1000 x 0 = 0.
Single-node optimization does matter.

Bettina Krammer, UVSQ/ECR, bettina.krammer@uvsq.fr
Georg Hager, Jan Treibig, Gerhard Wellein, RRZE, {firstname.lastname@rrze.uni-erlangen.de}
Anthony Scemama, CNRS, scemama@irsamc.ups-tlse.fr

Questions

- What's your background?
  - Application developer
  - Tools developer
  - Benchmarker
  - …

Questions

Who thinks that for large scale parallel algorithms single node issues are irrelevant?

Questions

Who has spent time on single node optimization?

Outline

- Introduction
  - Fooling the masses
  - Parallel for real men
  - Discussion
- How to get good single-node performance
  - Performance model
  - LBM case study
  - Discussion
- Parallel Applications
  - Sparse-matrix
  - Quantum chemistry
  - Conclusions
  - Discussion

How the masses are often fooled – reasons for <u>not</u> caring about node performance

Or: How do we disguise the fact that our code just doesn't cut it?

Speedup vs. performance

Scalability (speedup) and performance are different things.

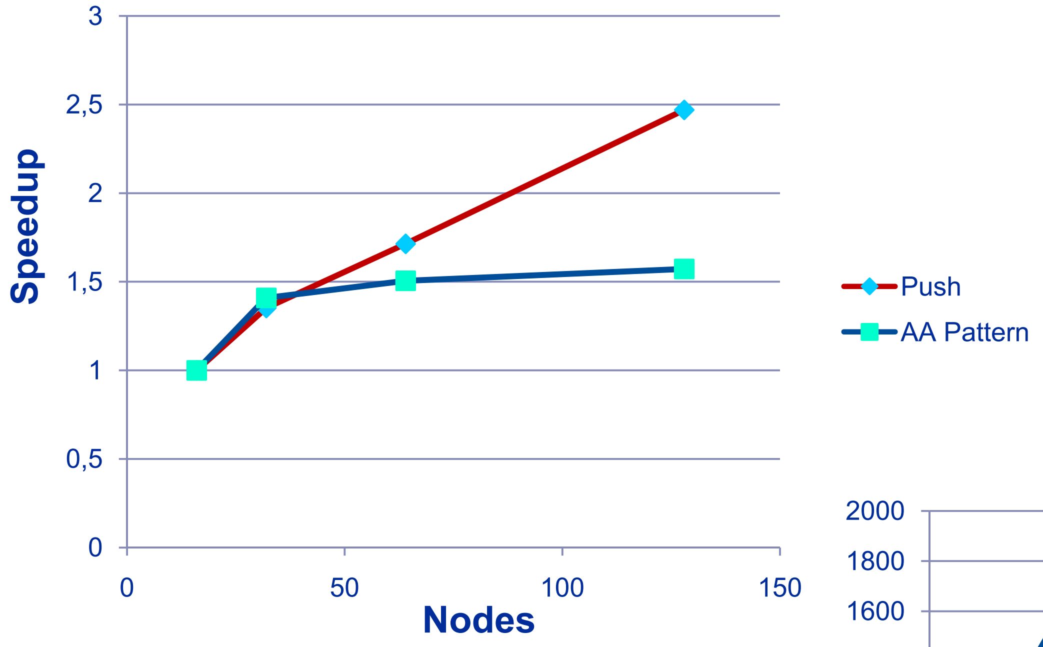
Speedup:

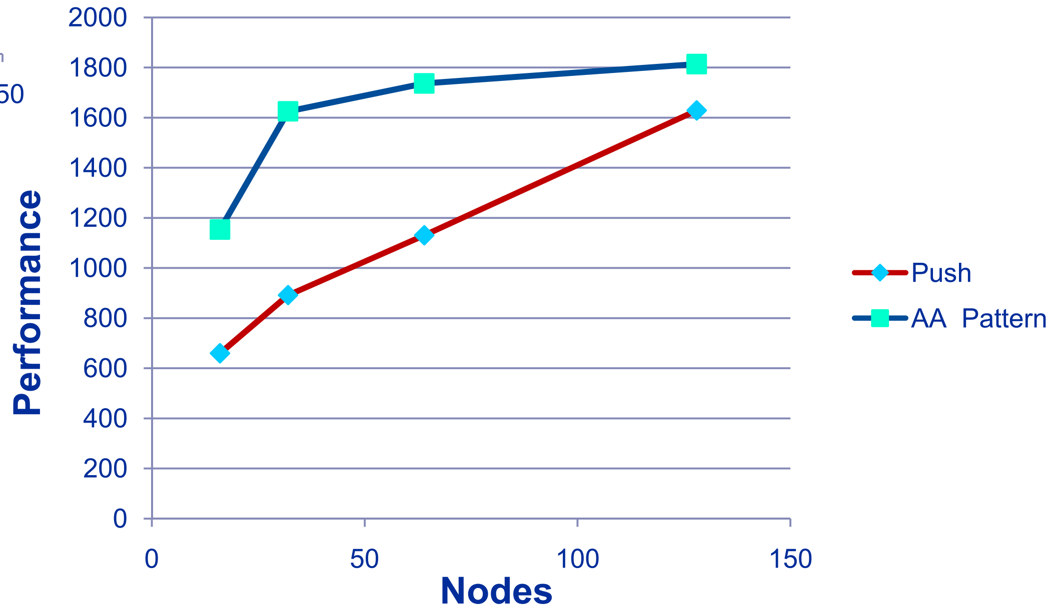
$$S(N) = \frac{\text{work/time with } N \text{ workers}}{\text{work/time with } 1 \text{ worker}}$$

"Good" scalability ↔ $S(N) \approx N$, but there is no mention of how fast you can solve your problem!

Consequence: Reporting speedups can easily conceal the fact that your code has mediocre performance!

Speedup vs. performance: An LBM example





Why is that?

Slowing down code execution gives better speedups.

Let's look at Amdahl's Law with some component that is related to parallel overhead:

Parallel speedup: $$S(N) = \frac{1}{s + \frac{1-s}{N} + c(N)}$$

Slowing down execution by a factor of $\mu>1$:

$$S_\mu(N) = \frac{\mu}{\mu\left(s + (1-s)/N\right) + c(N)} = \frac{1}{s + (1-s)/N + c(N)/\mu}$$

I.e., if there is overhead, the slow code/machine scales better:

$$S_\mu(N) > S_{\mu=1}(N) \quad \text{if} \quad c(N) > 0$$

Slow computing 4 teh win!

Corollaries:

1. Do not use high compiler optimization levels or the latest compiler versions. That will make your speedup graphs look much straighter!

2. If scalability is still bad, parallelize some short loops with OpenMP. That way you can get some extra bonus for a scalable hybrid code.

If someone asks for time to solution, answer that if you had a bigger machine, you could get the solution as fast as you want. This is of course due to the superior scalability of your code.

Parallel computing for real men

Popular belief:

If you take parallel computing seriously, single node performance is not a problem!

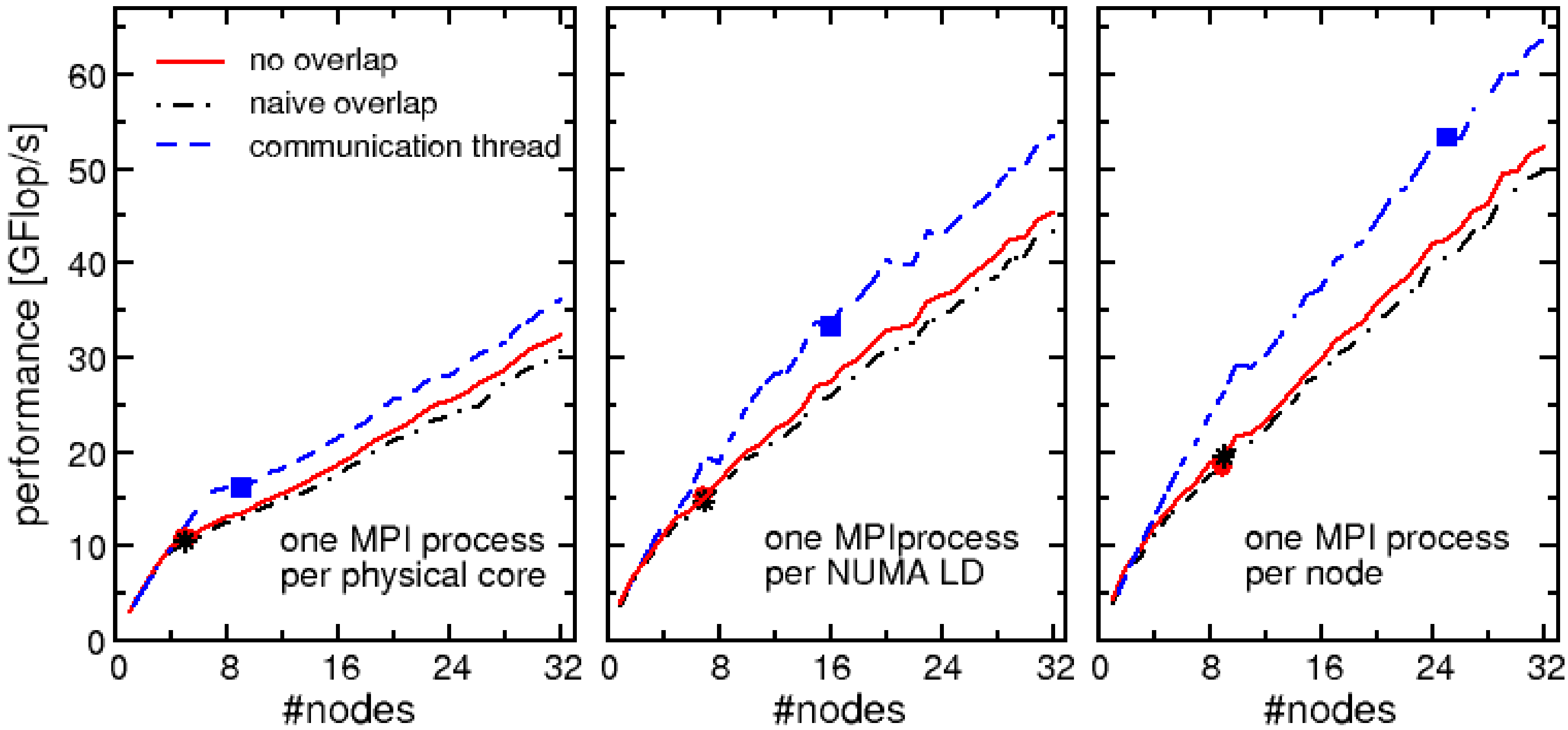
This is because everything is dominated by communication issues.

Parallel Computing for Real Men

- **If this is true something is really going wrong!**
  - If you are communication-bound, parallel efficiency is already unacceptably low

- **Possible reasons:**
  - Load imbalance (→ Parallel Profiling)
  - Low parallel efficiency: Strong scaling, memory consumption

Example: Hybrid-parallel sparse matrix-vector multiply



- 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

Questions

Who is confident that there is no optimization potential left in their single-node code?

Questions

What is an acceptable parallel efficiency?

- 90% ?
- 80% ?
- 70% ?
- 60% ?
- ….
- 20% ?