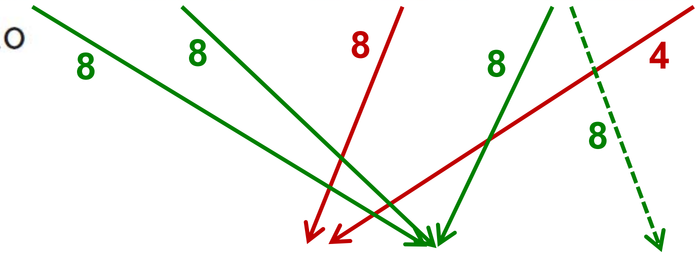
Case study:
Hybrid-parallel sparse MVM (again)

SpMVM node performance model

- **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

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

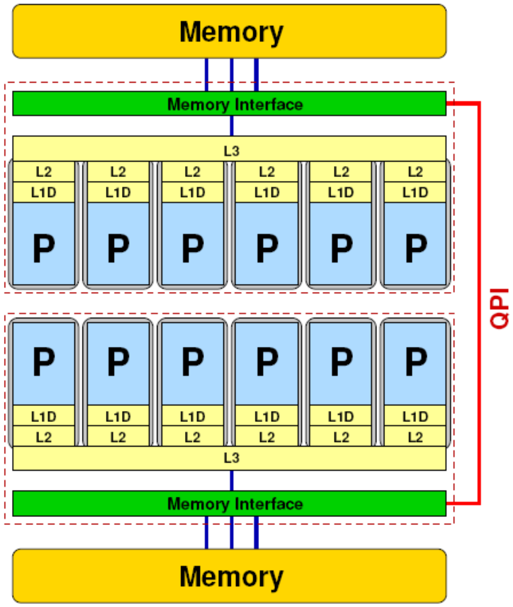
Test matrices: Sparsity patterns

- **Analysis for HMeP matrix on Nehalem EP socket**
  - BW used by spMVM kernel = 18.1 GB/s (as measured by likwid-perfctr)
    → 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

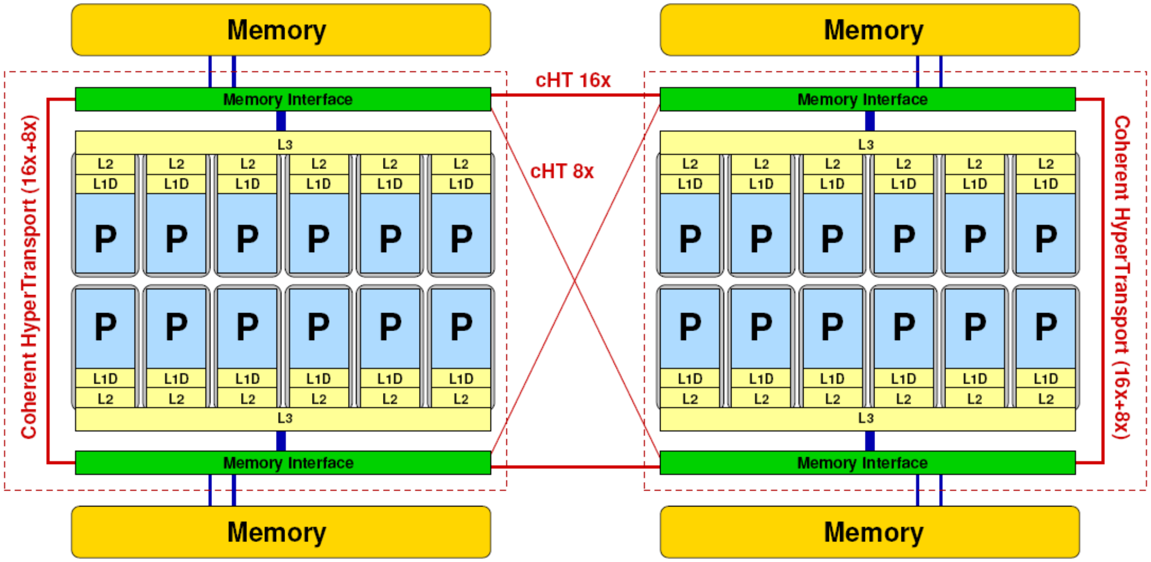
- **Special formats that exploit features of the sparsity pattern are not considered here**
  - Symmetry
  - Dense blocks
  - Subdiagonals (possibly w/ constant entries)

Test systems

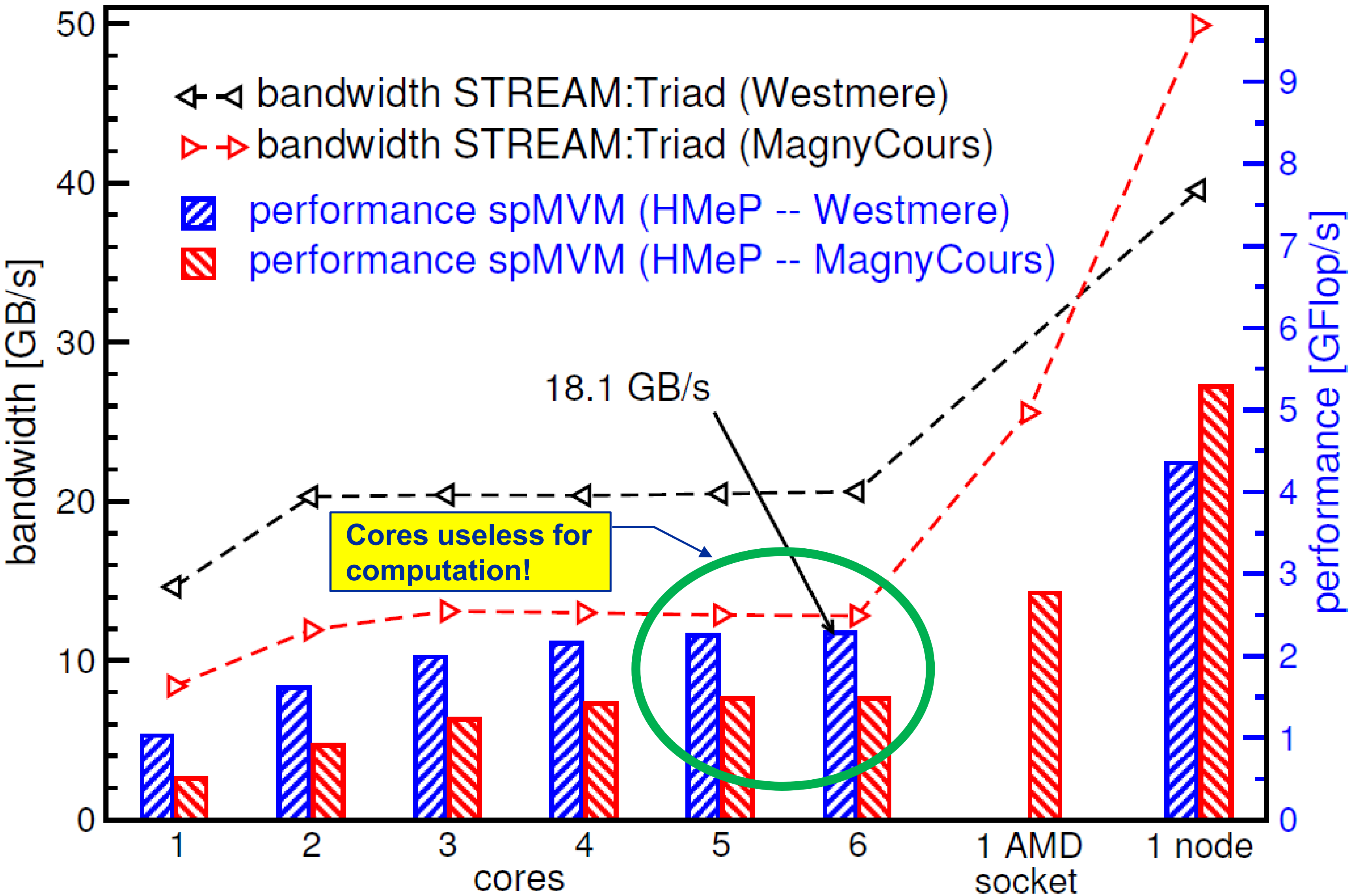


- **Intel Westmere EP (Xeon 5650)**
- **STREAM triad BW (NT stores suppressed, counting write-allocate transfers): 20.6 GB/s per domain**
- **QDR InfiniBand fully nonblocking fat-tree interconnect**

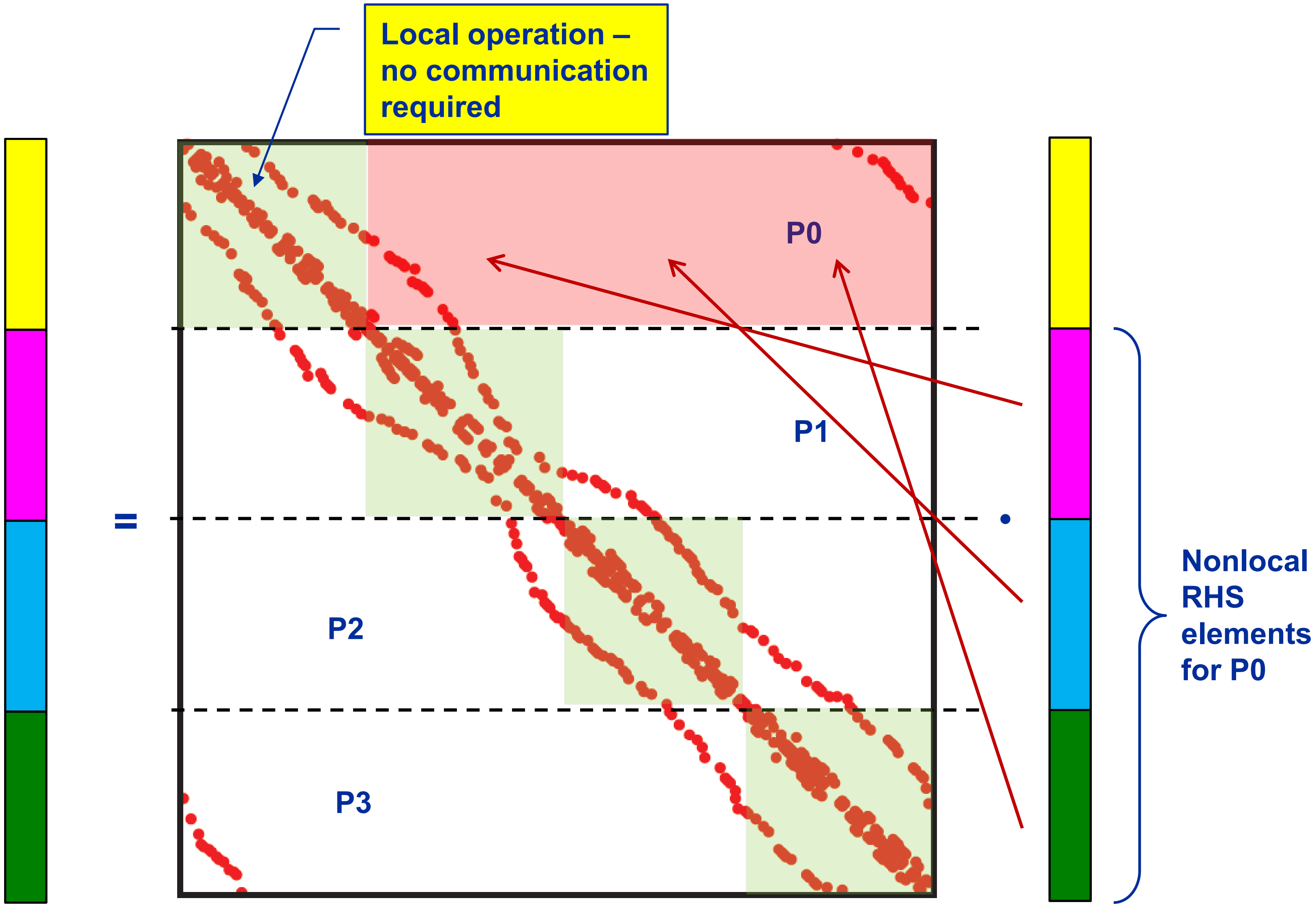
- **AMD Magny Cours (Opteron 6172)**
- **STREAM triad BW: 12.8 GB/s per domain**
- **Cray Gemini interconnect**



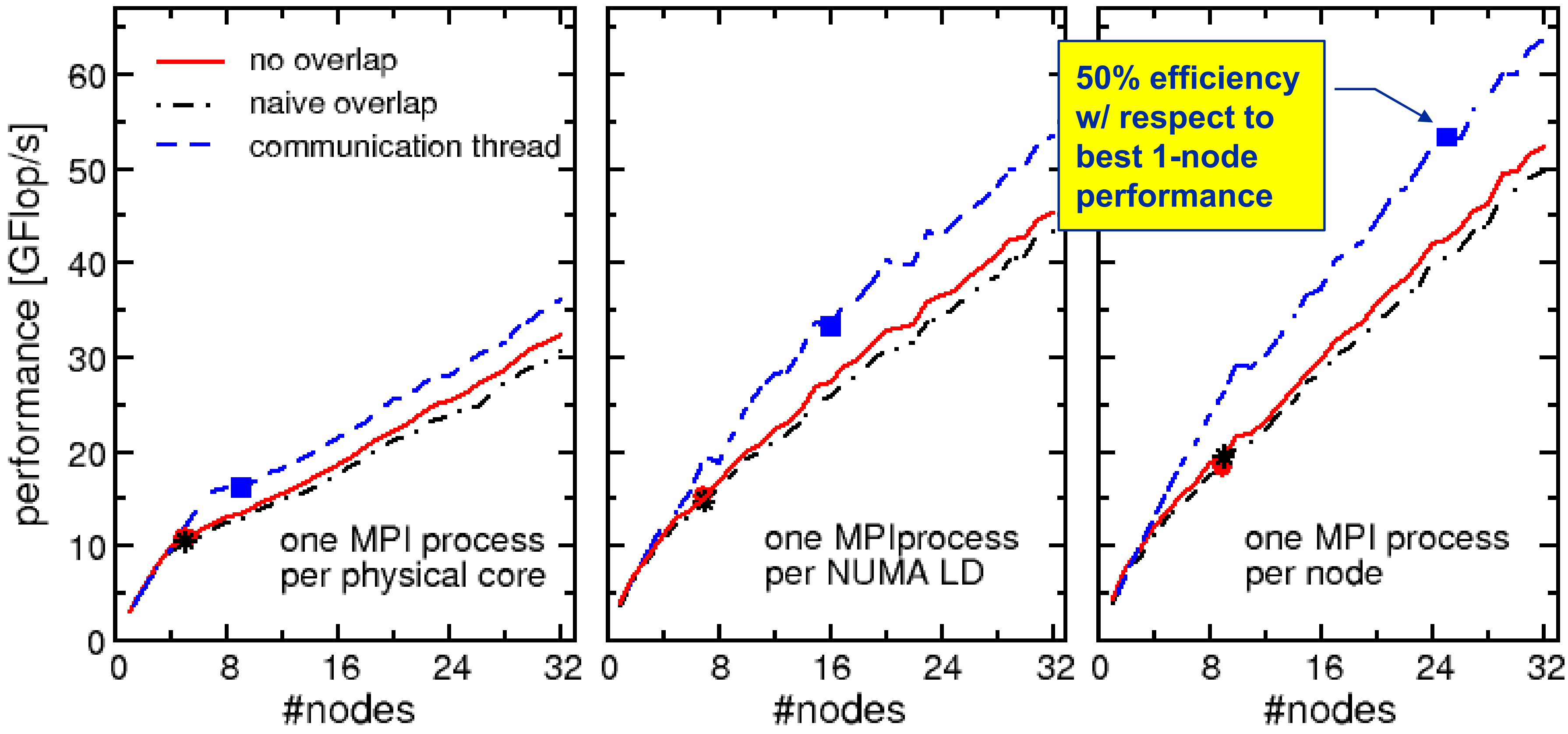
Node-level performance for HMeP: Westmere EP (Xeon 5650) vs. Cray XE6 Magny Cours (Opteron 6172)



Distributed-memory parallelization of spMVM



Results (again)



- Dominated by communication (and some load imbalance for large #processes)
- Comm overlap pays off especially with one process (12 threads) per node
- Communication overlap (over-)compensates additional LHS traffic

Conclusions from the spMVM case

- **We know that**
  - the implementation we have
  - of the algorithm at hand
  - on the machines we use

  makes best use of the relevant node resource (memory bandwidth)

- **How do we know?**
  - Performance measurement (using a stopwatch)
  - Bandwidth measurement (using a simple tool)
  - Along the way we generated some understanding about data transfer properties
- **Then we investigated hybrid MPI/OpenMP programming and**
  - mitigated load imbalance on the node
  - overlapped communication and computation

  to finally shift the 50% efficiency point to larger node counts