Using Accelerators
The FLOPs, the Bandwidth, and the Latency

Karl Rupp[1,2]

rupp@iue.tuwien.ac.at
http://karlrupp.net/

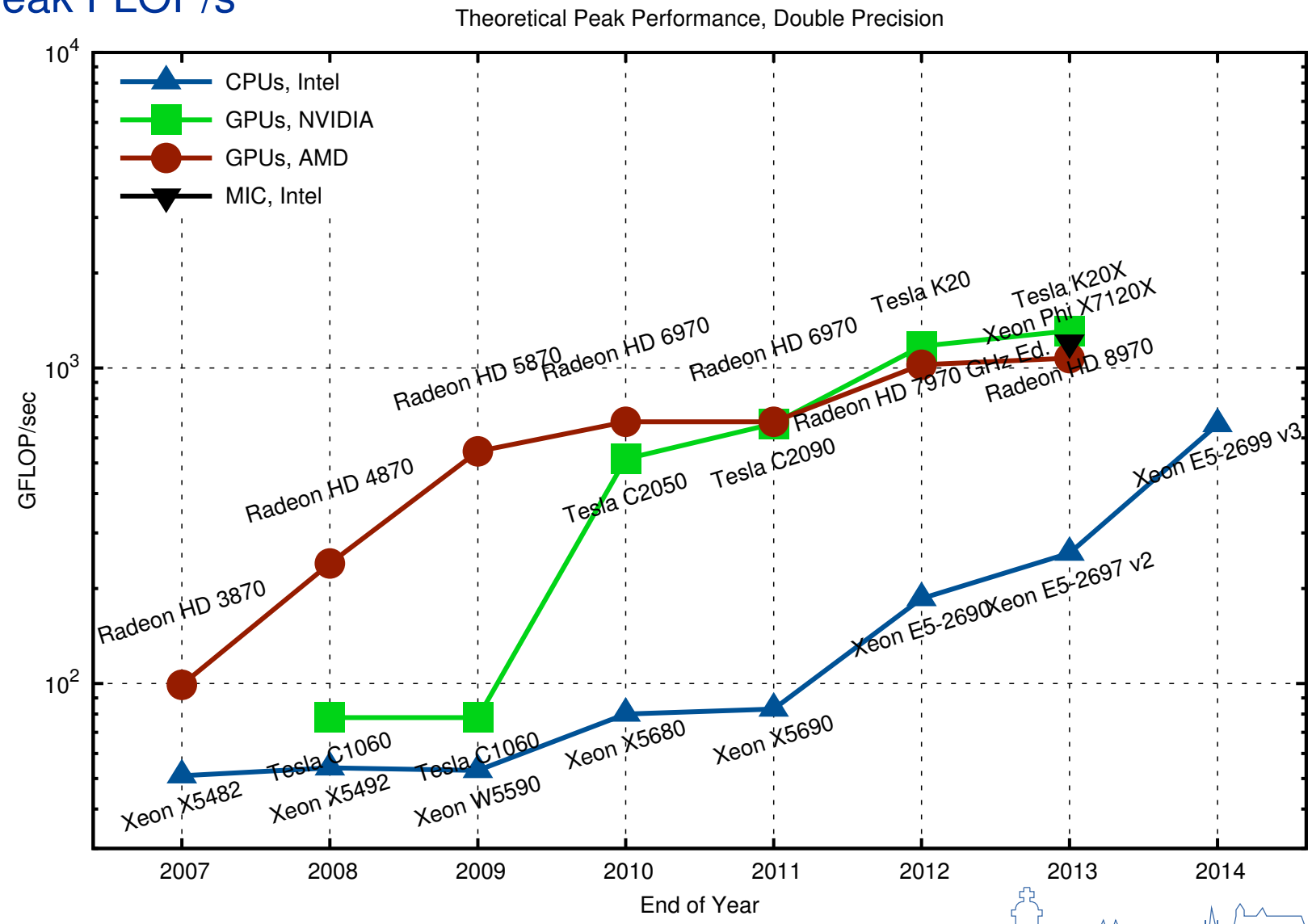
based on stimuli from PETSc+ViennaCL users

[1] Institute for Microelectronics, TU Wien, Austria
[2] Institute for Analysis and Scientific Computing, TU Wien, Austria

October 1st, 2014

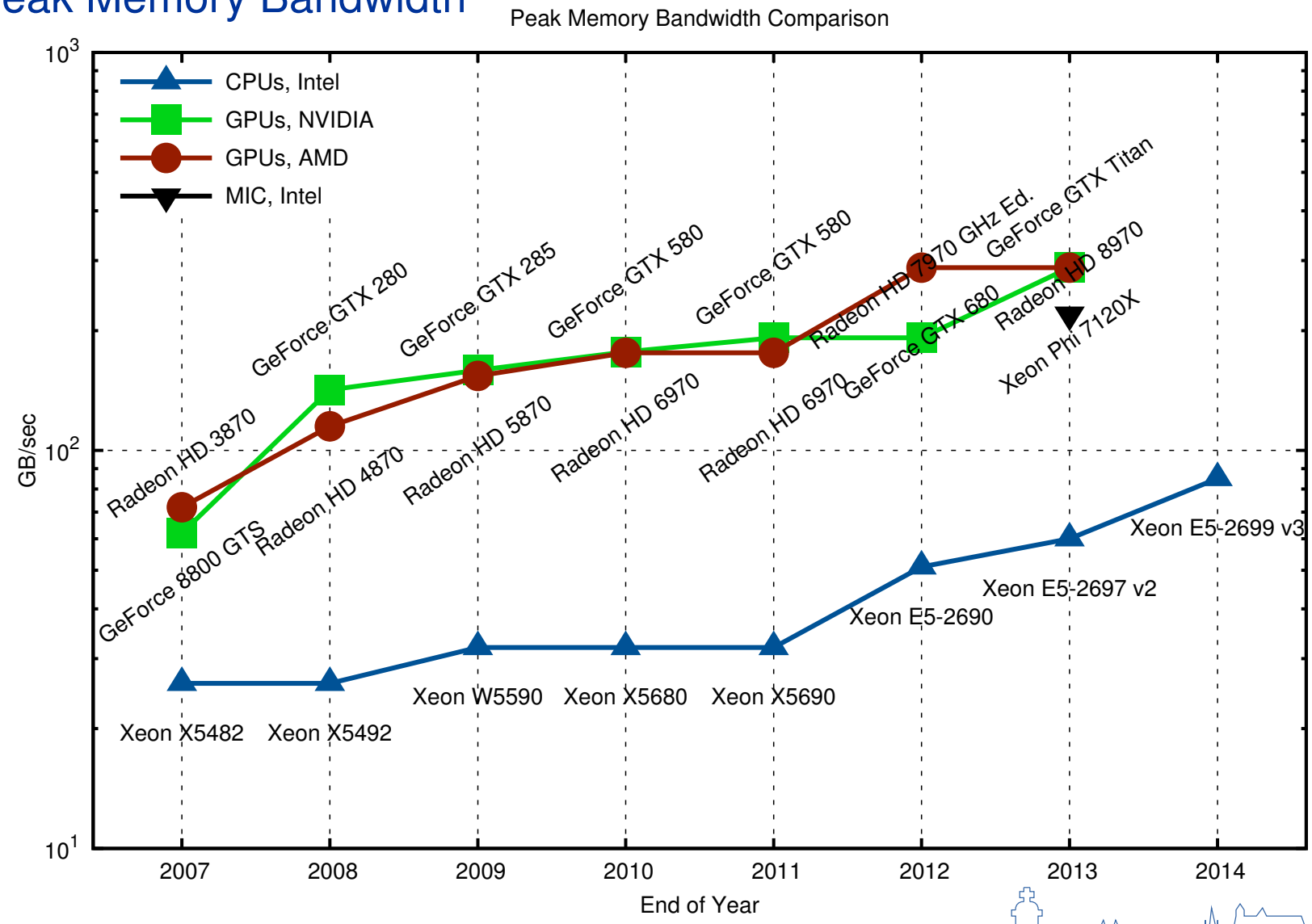
Hardware Specifications

Peak FLOP/s



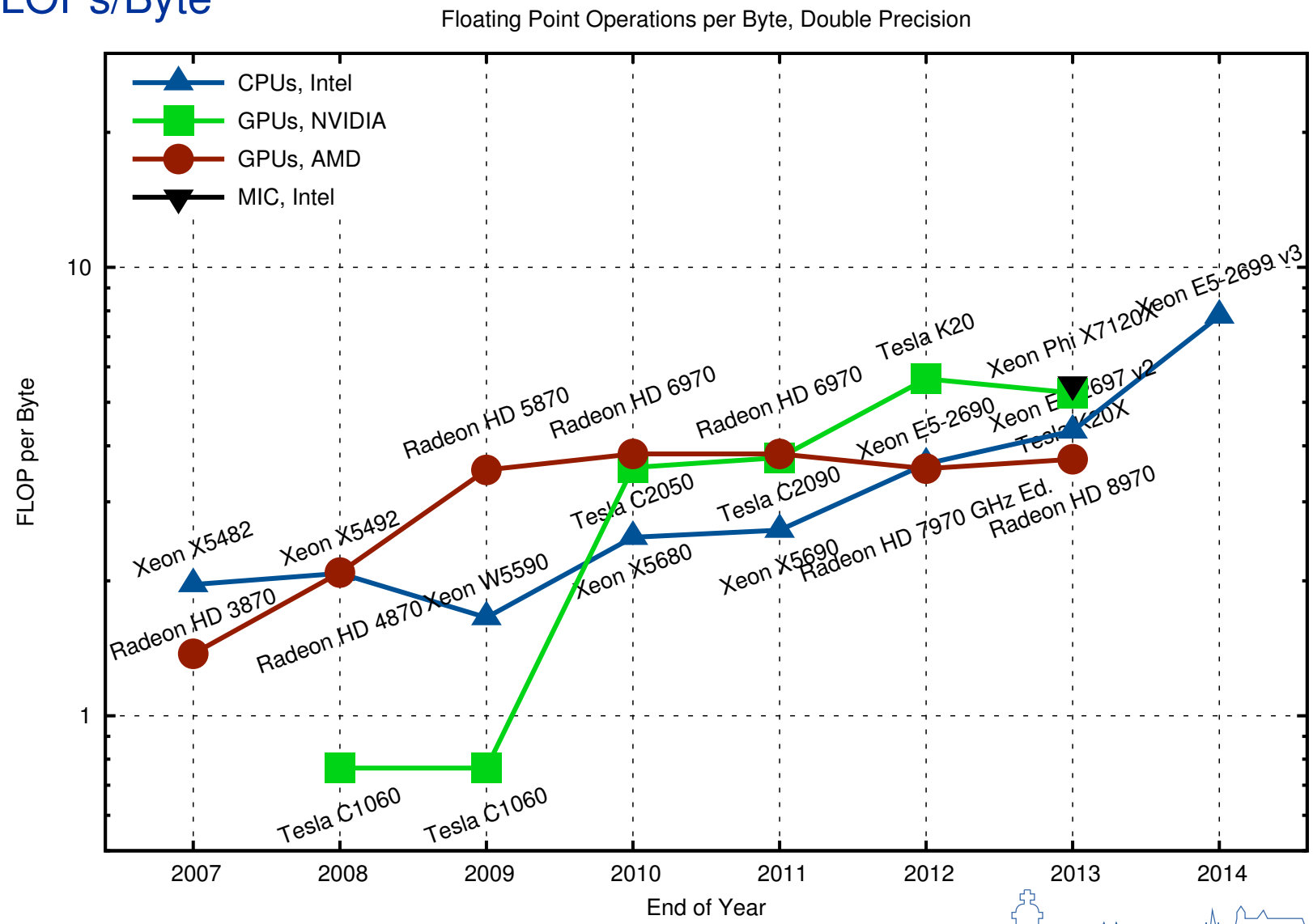
Hardware Specifications

Peak Memory Bandwidth

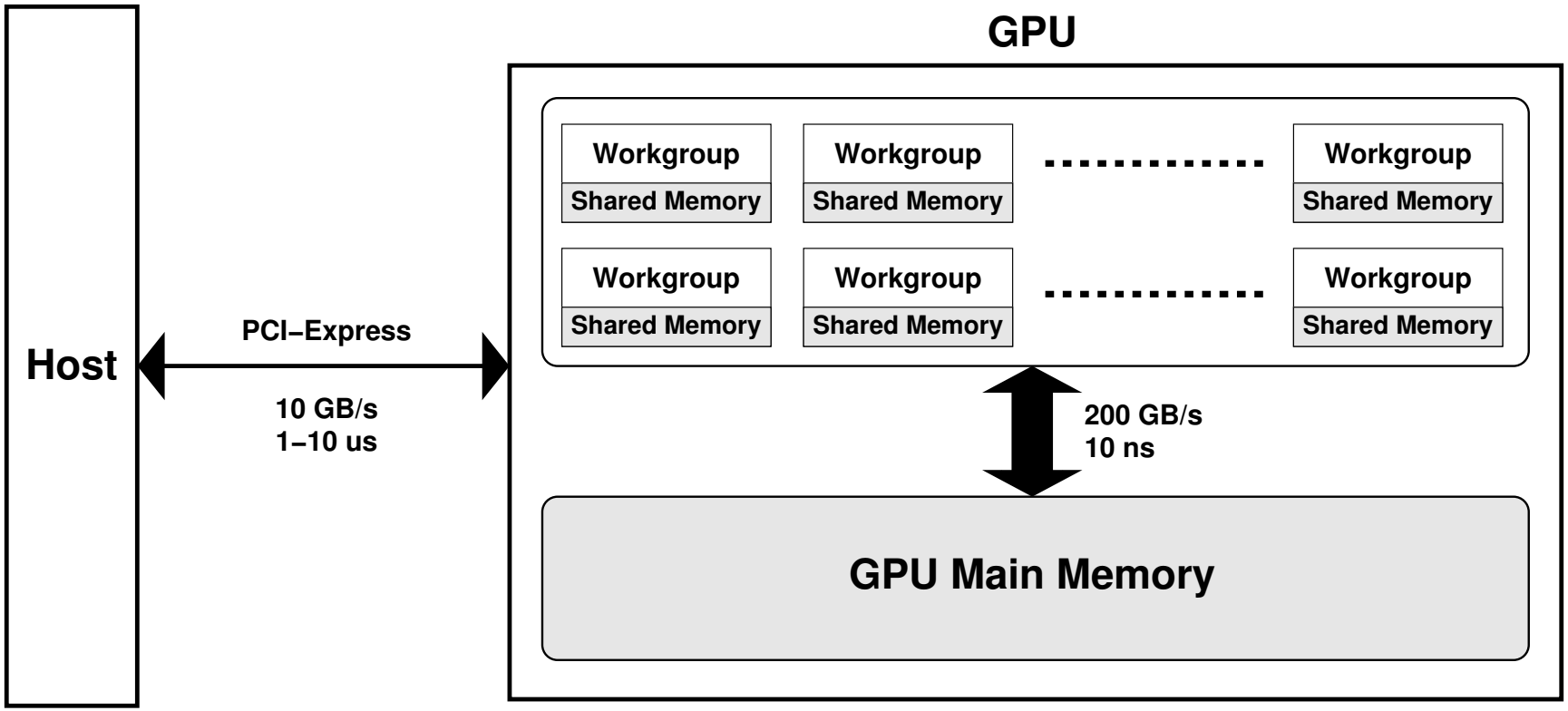


Hardware Specifications

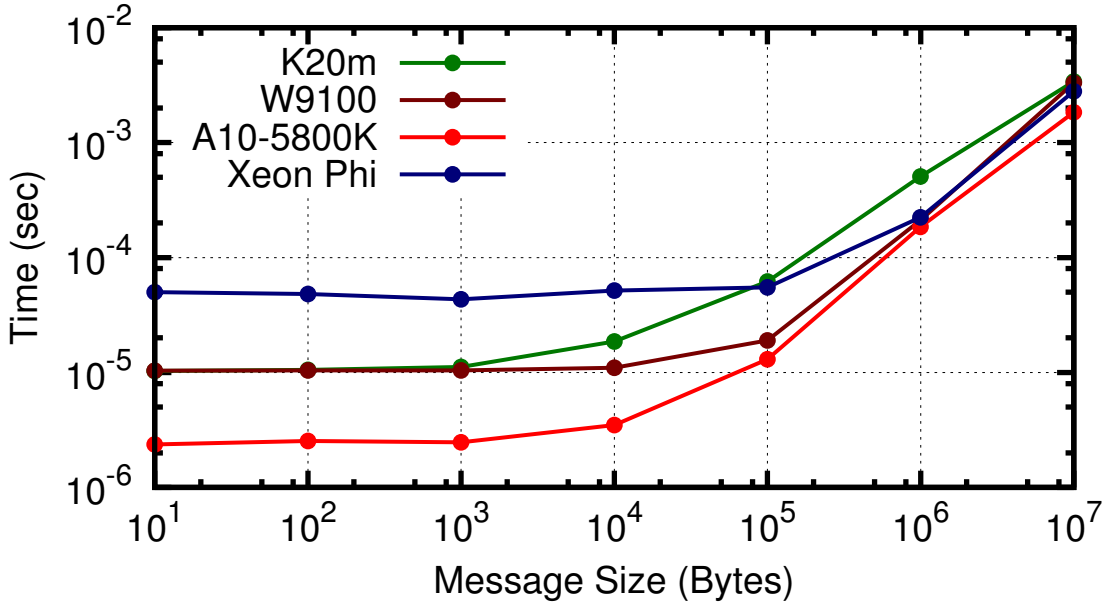
FLOPs/Byte

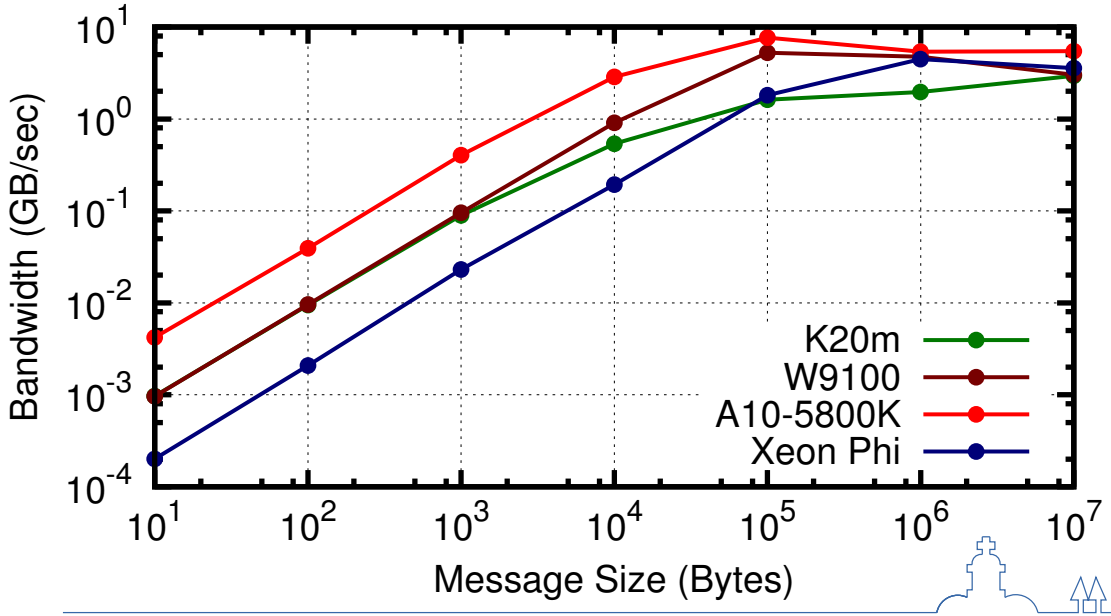


GPUs: Disillusion



Host-Device Communication





Conjugate Gradients

Pseudocode

Choose x_0
$p_0 = r_0 = b - Ax_0$
For $i = 0$ until convergence

1. Compute and store Ap_i
2. Compute $\langle p_i, Ap_i \rangle$
3. $\alpha_i = \langle r_i, r_i \rangle / \langle p_i, Ap_i \rangle$
4. $x_{i+1} = x_i + \alpha_i p_i$
5. $r_{i+1} = r_i - \alpha_i Ap_i$
6. Compute $\langle r_{i+1}, r_{i+1} \rangle$
7. $\beta_i = \langle r_{i+1}, r_{i+1} \rangle / \langle r_i, r_i \rangle$
8. $p_{i+1} = r_{i+1} + \beta_i p_i$

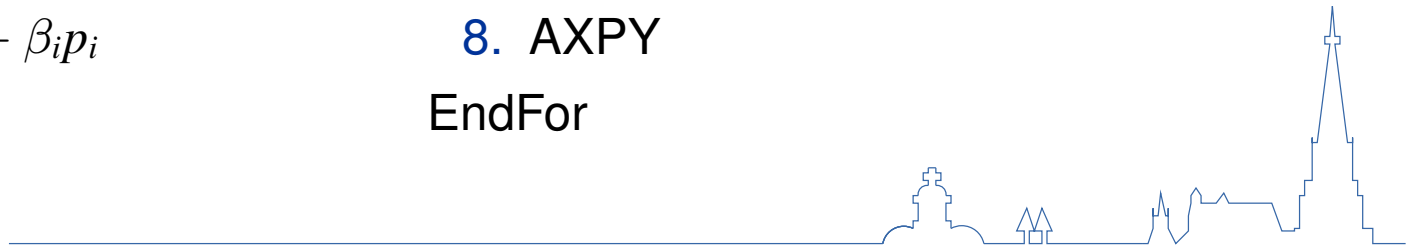
EndFor

BLAS-based Implementation

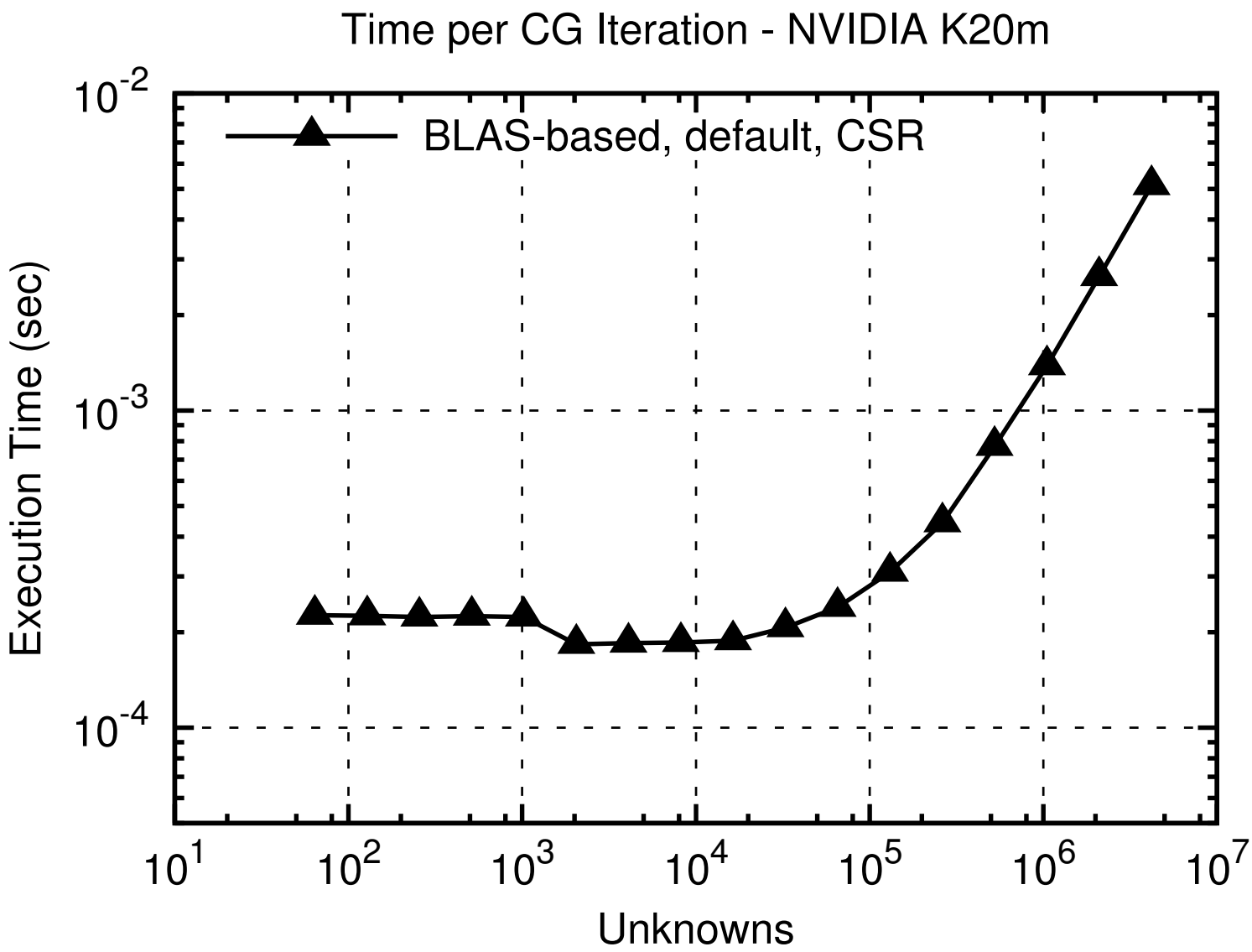
-
SpMV, AXPY
For $i = 0$ until convergence

1. SpMV ← No caching of Ap_i
2. DOT ← Global sync!
3. -
4. AXPY
5. AXPY ← No caching of r_{i+1}
6. DOT ← Global sync!
7. -
8. AXPY

EndFor

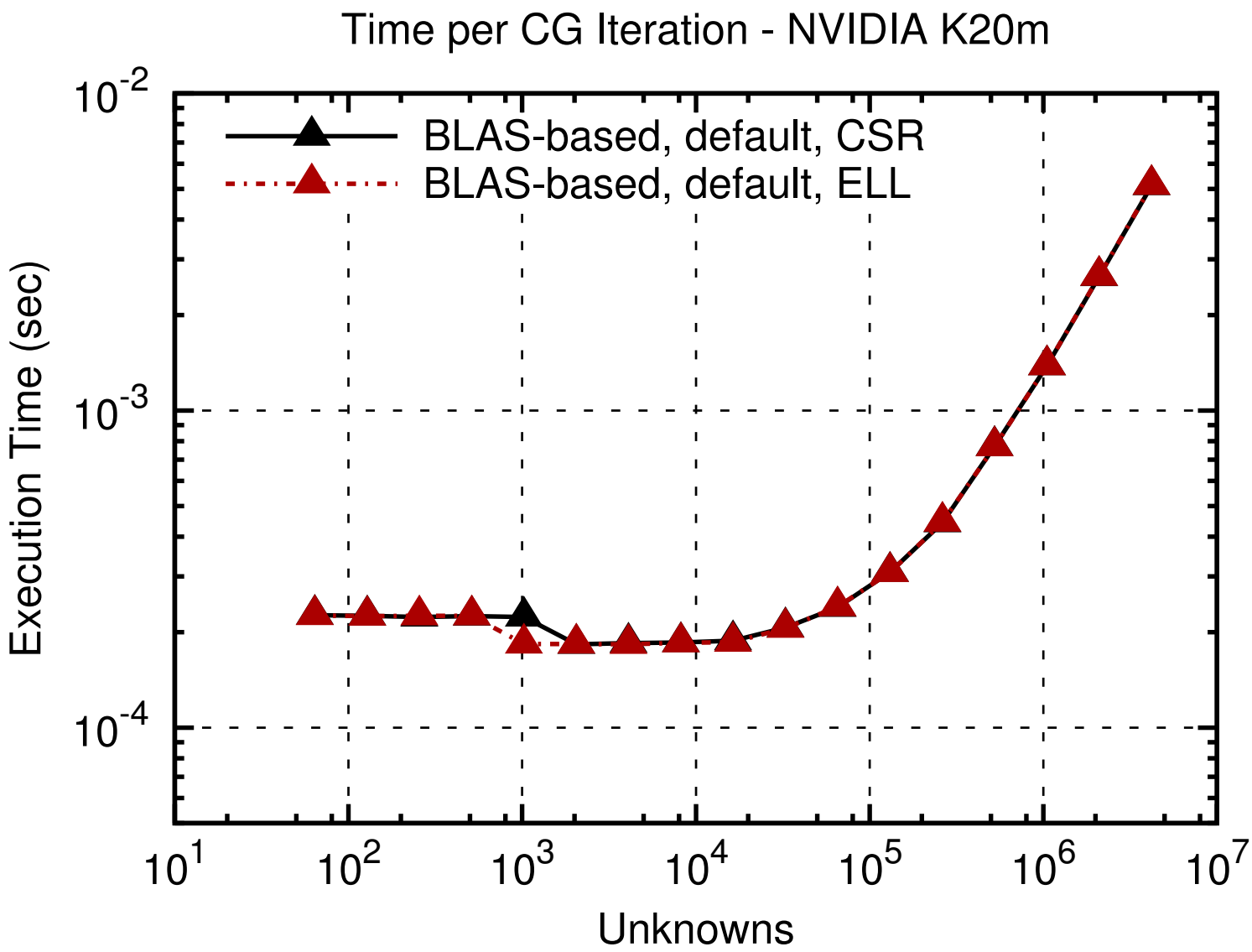


Conjugate Gradients



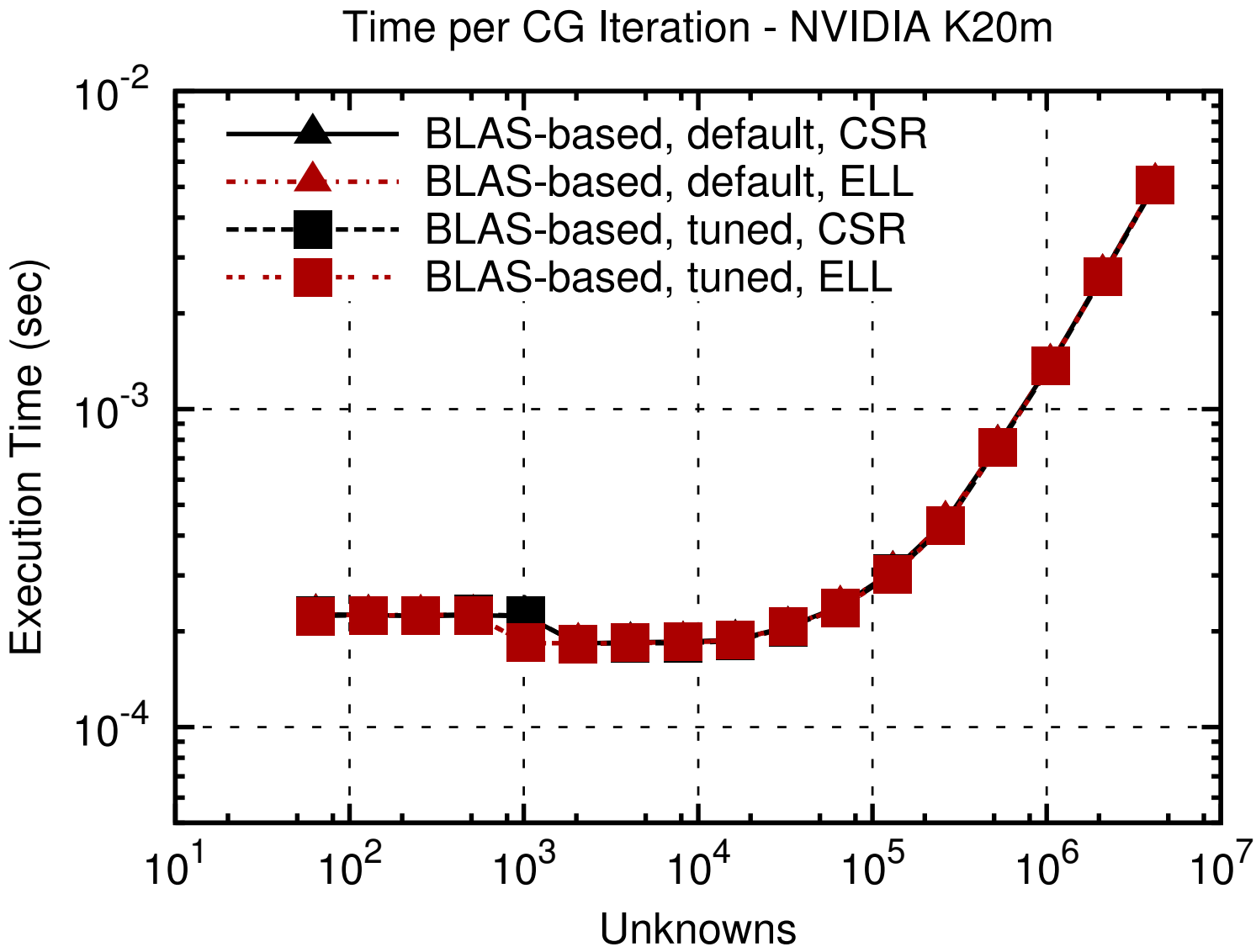
(2D Finite Difference Discretization)

Conjugate Gradients



(2D Finite Difference Discretization)

Conjugate Gradients

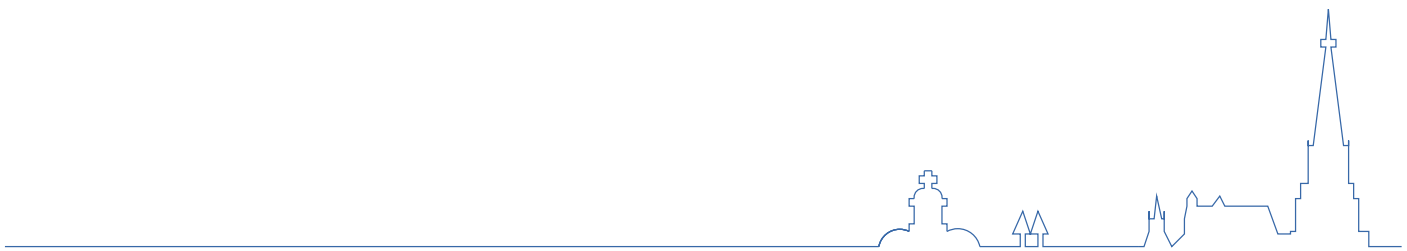


(2D Finite Difference Discretization)

Conjugate Gradient Optimizations

Optimization: Rearrange the algorithm

- Remove unnecessary reads
- Remove unnecessary synchronizations
- Use custom kernels instead of standard BLAS



Conjugate Gradients

Standard CG

Choose x_0

$p_0 = r_0 = b - Ax_0$

For $i = 0$ until convergence

1. Compute and store Ap_i
2. Compute $\langle p_i, Ap_i \rangle$
3. $\alpha_i = \langle r_i, r_i \rangle / \langle p_i, Ap_i \rangle$
4. $x_{i+1} = x_i + \alpha_i p_i$
5. $r_{i+1} = r_i - \alpha_i Ap_i$
6. Compute $\langle r_{i+1}, r_{i+1} \rangle$
7. $\beta_i = \langle r_{i+1}, r_{i+1} \rangle / \langle r_i, r_i \rangle$
8. $p_{i+1} = r_{i+1} + \beta_i p_i$

EndFor

Pipelined CG

Choose x_0

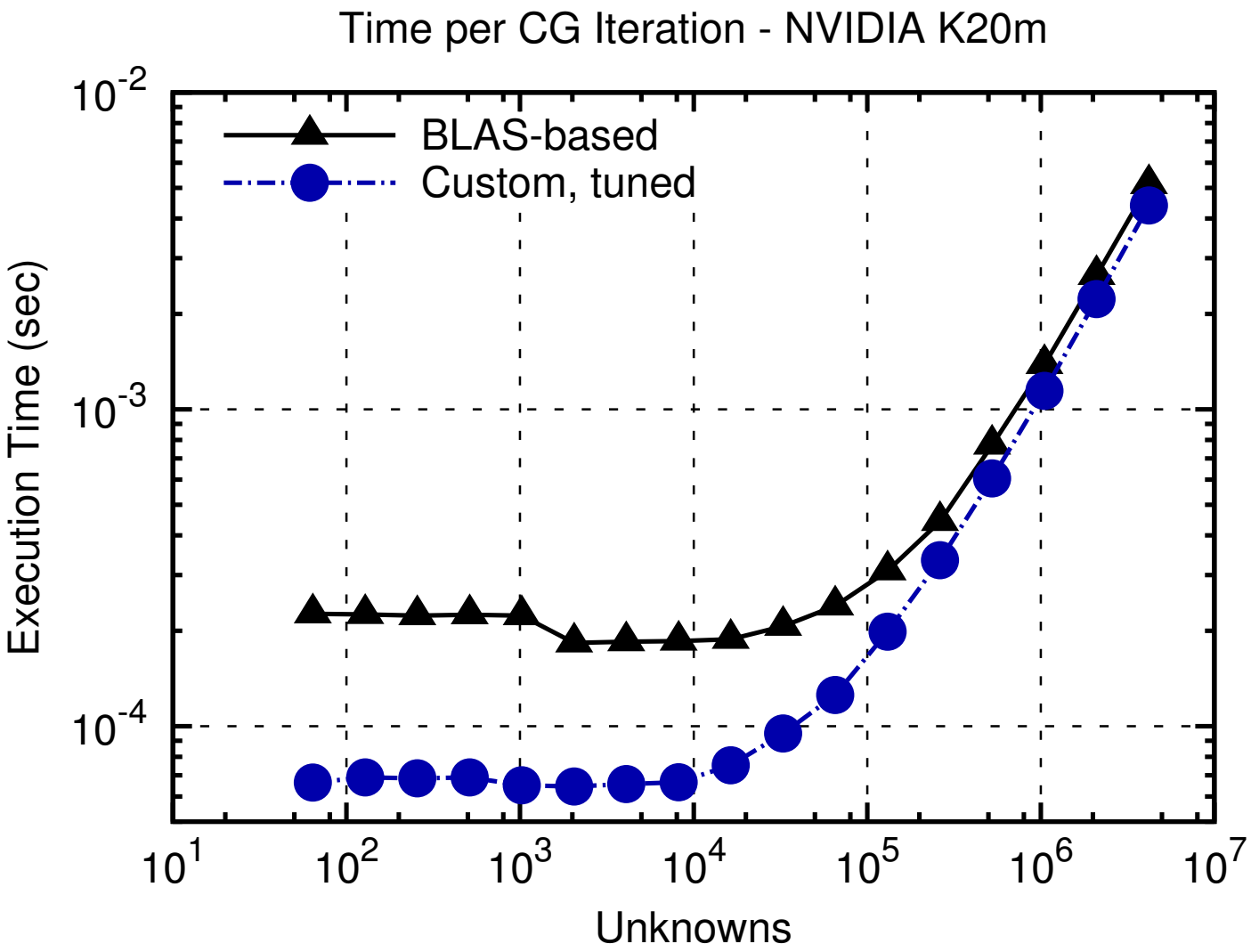
$p_0 = r_0 = b - Ax_0$

For $i = 1$ until convergence

1. $i = 1$: Compute α_0, β_0, Ap_0
2. $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
3. $r_i = r_{i-1} - \alpha_{i-1} Ap_i$
4. $p_i = r_i + \beta_{i-1} p_{i-1}$
5. Compute and store Ap_i
6. Compute $\langle Ap_i, Ap_i \rangle$, $\langle p_i, Ap_i \rangle$, $\langle r_i, r_i \rangle$
7. $\alpha_i = \langle r_i, r_i \rangle / \langle p_i, Ap_i \rangle$
8. $\beta_i = (\alpha_i^2 \langle Ap_i, Ap_i \rangle - \langle r_i, r_i \rangle) / \langle r_i, r_i \rangle$

EndFor

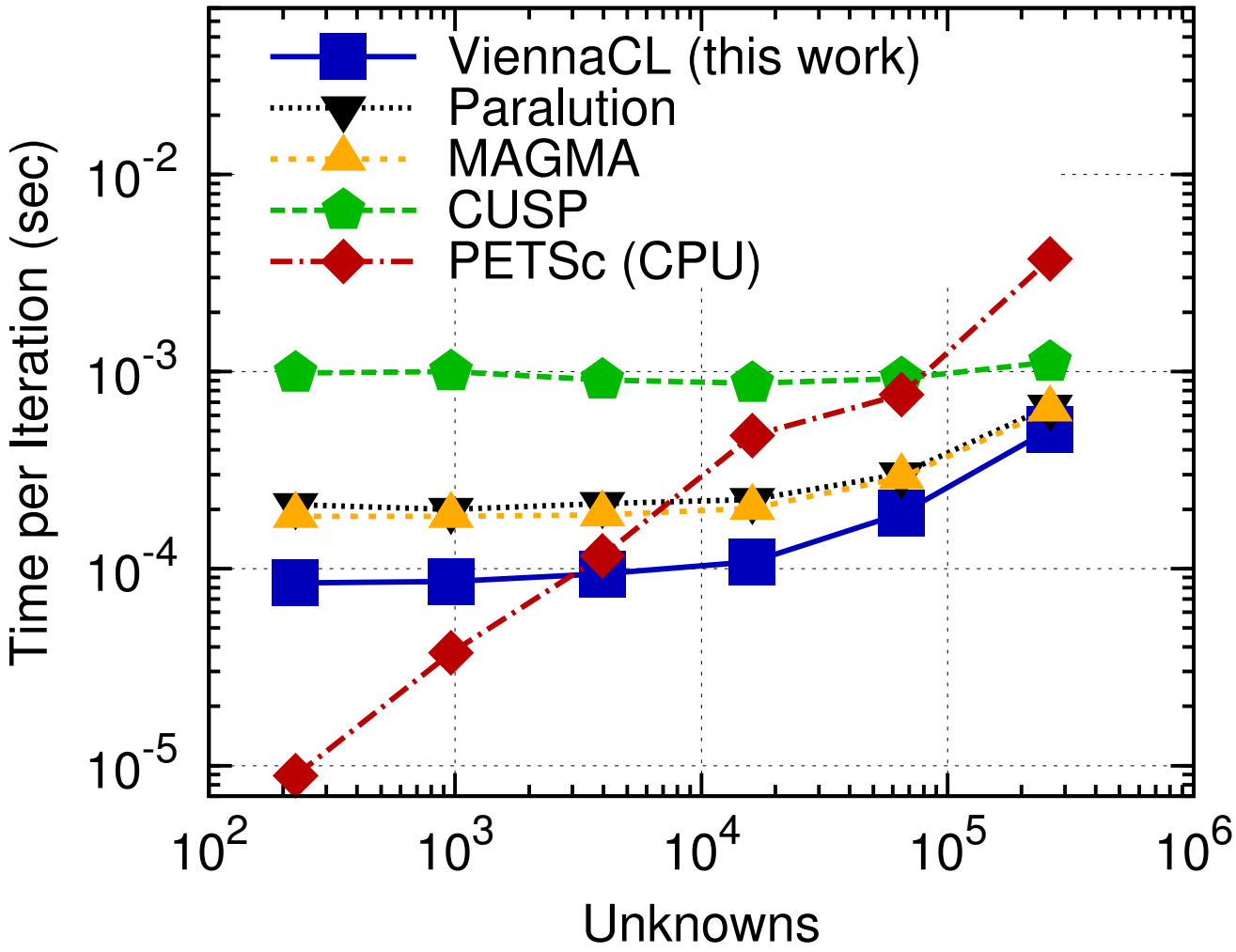
Conjugate Gradients



(2D Finite Difference Discretization)

Conjugate Gradients

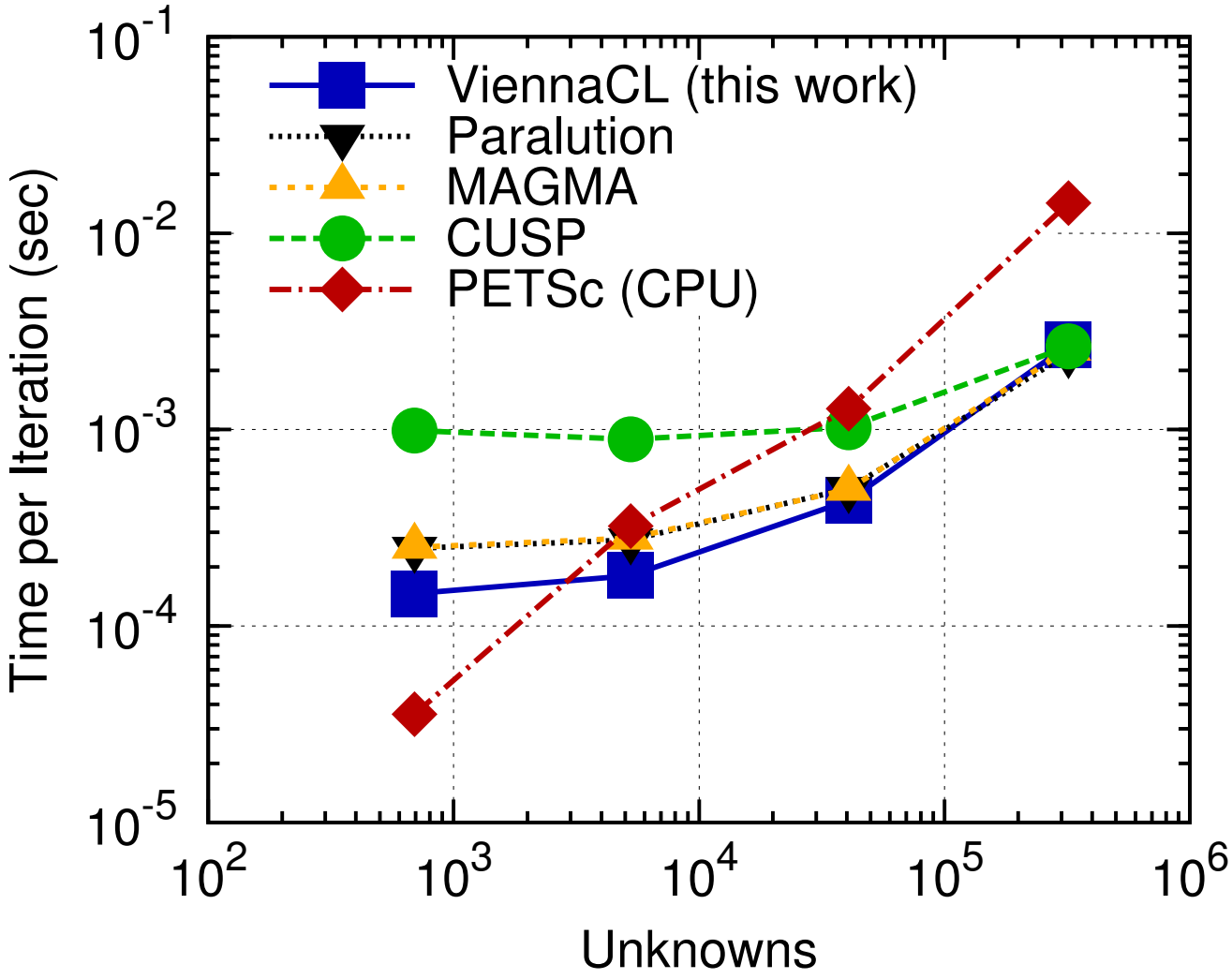
Time per CG Iteration - NVIDIA K20m



(Laplace 2D Finite Element Discretization, Unstructured)

Conjugate Gradients

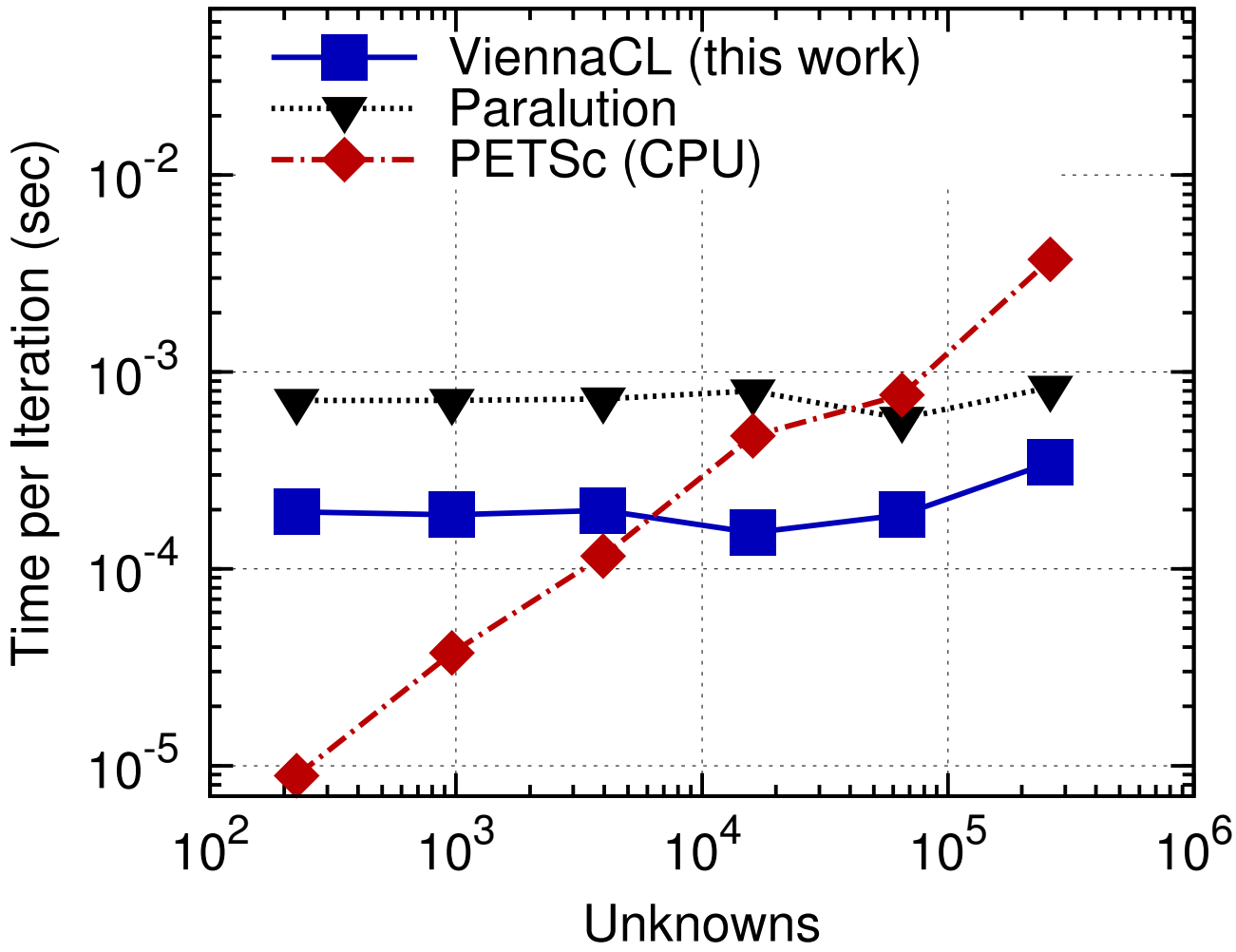
Time per CG Iteration - NVIDIA K20m



(3D Linear Elasticity, Linear Finite Elements, Unstructured)

Conjugate Gradients

Time per CG Iteration - AMD FirePro W9100



(Laplace 2D Finite Element Discretization, Unstructured)

Summary and Conclusion

Accelerator Architecture

- FLOPs for free
- Limited 'work window'

Host-Device Communication

- Latency of about 1-10 microseconds
- PCI-Express 3.0 addresses bandwidth, not latency
- Tighter integration with host desired

Advice for Implementations

- Count kernel launches to estimate minimum execution time
- Minimize data transfers to/from main memory