Parallel sparse matrix-vector multiplication as a test case for hybrid MPI+OpenMP programming

Gerald Schubert[1], Georg Hager[2], Holger Fehske[1], Gerhard Wellein[2,3]

[1]Institute of Physics, University of Greifswald, Germany

[2]Erlangen Regional Computing Center (RRZE), Germany

[3]Department for Computer Science, Friedrich-Alexander-University Erlangen-Nuremberg, Germany

Workshop on Large-Scale Parallel Processing 2011 (LSPP2011), May 20, 2011, Anchorage, AK

KONWIHR II

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Agenda

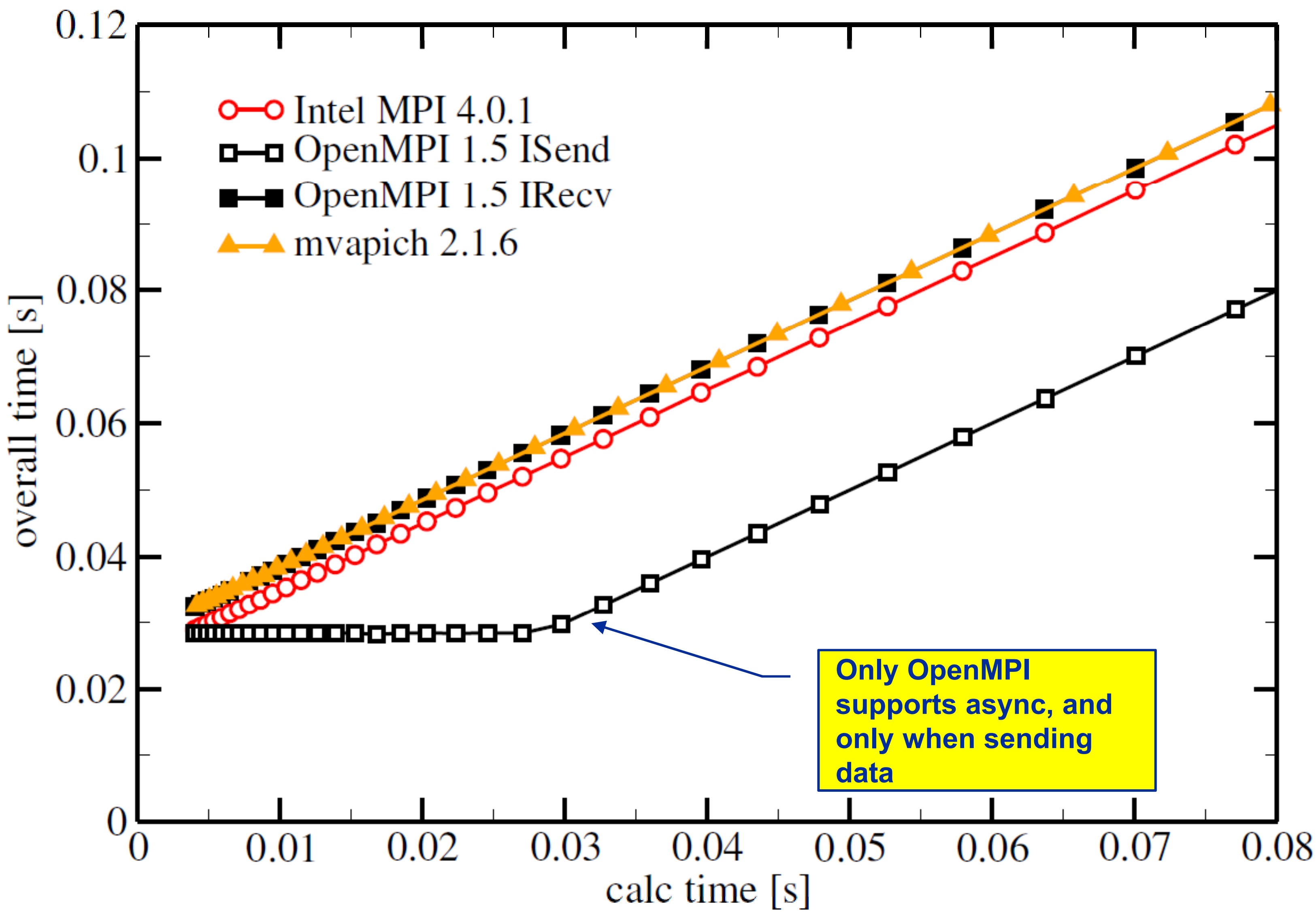
- **MPI nonblocking != asynchronous**
  - Well known for along time, but need to check once in a while…
- **Options for really asynchronous communication**
  - MPI sometimes does it ok
  - Separate explicit communication thread
  - Use something else that supports async by definition
- **Example: Sparse matrix-vector multiply (spMVM)**
  - Motivation
  - Properties of the CRS format
  - Node performance model on different multicore hardware
  - Distributed-memory parallelization
  - Hiding communication: “vector mode” vs. “task mode”
- **Results**
  - Westmere EP InfiniBand cluster (plus some Cray XE6 results)

- Is nonblocking automatically asynchronous? → Simple benchmark:

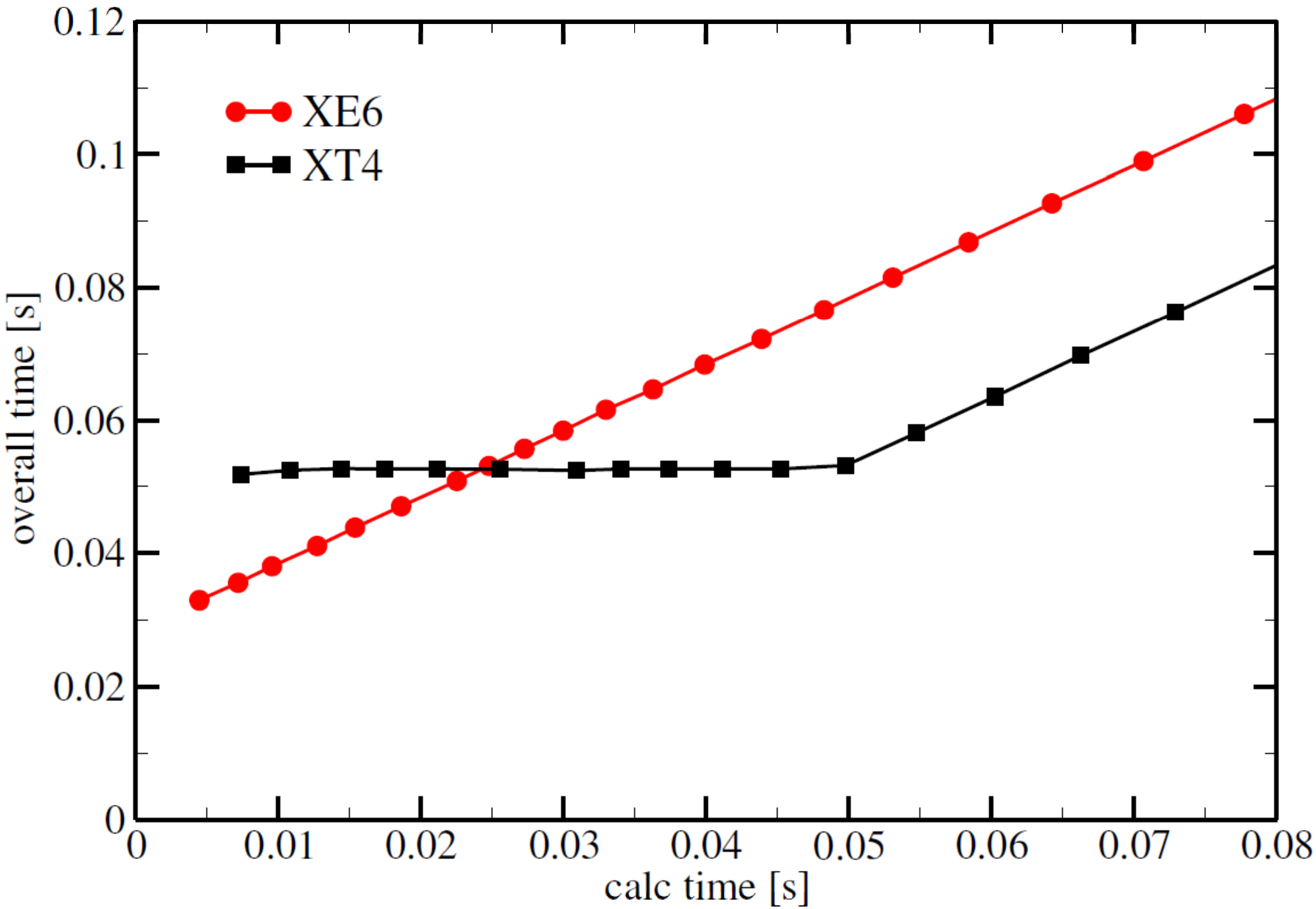
```
if(rank==0) {
  stime = MPI_Wtime();
  MPI_Irecv(rbuf,mcount,MPI_DOUBLE,1,0,
     MPI_COMM_WORLD,&req);
  do_work(calctime);
  MPI_Wait(req, &status);
  etime = MPI_Wtime();
  cout << calctime << "␣" << etime-stime << endl;
} else {
  MPI_Send(sbuf,mcount,MPI_DOUBLE,0,0,
     MPI_COMM_WORLD);
}
```

- For low calctime, **execution time is constant if async works**!
- Benchmark: **80 MByte** message size, in-register workload (do_work)
- Generally **no intranode async** supported!

- **Internode results for Westmere cluster (QDR-IB)**



- **Internode results for Cray XT4 and XE6**



MPI nonblocking – results and consequences

- **Asynchronous nonblocking MPI does not work in general for large messages**

- **Consequences**
  - If we need async, check if it works
  - If it doesn't, perform comm/calc overlap manually

- **Comm/calc overlap: Options with MPI and MPI/OpenMP**
  - Nonblocking MPI
  - Sacrifice one thread for communication
    - Compute performance impact?
    - Where/how to run? Threads vs. processes?
    - Can SMT be of any use?

- **Case study: Sparse matrix-vector multiply (spMVM)**

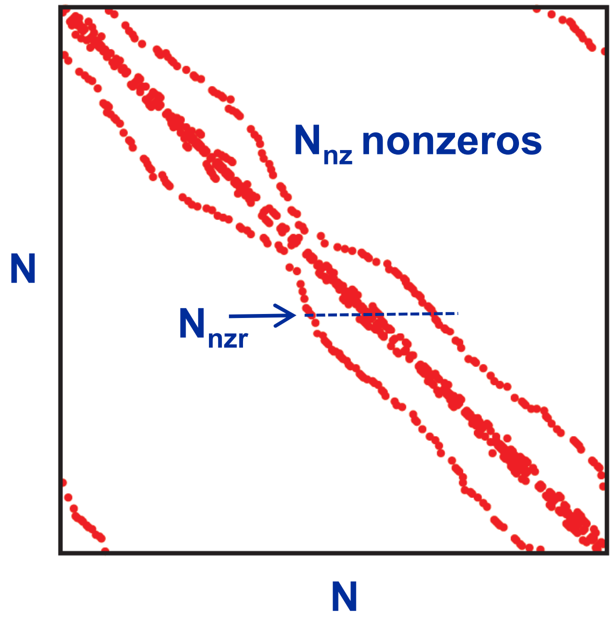
- **Why spMVM?**
  → Dominant operation in many algorithms/applications

- **Physics applications:**
  - Ground state phase diagram Holstein-Hubbard model
  - Physics at the Dirac point in Graphene
  - Anderson localization in disordered systems
  - Quantum dynamics on percolative lattices
- **Algorithms:**
  - Lanczos – extremal eigenvalues
  - JADA – degenerate & inner eigenvalues
  - KPM – spectral properties
  - Chebyshev time evolution

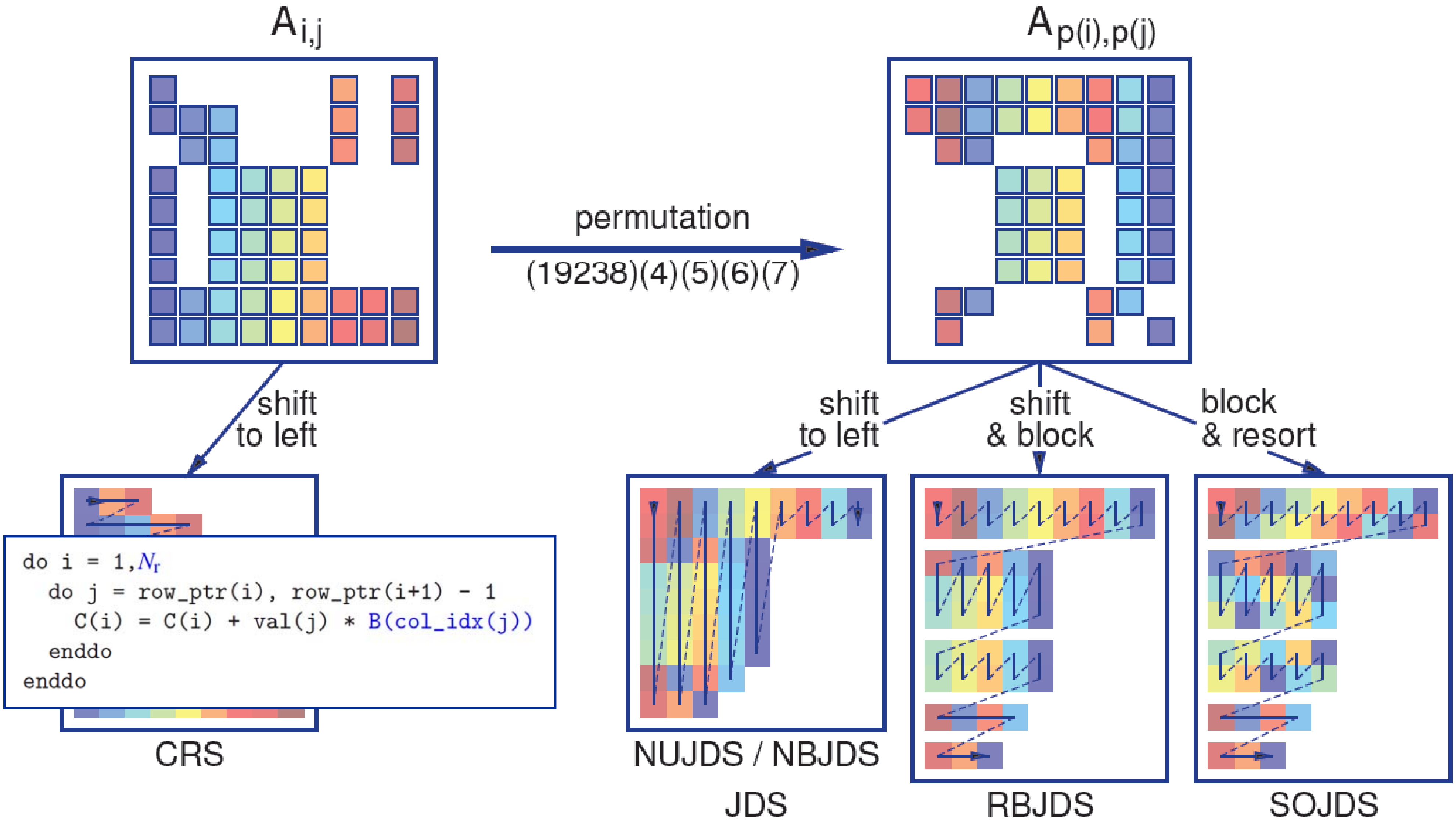
- **Fraction of total time spent in SpMVM (all of those): 85 – 99.99%**

Sparse MVM properties

- "**Sparse**" **matrix ≅ N_{nz} grows slower than quadratically with N**
  - N_{nzr} = avg. # nonzeros per row
- **A different sparsity pattern ("fingerprint") for each problem**
  - Even changes with different numbering of DoFs
- **Performance of spMVM c = A·b**
  - Always memory-bound for large N (see later)
  - Usage of memory BW divided between nonzeros and RHS/LHS vectors
  - Sparsity pattern has strong impact
  - Storage format, too
- **Storage formats**
  - Compressed Row Storage (CRS): Best for modern cache-based µP
  - Jagged Diagonals Storage (JDS): Best for vector(-like) architectures
  - Special formats exploit specific matrix properties



A quick glance on CRS and JDS variants…

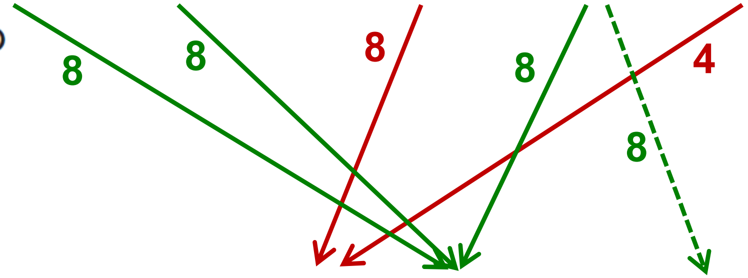



```
do i = 1,Nr
  do j = row_ptr(i), row_ptr(i+1) - 1
    C(i) = C(i) + val(j) * B(col_idx(j))
  enddo
enddo
```


G. Schubert, G. Hager and H. Fehske: *Performance limitations for sparse matrix-vector multiplications on current multicore environments.* In: S. Wagner et al., High Performance Computing in Science and Engineering, Garching/Munich 2009. Springer, ISBN 978-3642138713 (2010), 13–26. DOI: 10.1007/978-3-642-13872-0_2, Preprint: arXiv:0910.4836.

- **Concentrate on double precision CRS:**

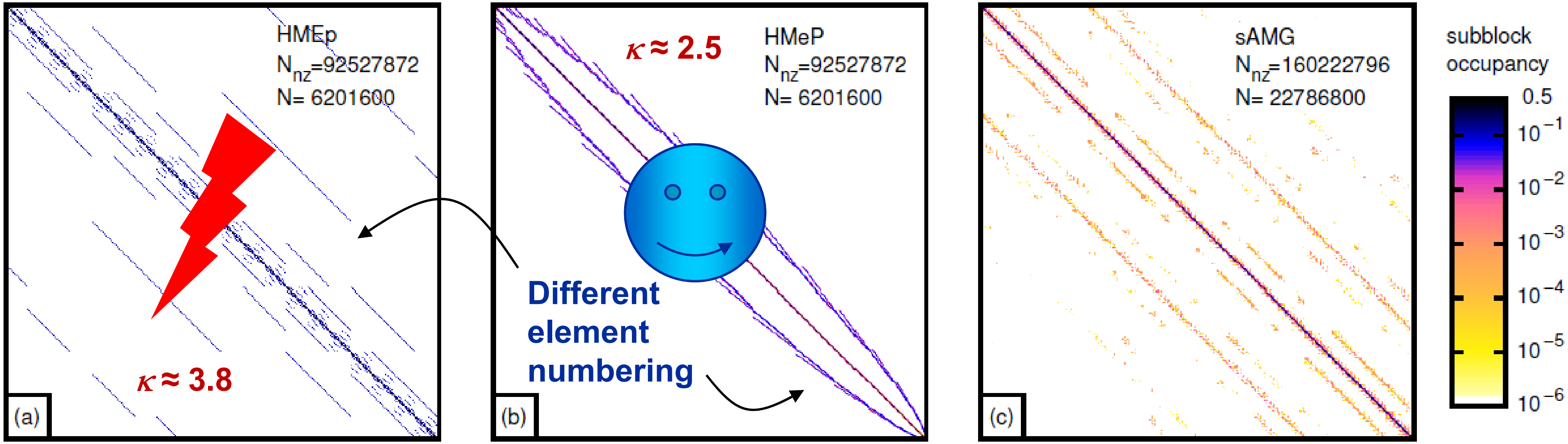
```
do i = 1,Nr
  do j = row_ptr(i), row_ptr(i+1) - 1
    C(i) = C(i) + val(j) * B(col_idx(j))
  enddo
enddo
```



- **DP CRS code balance**
  - κ quantifies extra traffic for loading RHS more than once
  - Predicted Performance = streamBW/B_{CRS}
  - Determine κ by measuring performance and actual memory BW

$$B_{\mathrm{CRS}} = \left(\frac{12+24/N_{\mathrm{nzr}}+\kappa}{2}\right)\frac{\mathrm{bytes}}{\mathrm{flop}} = \left(6+\frac{12}{N_{\mathrm{nzr}}}+\frac{\kappa}{2}\right)\frac{\mathrm{bytes}}{\mathrm{flop}} .$$

- **Matrices in our test cases: $N_{nzr} \approx 7 \ldots 15$ → RHS and LHS do matter!**
  - HM: Hostein-Hubbard Model, 6-site lattice, 6 electrons, 15 phonons, $N_{nzr} \approx 15$
  - sAMG: Adaptive Multigrid method, irregular discretization of Poisson stencil on car geometry, $N_{nzr} \approx 7$
  - Considered Reverse Cuthill-McKee (RCM) transformation, but no gain

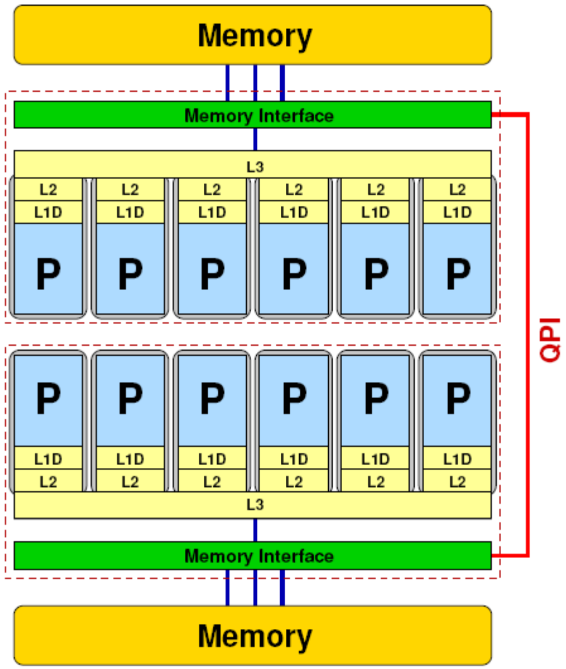


Test matrices: Sparsity patterns

- **Analysis for HMeP matrix on Nehalem EP socket**
  - BW used by spMVM kernel = 18.1 GB/s → should get ≈ 2.66 Gflop/s spMVM performance
  - Measured spMVM performance = 2.25 Gflop/s
  - Solve 2.25 Gflop/s = BW/B_{CRS} for $\kappa \approx 2.5$

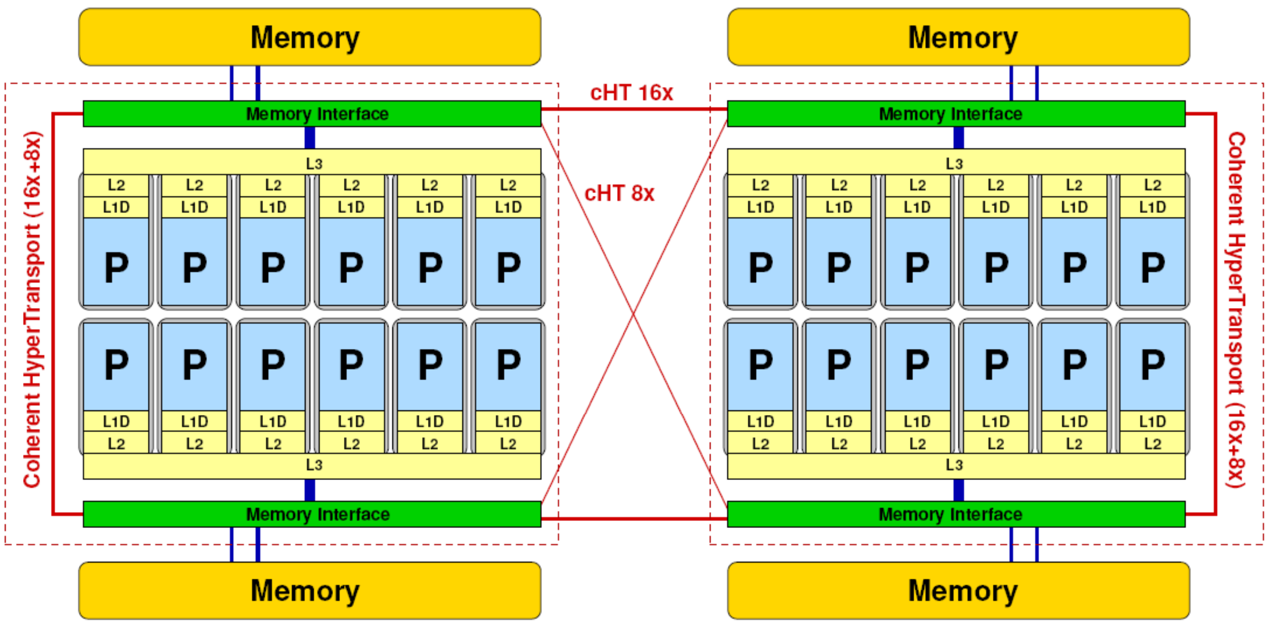
    → 37.5 extra bytes per row
    → RHS is loaded 6 times from memory
    → about 33% of BW goes into RHS

- **Special formats that exploit features of the sparsity pattern are not considered here**
  - Symmetry
  - Dense blocks
  - Subdiagonals (possibly w/ constant entries)

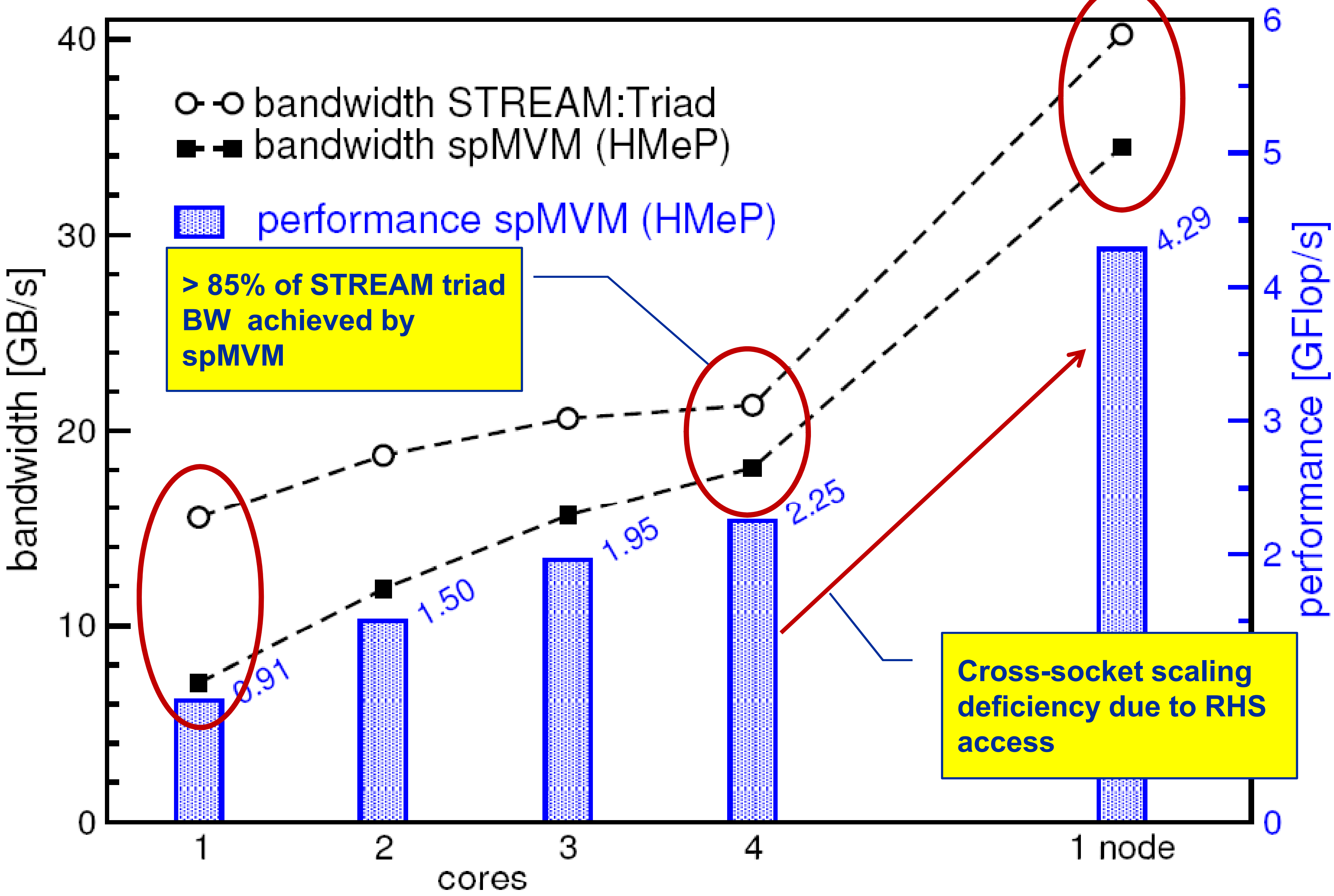


- **Intel Westmere EP (Xeon 5650)**
- **STREAM triad BW (NT stores suppressed, counting write-allocate transfers): 20.6 GB/s per domain**
- **QDR InfiniBand fully nonblocking fat-tree interconnect**

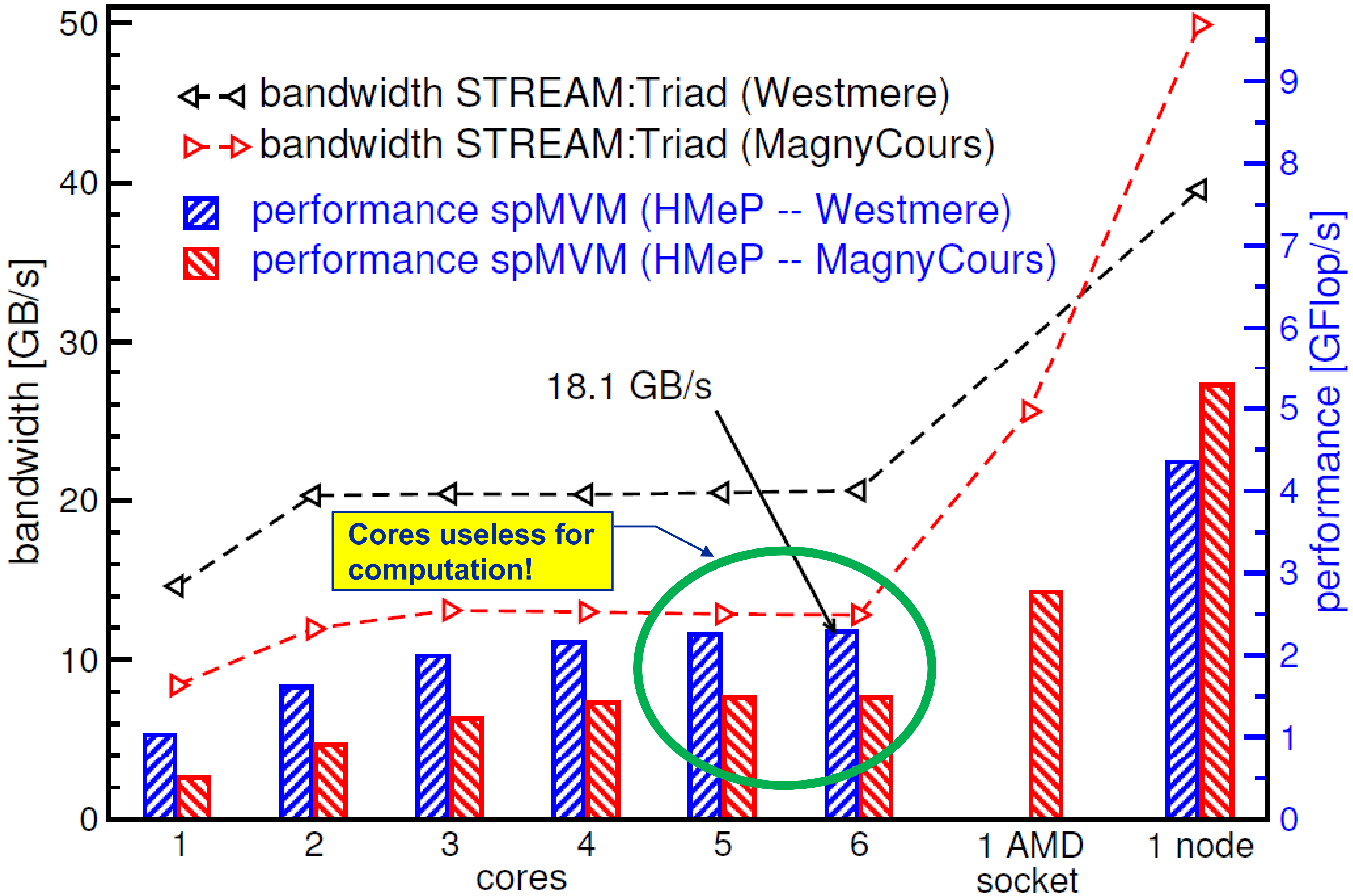
- **AMD Magny Cours (Opteron 6172)**
- **STREAM triad BW: 12.8 GB/s per domain**
- **Cray Gemini interconnect**



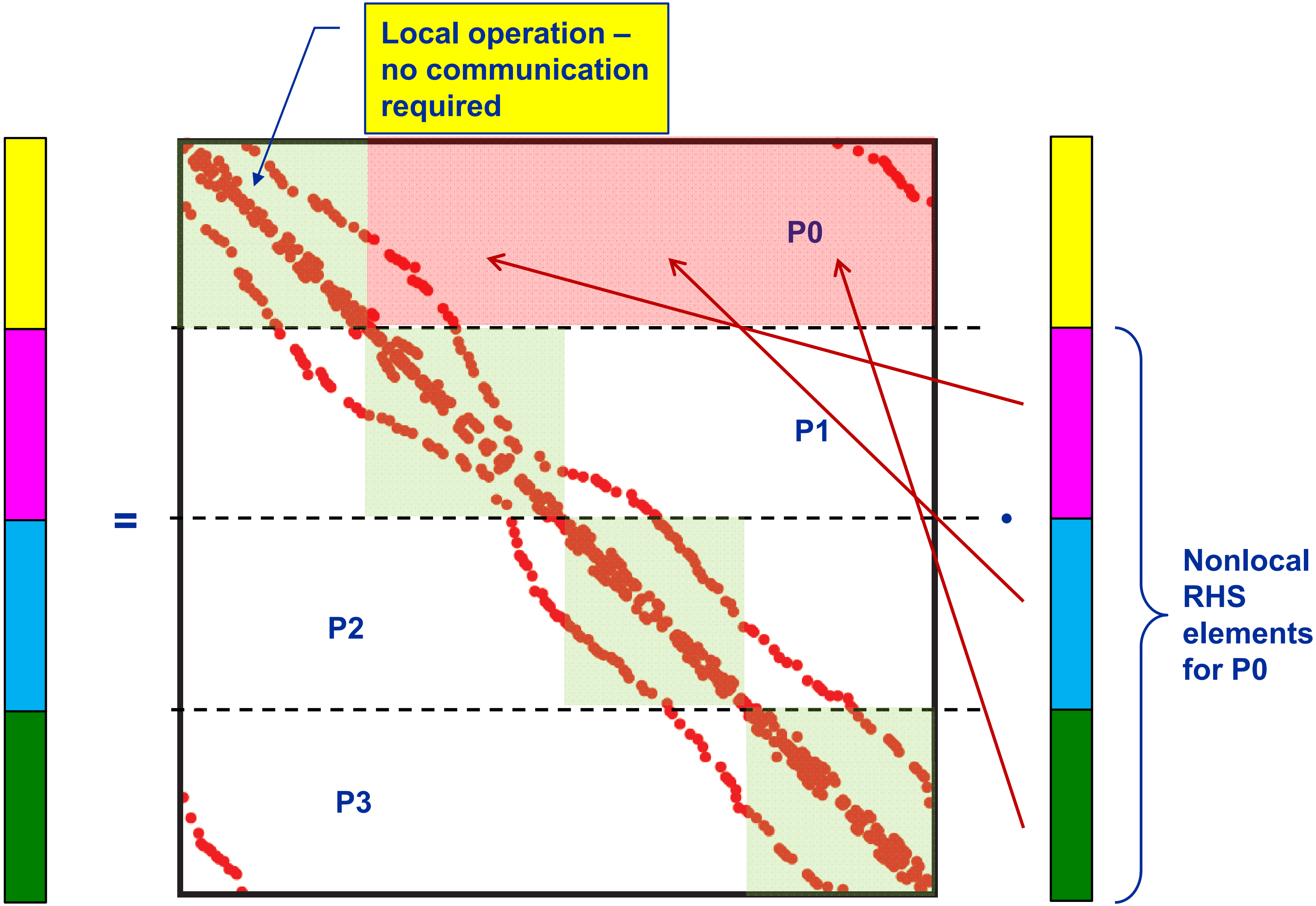
Node-level performance for HMeP: Nehalem EP (Xeon 5550)



Node-level performance for HMeP: Westmere EP (Xeon 5650) vs. Cray XE6 Magny Cours (Opteron 6172)

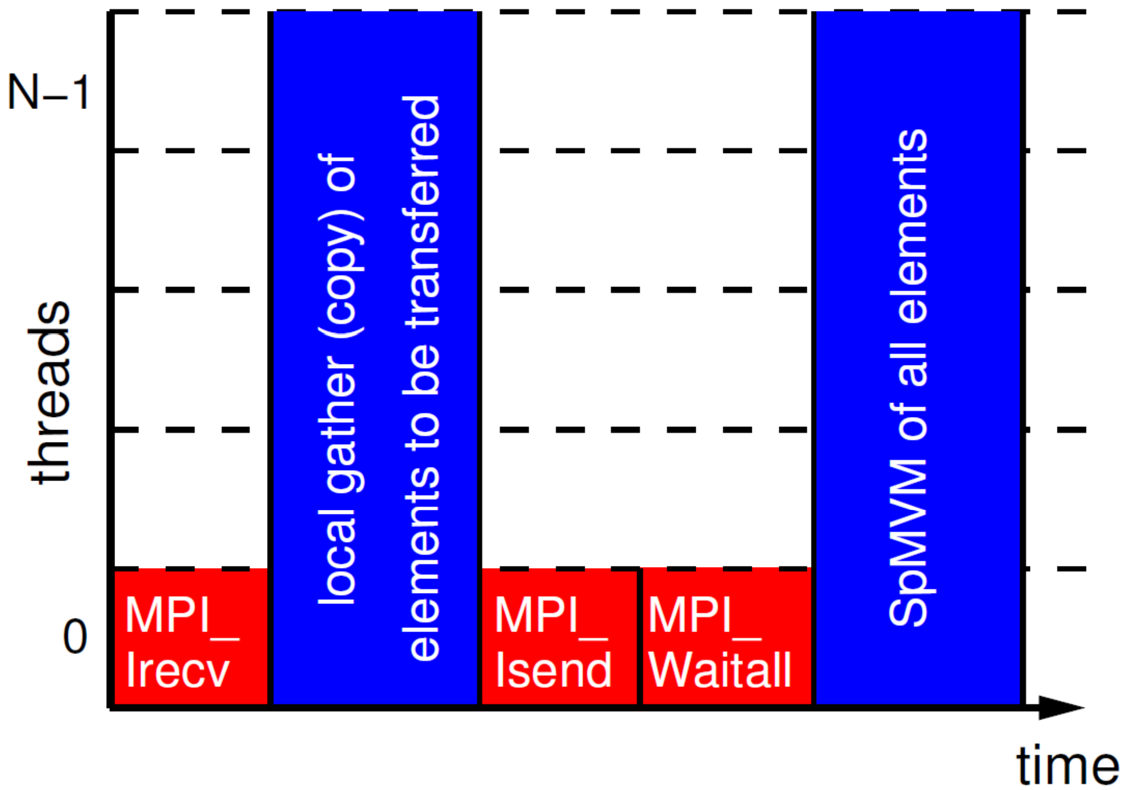


Distributed-memory parallelization of spMVM



- **Variant 1: "Vector mode" without overlap**

- **Standard concept for "hybrid MPI+OpenMP"**
- **Multithreaded computation (all threads)**
- **Communication only outside of computation**



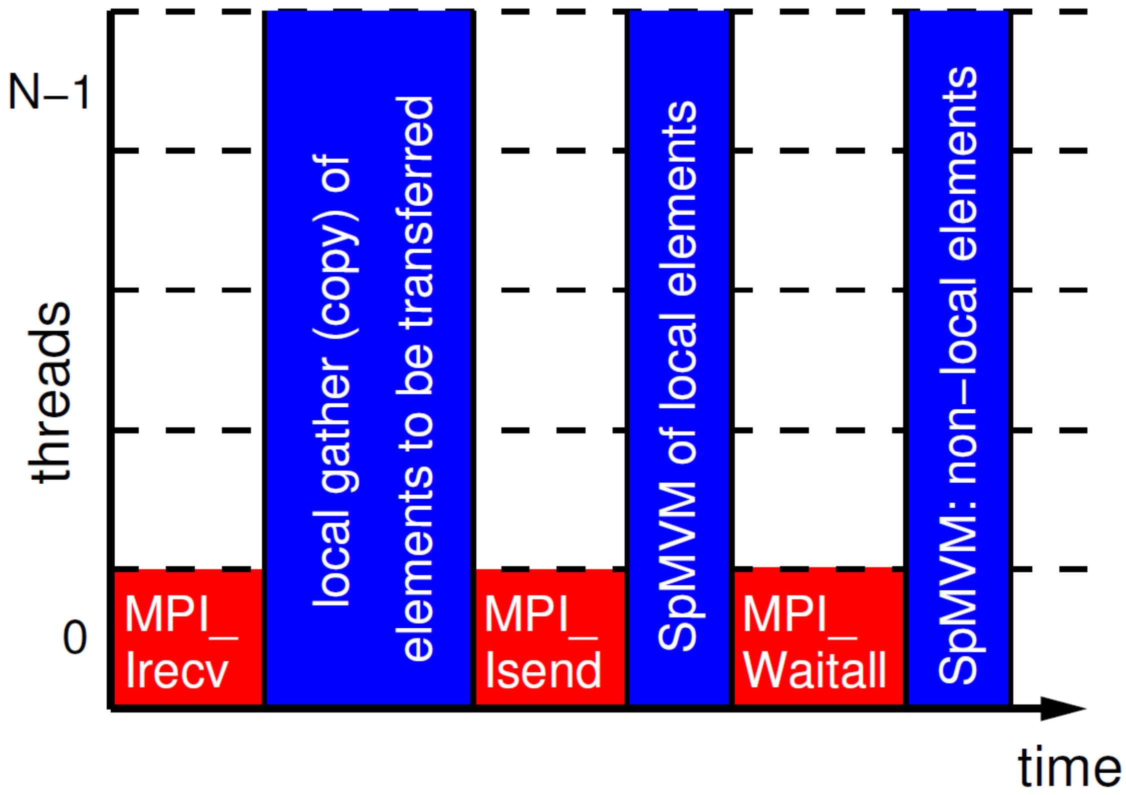
- **Benefit of threaded MPI process only due to message aggregation and (probably) better load balancing**

G. Hager, G. Jost, and R. Rabenseifner: *Communication Characteristics and Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-core SMP Nodes.* In: Proceedings of the Cray Users Group Conference 2009 (CUG 2009), Atlanta, GA, USA, May 4-7, 2009. PDF

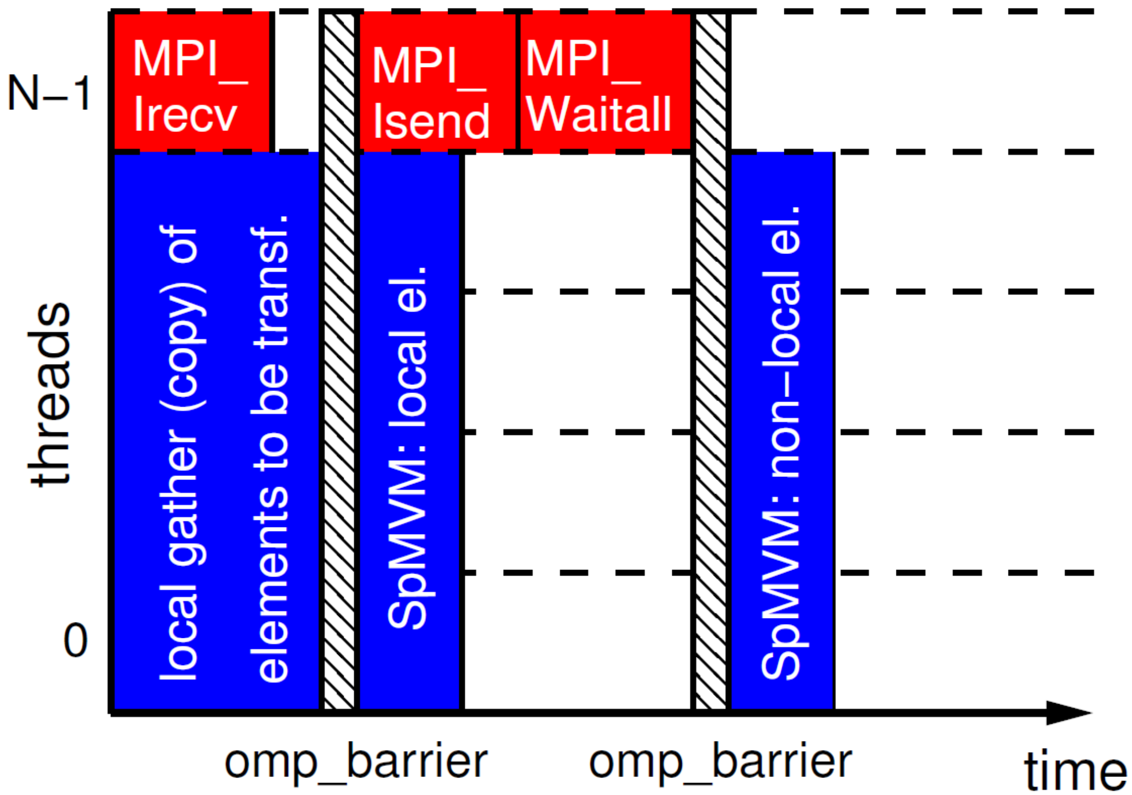
- Variant 2: "Vector mode" with naïve overlap ("good faith hybrid")

- Relies on MPI to support async nonblocking PtP
- Multithreaded computation (all threads)

- Still simple programming
- Drawback: Result vector is written twice to memory
  - modified performance model



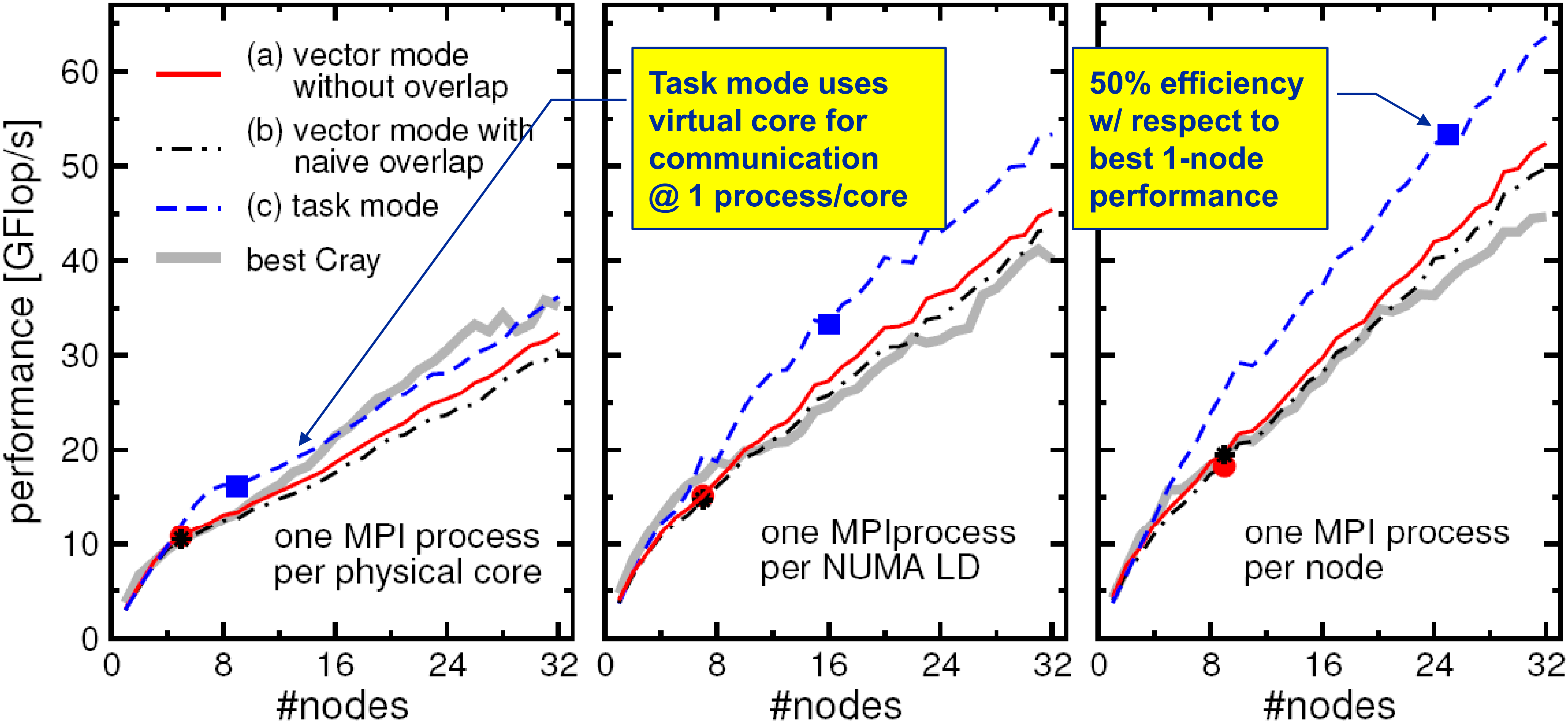
- **Variant 3: "Task mode" with dedicated communication thread**
- **Explicit overlap, more complex to implement**
- **One thread missing in team of compute threads**
  - But that doesn't hurt here…
  - Using tasking seems simpler but may require some work on NUMA locality
- **Drawbacks**
  - Result vector is written twice to memory
  - No simple OpenMP worksharing (manual, tasking)



R. Rabenseifner and G. Wellein: *Communication and Optimization Aspects of Parallel Programming Models on Hybrid Architectures.* International Journal of High Performance Computing Applications **17**, 49-62, February 2003. DOI:10.1177/1094342003017001005

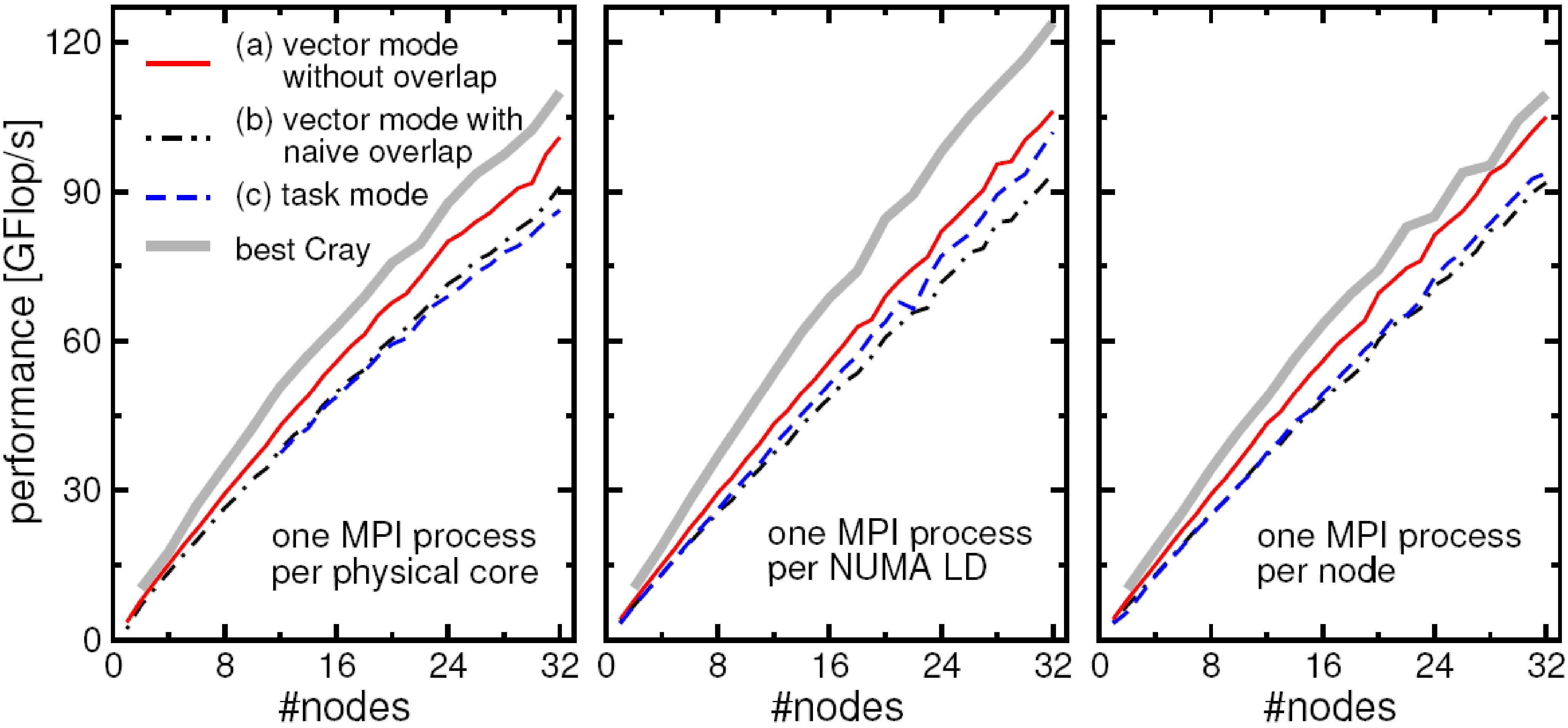
M. Wittmann and G. Hager: *Optimizing ccNUMA locality for task-parallel execution under OpenMP and TBB on multicore-based systems*. Technical report. Preprint:arXiv:1101.0093

Results HMeP



- Dominated by communication (and some load imbalance for large #procs)
- Single-node Cray performance cannot be maintained beyond a few nodes
- Task mode pays off esp. with one process (12 threads) per node
- Task mode overlap (over-)compensates additional LHS traffic

Results sAMG



- **Much less communication-bound**
- **XE6 outperforms Westmere cluster, can maintain good node performance**
- **Hardly any discernible difference as to # of threads per process**
- **If pure MPI is good enough, don't bother going hybrid!**

Conclusions & outlook

- Do not rely on asynchronous MPI progress
- Sparse MVM leaves resources (cores) free for use by communication threads
- Simple "vector mode" hybrid MPI+OpenMP parallelization is not good enough if communication is a real problem
- "Task mode" hybrid can truly hide communication and overcompensate penalty from additional memory traffic in spMVM
- Comm thread can share a core with comp thread via SMT and still be asynchronous
- If pure MPI scales ok and maintains its node performance according to the node-level performance model, don't bother going hybrid

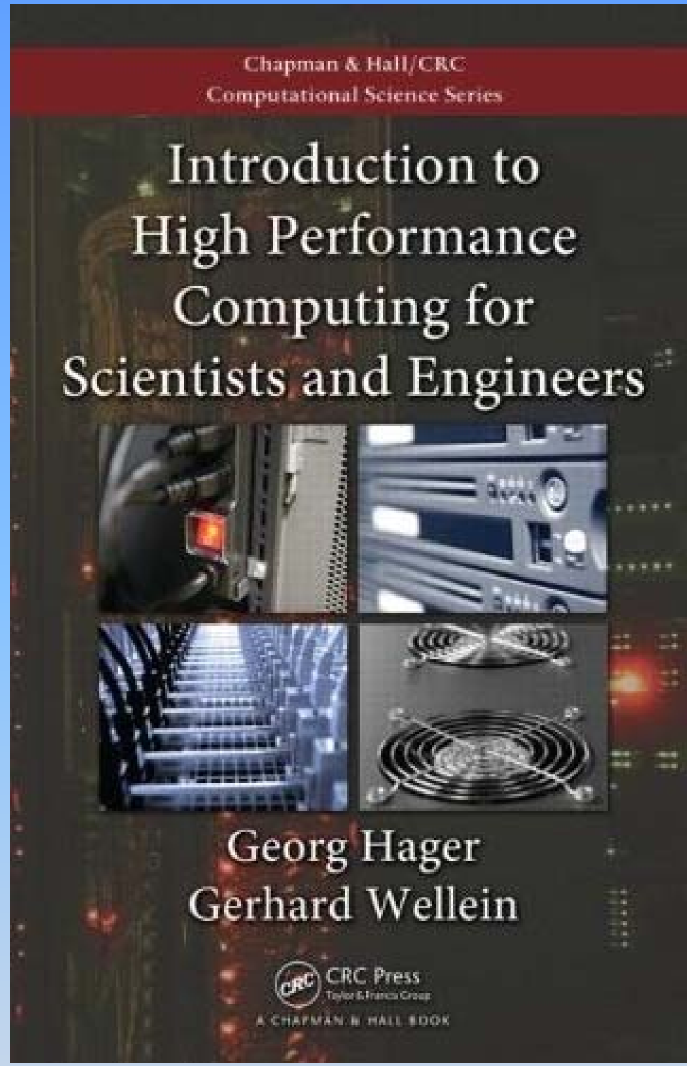
- Work in progress: multi-GPU implementation
  - Overlap even more essential
  - Matrices with small N_{nzr} are a problem (PCIe)

New HPC textbook

Georg Hager and Gerhard Wellein:

Introduction to High Performance Computing for Scientists and Engineers

CRC Press, ISBN 978-1439811924
356 pages
July 2010



"Georg Hager and Gerhard Wellein have developed a very approachable introduction to high performance computing for scientists and engineers. Their style and descriptions are easy to read and follow. … This book presents a balanced treatment of the theory, technology, architecture, and software for modern high performance computers and the use of high performance computing systems. The focus on scientific and engineering problems makes it both educational and unique. I highly recommend this timely book for scientists and engineers. I believe it will benefit many readers and provide a fine reference.“

— From the Foreword by Jack Dongarra, University of Tennessee, Knoxville, USA