

On the limits of GPU acceleration

By Vuduc et al
Presented by Eric Christiansen

Goal

What is the boundary between computations that can and cannot be effectively accelerated by GPUs, relative to general-purpose multi-core CPUs within a roughly comparable power footprint?

Goal

Inform future architectures

Case study: problems on the boundary

From the set of classic HPC problems:

- iterative sparse linear solvers
- direct sparse solvers: sparse Cholesky factorization
- generalized N-body problem: fast multipole methods

Limits of approach

- example-based, and only look at three examples
- cannot be sure have reached optimization limits
- only look at existing platforms

Iterative sparse solvers

- eg for sparse A
 - solve $Ax == b$ for x
 - find eigenvalues of A

Iterative sparse solvers

- Conjugate gradients or Lanczos algorithm
 - inner-loop operation is $y = Ax$
 - sparse matrix-vector multiplication (SpMV)

Iterative sparse solvers

(Data already on GPU)

- sparse matrix-vector multiplication (SpMV)
 - bandwidth bottleneck: streaming A
 - GPUs have higher bandwidth to global memory than CPUs
 - 2x speedup for GTX 285 over Nehalem
 - 3.1x bandwidth

Iterative sparse solvers

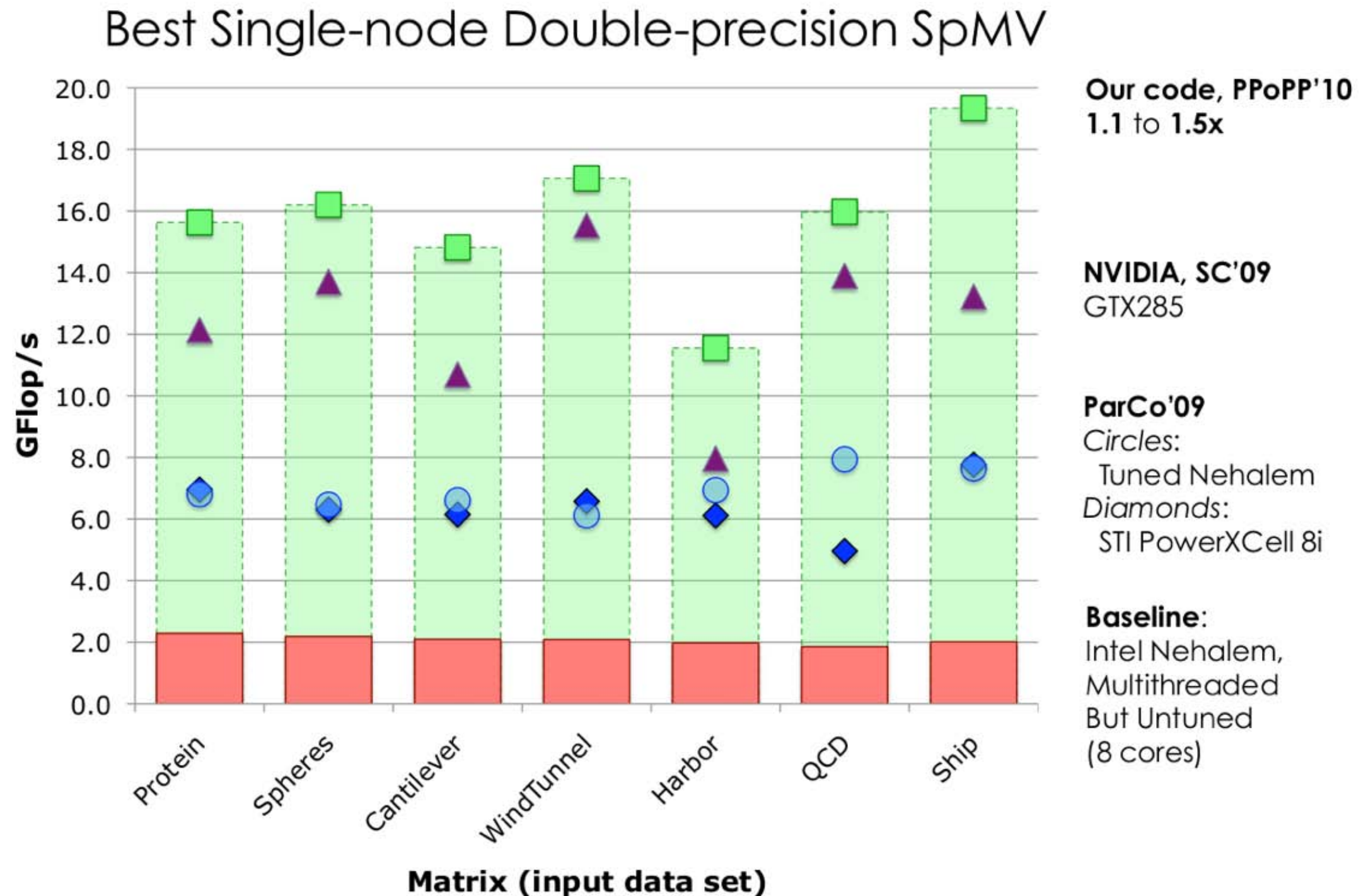
(Data not already on GPU)

$$\begin{aligned}\tau_{\text{cpu}} &\geq \tau_{\text{gpu}} \\ \Rightarrow \frac{k \cdot q}{\beta_{\text{cpu}}} &\geq k \cdot \left(\frac{1}{\beta_{\text{reorg}}} + \frac{1}{\beta_{\text{transfer}}} + \frac{q}{\beta_{\text{gpu}}} \right) \\ \Rightarrow q &\geq \frac{\frac{1}{\beta_{\text{reorg}}} + \frac{1}{\beta_{\text{transfer}}}}{\frac{1}{\beta_{\text{cpu}}} - \frac{1}{\beta_{\text{gpu}}}}\end{aligned}$$

- k : matrix size, q : iterations
- β_{cpu} : CPU bandwidth, β_{reorg} : data reorganization, β_{transfer} : PCIe bandwidth, β_{gpu} : internal GPU bandwidth

Iterative sparse solvers

(Data not already on GPU)



Direct sparse solvers

- eg for sparse A
 - compute Cholesky decomposition of A
(matrix square root)

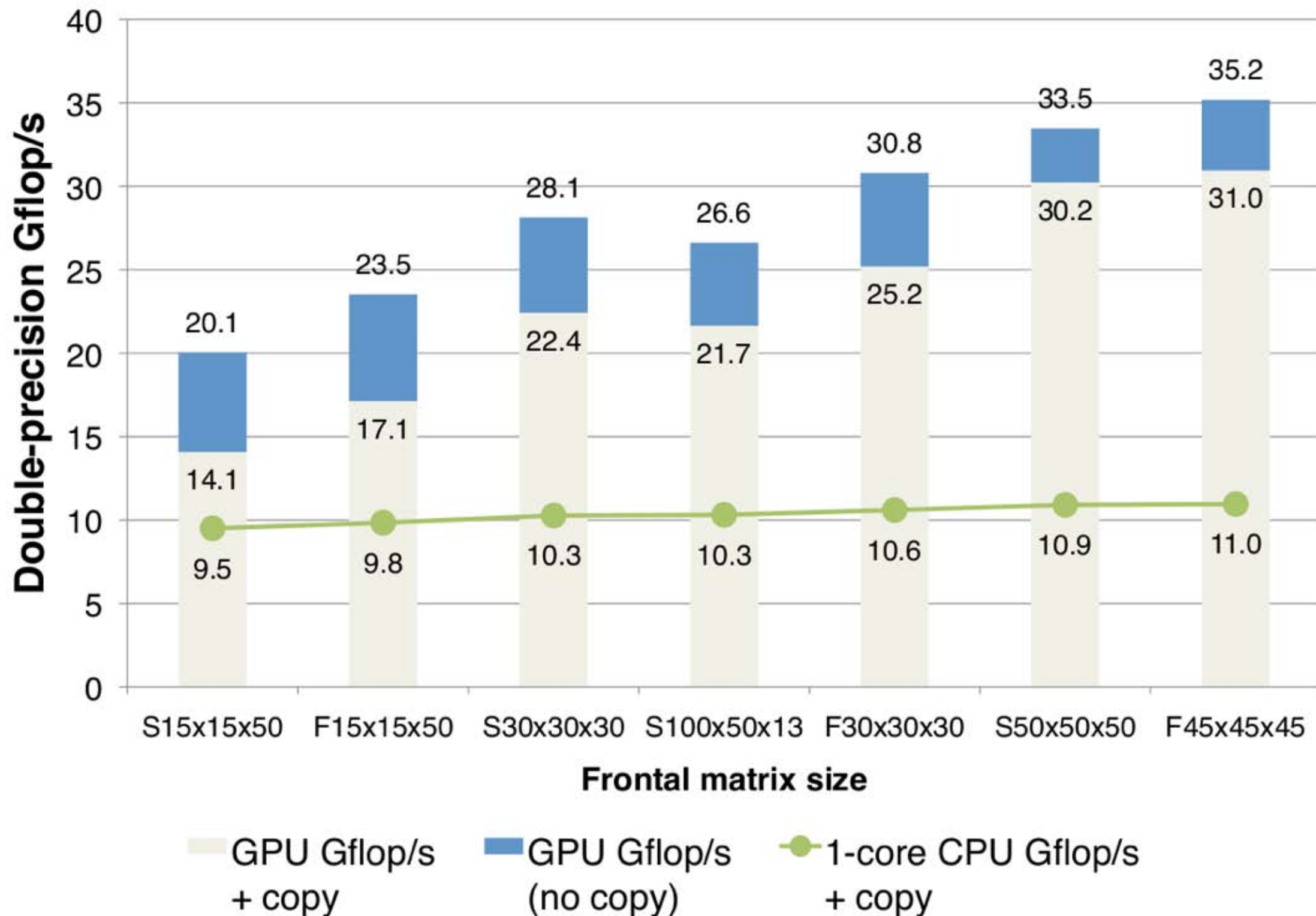
Direct sparse solvers

- multifrontal methods for Cholesky decomposition
- composition of dense matrix subproblems

Direct sparse solvers

- composition of dense matrix subproblems
 - size and number of problems depends on data
 - difficult to tune
 - requires orchestration by CPU (PCIe bandwidth)

Direct sparse solvers



Generalized N-body solvers

- eg kNN or kernel density estimation
- generally: approximate $O(N)$ or $O(N \log N)$ algorithm for $O(N^2)$ problem

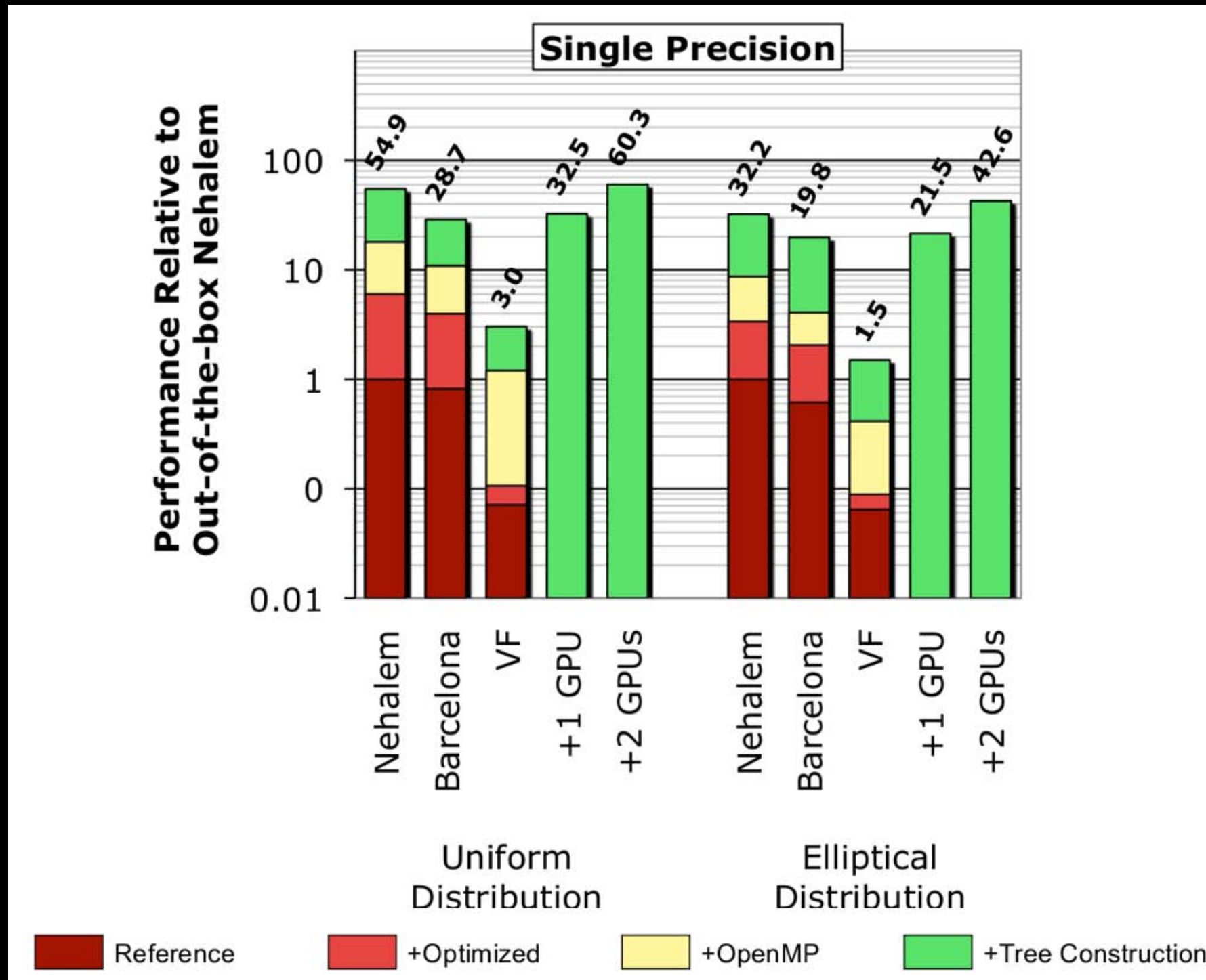
Generalized N-body solvers

- fast multipole method
- composition of parallelizable tree traversals and graph-based neighborhood traversals

Generalized N-body solvers

- composition of parallelizable tree traversals and graph-based neighborhood traversals
- size, shape, and number of problems depends on data
 - difficult to tune
 - requires orchestration by CPU (PCIe bandwidth)

Generalized N-body solvers



Conclusion

- examples of problems at CPU / GPGPU border
- PCIe bandwidth limiting factor