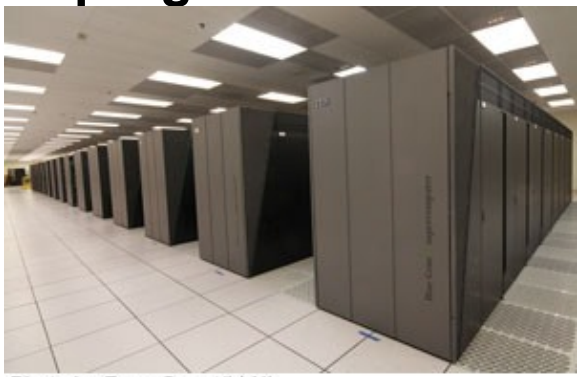


Photo by Ross Gaunt/LLNL

- largest Cluster: Sequoia
- IBM Blue Gene/Q
- 1,572,864 compute cores
- 20.1 PFLOP/s peak performance

SMPs

- one big shared address space
- one operating system instance
- programmable with OpenMP or MPI …

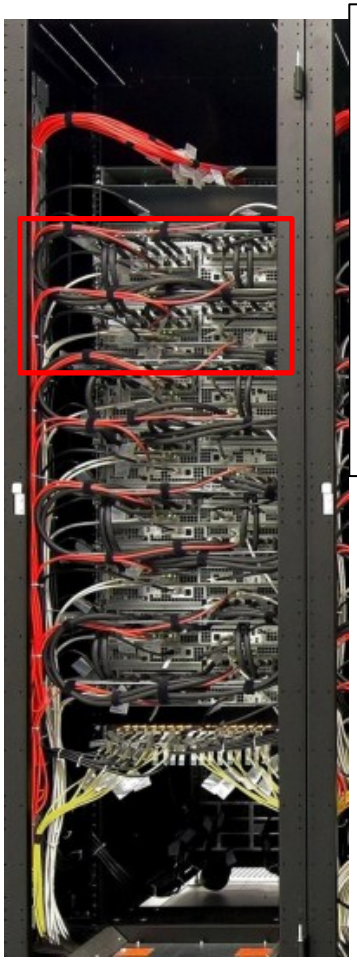
largest SMP:

SGI Altix UV

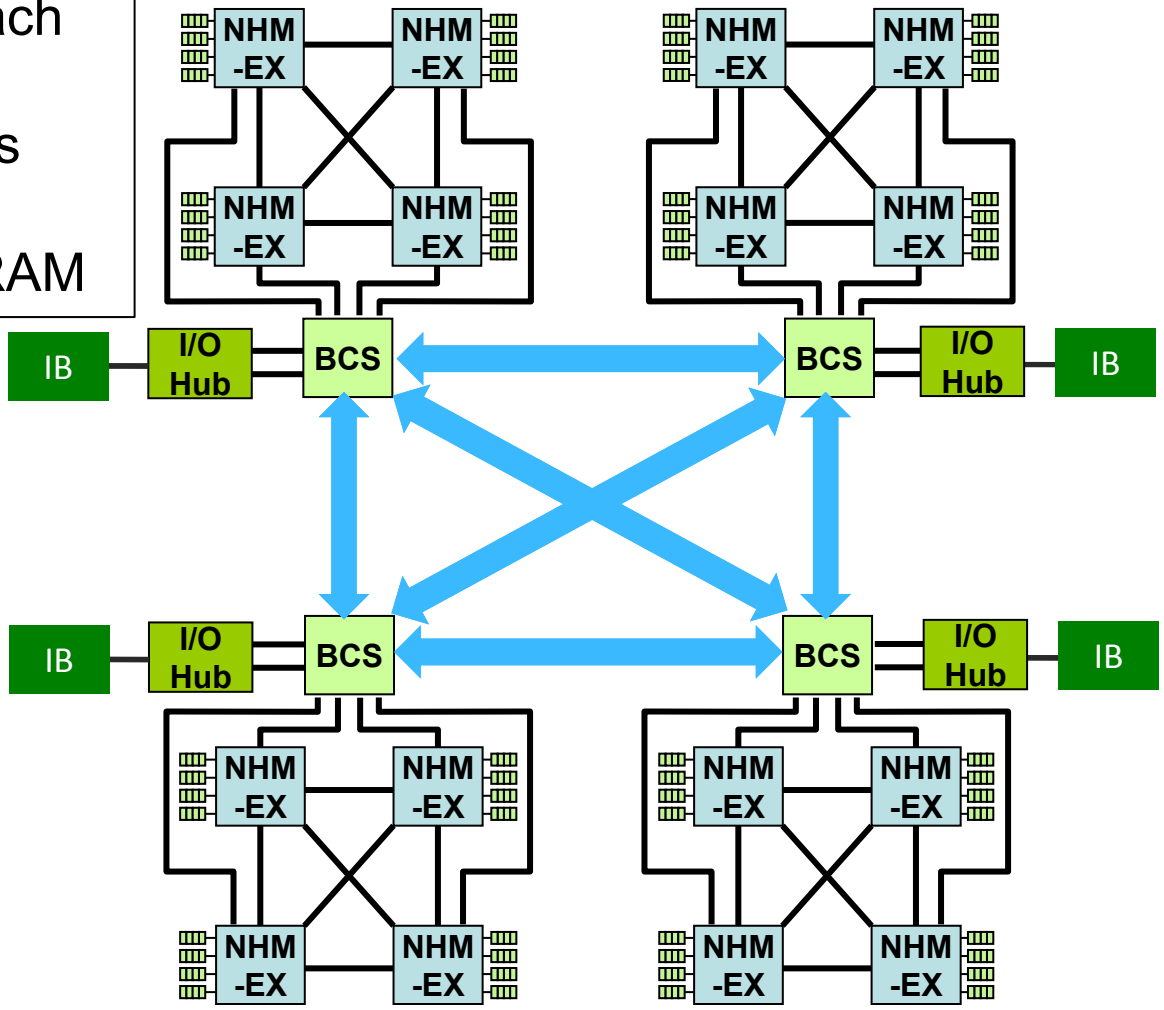
- 2048 Cores
- max 64 GB shared memory
- ~47 TFLOP/s peak performance

www.sgi.com/uv

BCS systems @ RWTH Aachen

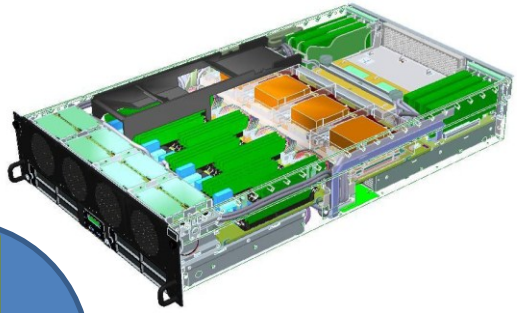


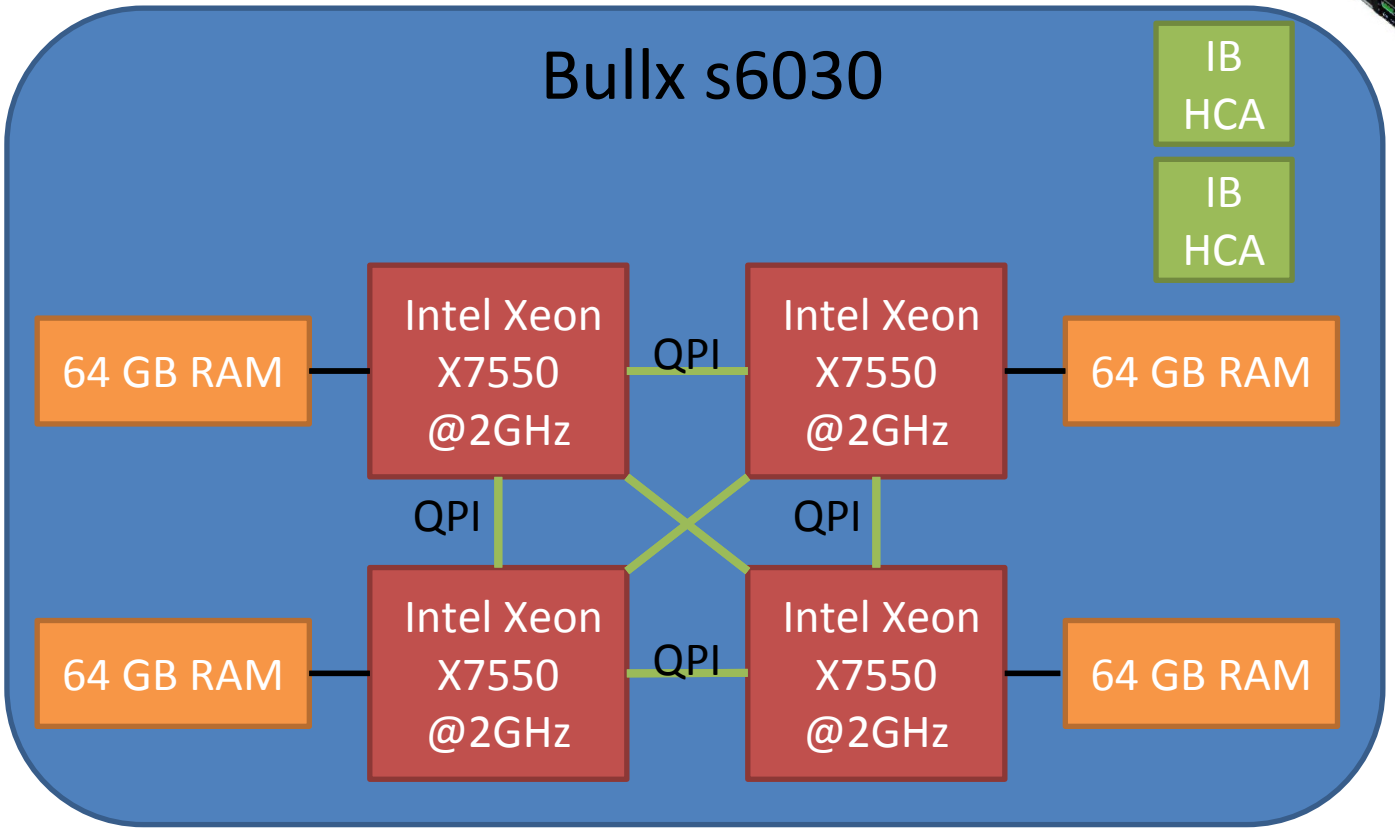
High-level overview
- 4 boards
- 4 sockets each
- 2 levels of NUMAness
- 128 Cores
- up to 2 TB RAM



Source: D. Gutfreund, Bull GmbH

ScaleMP

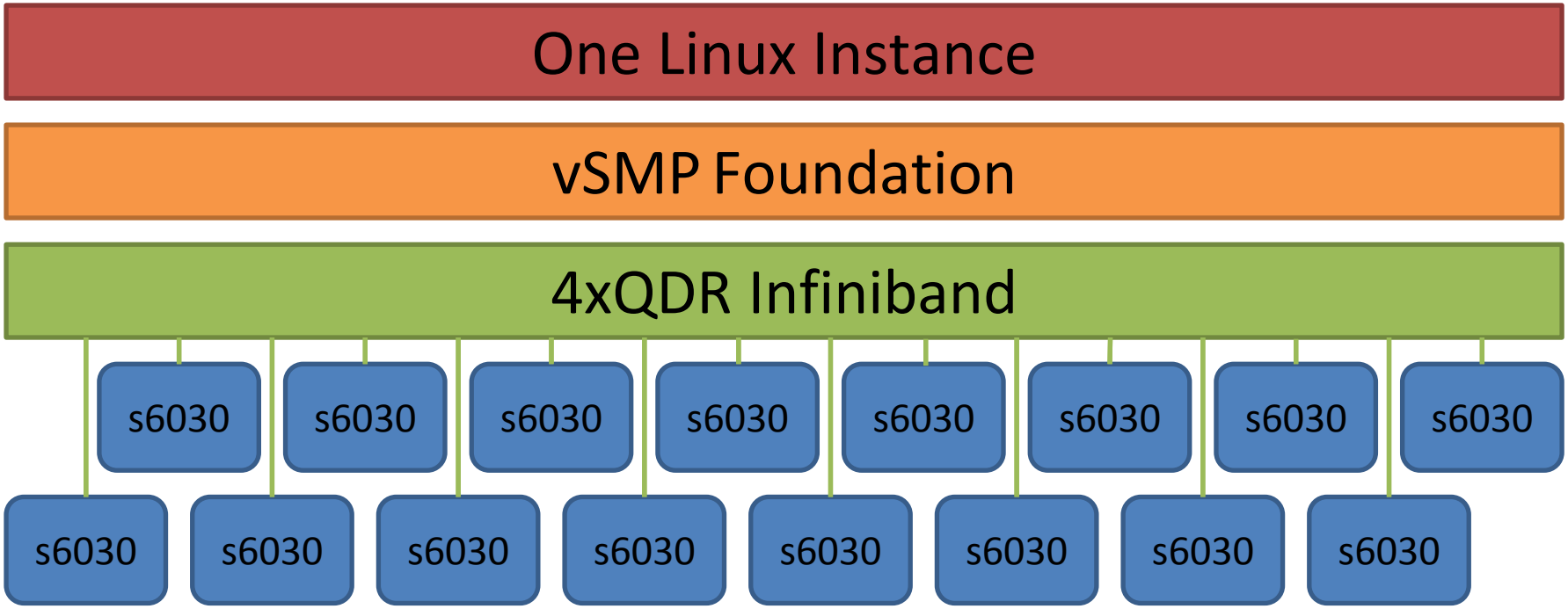



Bullx s6030
IB HCA
IB HCA
64 GB RAM
Intel Xeon X7550 @2GHz
QPI
Intel Xeon X7550 @2GHz
64 GB RAM
QPI
QPI
64 GB RAM
Intel Xeon X7550 @2GHz
QPI
Intel Xeon X7550 @2GHz
64 GB RAM

ScaleMP

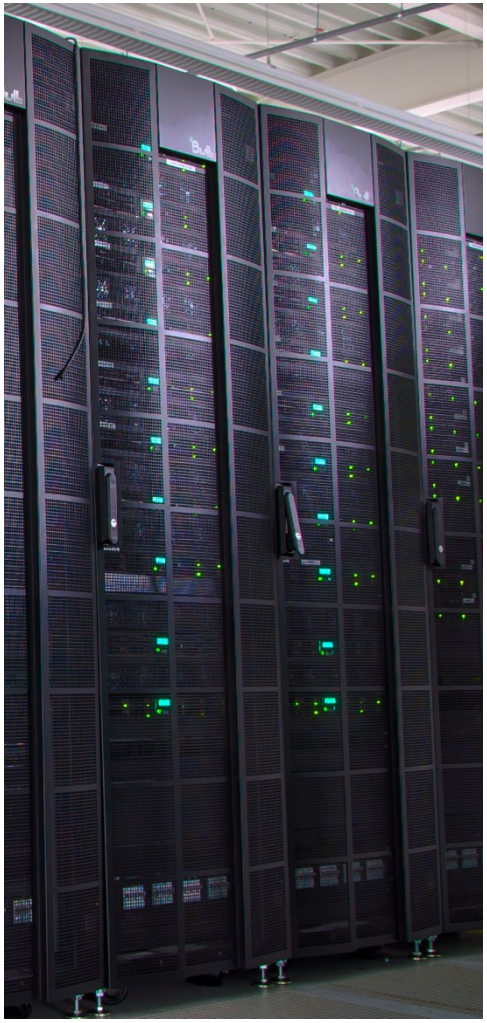
vSMP Foundation:

- creates the single system image
- internally uses prefetching and caching mechanisms to optimize memory accesses
- most of the mechanisms work on a 4k basis



ScaleMP system @ RWTH Aachen

Recourse	ScaleMP system
CPUs	64 Xeon X7550
Cores	512
Clockrate	2 GHz
Memory	~ 4 TB
Memory bandwidth	886 GB/s
Accumulated L3 Cache	1.2 GB
Peak Performance	4 TFLOP/s
vSMP version	4.0

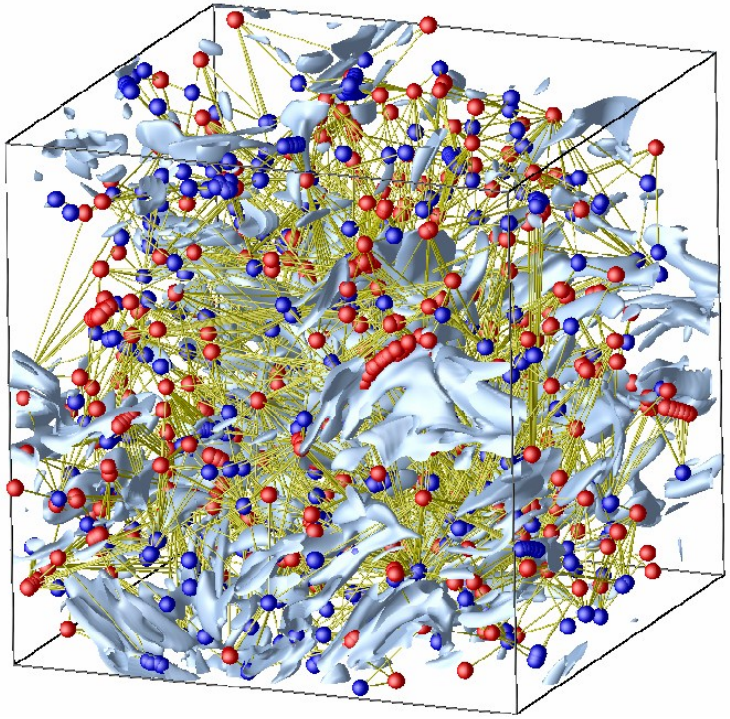


Operated in batch mode under LSF:

- jobs scheduled board wise
- a binding script restricts execution to a set of boards to minimize interference between different jobs

Case Study: TrajSearch

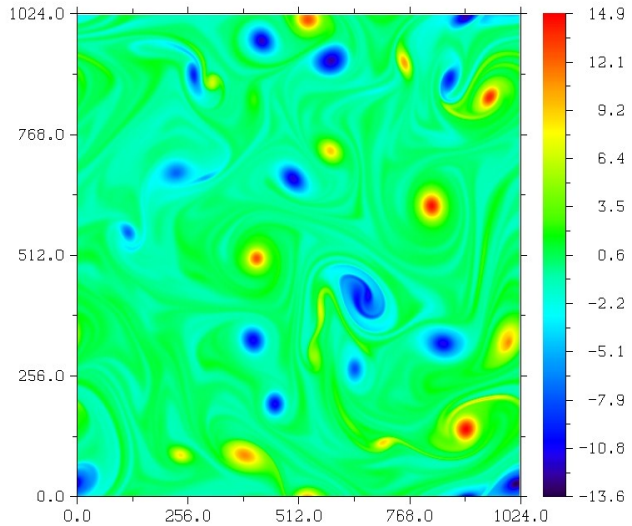
- **Direct Numerical Simulation of three dimensional turbulent flow field produces large output arrays**
- **16384 Procs@BlueGene ~ ½ year of computation produced 2048^3 output grid (320GB)**
- **Trajectory Analysis (TrajSearch) implemented with OpenMP was running fine on standard 8-Core System for 1024^3 grid cells data**
- **Analysis did not work for 2048^3 dataset**

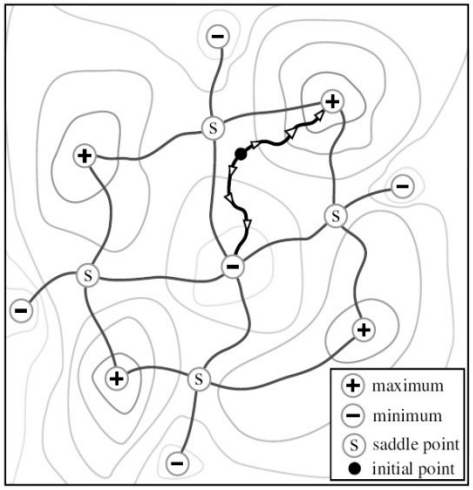


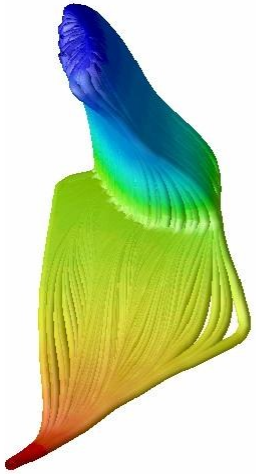
Institute for Combustion Technology
Chair for Operating Systems
Center for Computing and Communication

Case Study: TrajSearch

- **Post-processing code for dissipation element analysis**
- **Follows Trajectories starting at each grid cell in direction of ascending and descending gradient of an passive scalar field**
- **Trajectories lead to a local maximum or minimum respectively**
- **The composition of all grid cells of which trajectories end in the same pair of extremal points defines a dissipation element**







Reducing synchronization

RWTH AACHEN UNIVERSITY

OpenMP Synchronization:

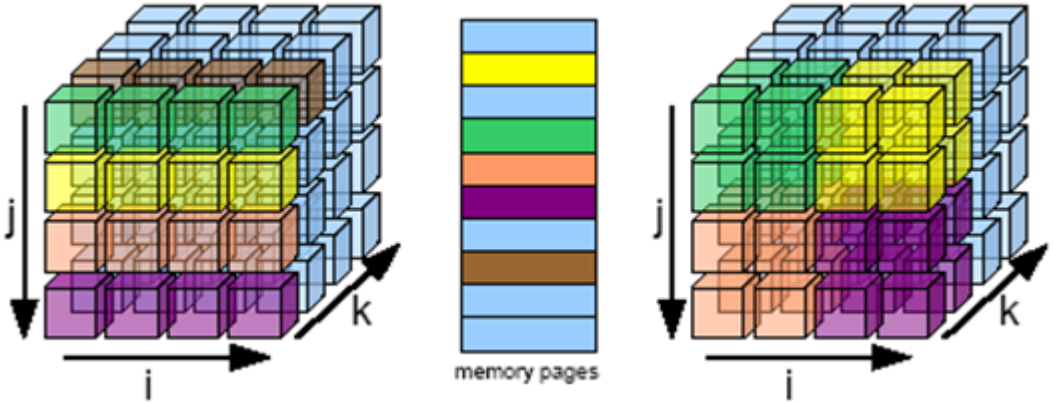
- **Global list of extremal points (exp)**
  - → Use individual local copies
  - → Merge local copies once only
- **Trajectory count field (trajcount)**
  - → Locally buffered increment operations
  - → Continuously flushed

OS/Runtime hidden synchronization

- **Random Number generation**
  - → Implement own Random Number Generator
- **Memory Allocation**
  - → switch to kmp_malloc

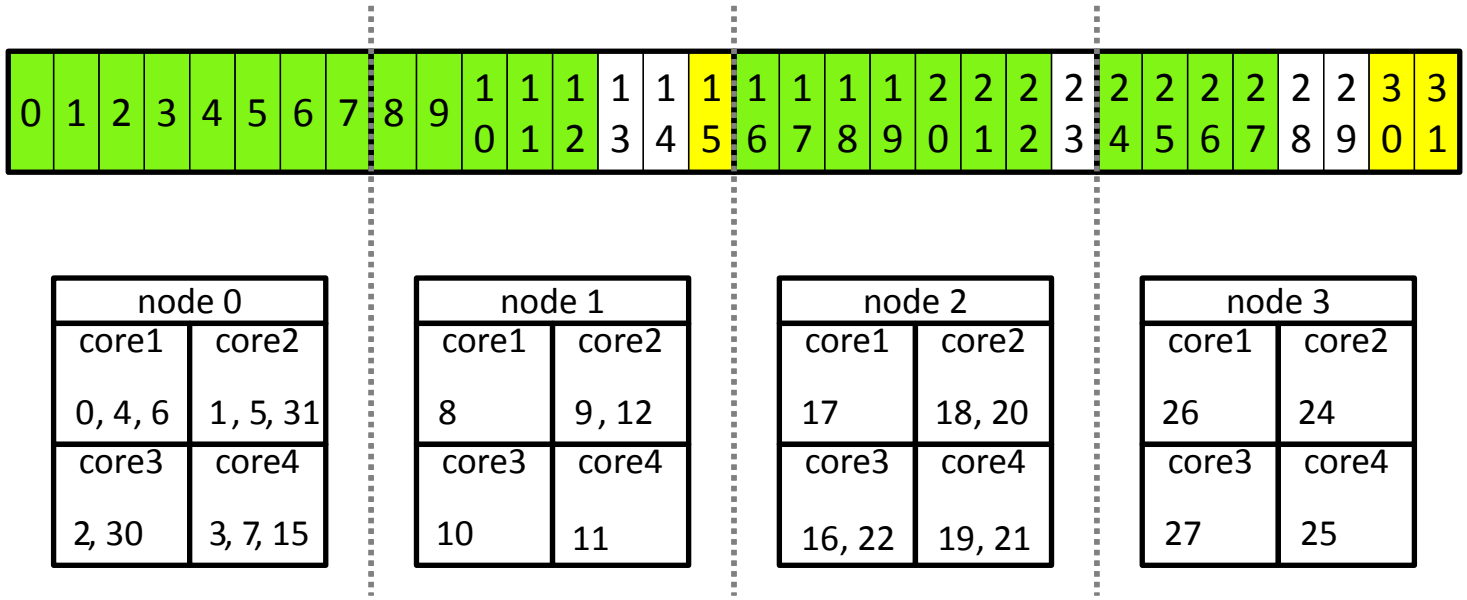
Reducing remote memory accesses

- **column-major order arrays**
  - → Trajectory searches need neighbors in 3D and move in a-priori unknown directions
  - → only data in one direction is loaded
  - → nearly no reuse of data on a loaded remote page
- **3D Memory blocks**
  - → cacheline-locality in a three-dimensional fashion
  - → 8x8x8 double Elements are combined to a 4KByte block
  - → Data has to be reordered in memory



Reducing remote memory accesses

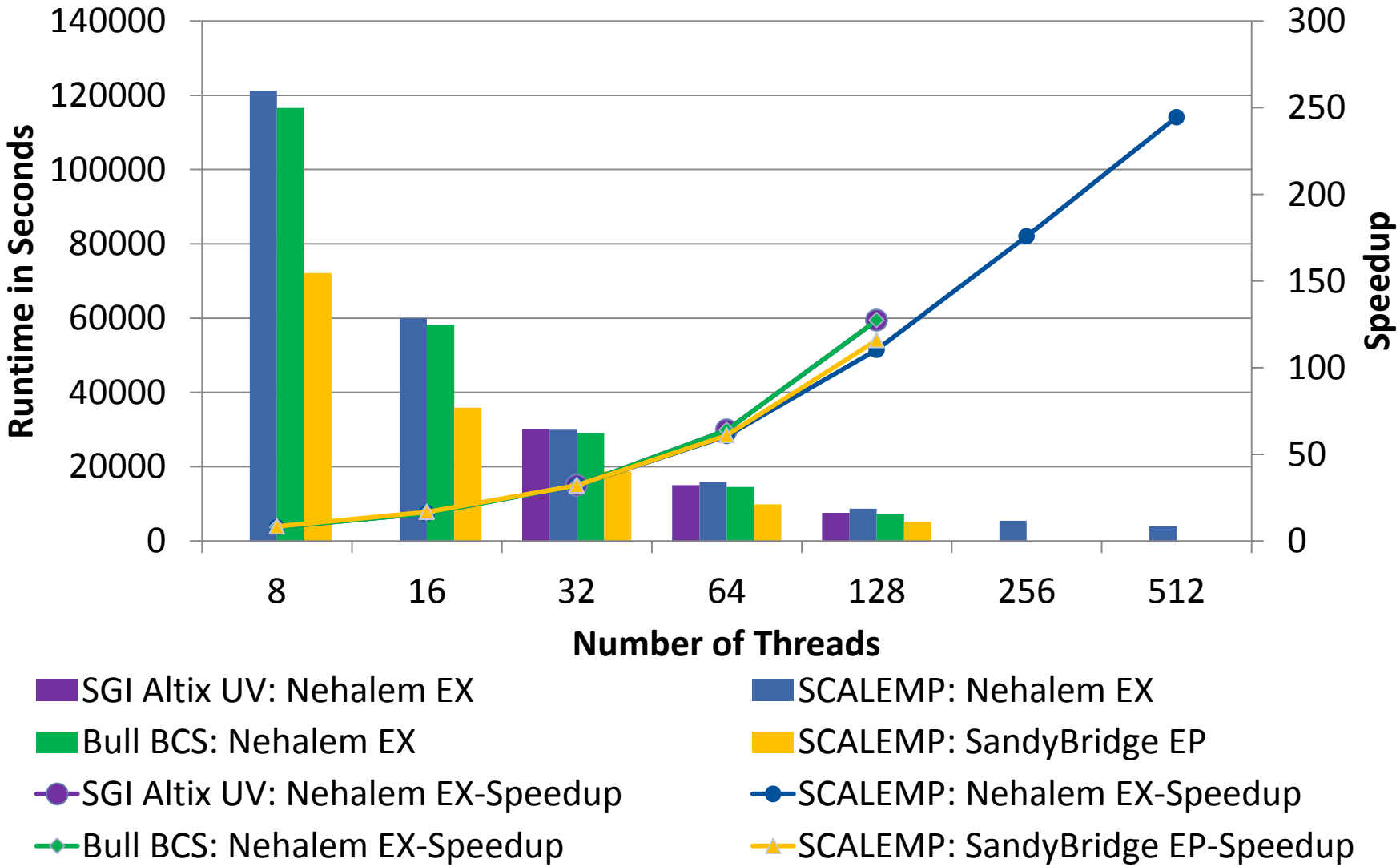
- **Adaptive NUMA-scheduler**
  - → Combines principle of static and dynamic scheduling
  - → Static layout defines how work should be distributed across nodes
  - → Work is distributed dynamically between threads on the same node
  - → Idle nodes will take work from the node with least progress
  - → To reduce interference work of foreign nodes will get iterations from the highest index backwards



Machine Comparison

Recources	SCALEMP: Nehalem EX	SCALEMP: SandyBridge EP	Bull BCS: Nehalem EX	SGI Altix UV: Nehalem EX
Processors	64 Xeon X7550	16 Xeon E5-2670	16 Xeon X7550	32 Xeon X7550
Sockets per board	4	2	4	2
Cores	512	128	128	256
Clockrate	2.00 GHz	2.60 GHz	2.00 GHz	2.00 GHz
RAM	~ 4 TB	512 GB	256 GB – 2 TB	256 GB
vSMP version	4.0	5.0	-	-
Interconnect	Infiniband	Infiniband	Bull Coherence Switch	NUMAlink

Performance Comparison: medium dataset



Conclusion

- **OpenMP programming is easy but limited to smaller machines.**
- **Getting good performance can be hard.**
- **Big SMP machines like the ScaleMP machine can deliver good performance.**
- **An incremental approach can be used instead of completely rewriting the simulation code.**