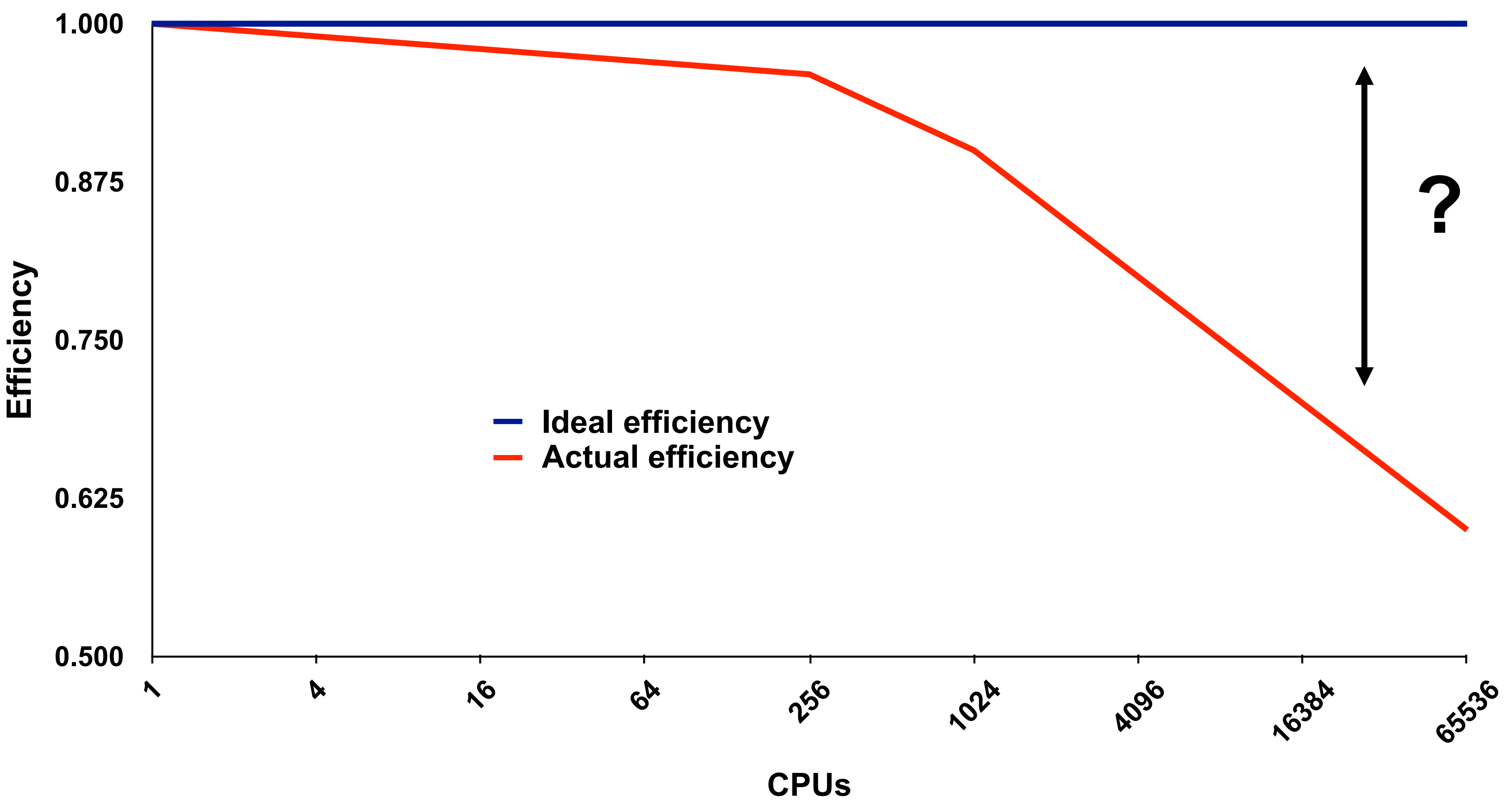
Identifying Scalability Bottlenecks with HPCToolkit

John Mellor-Crummey

Department of Computer Science
Rice University

15 November 2021

The Problem of Scaling



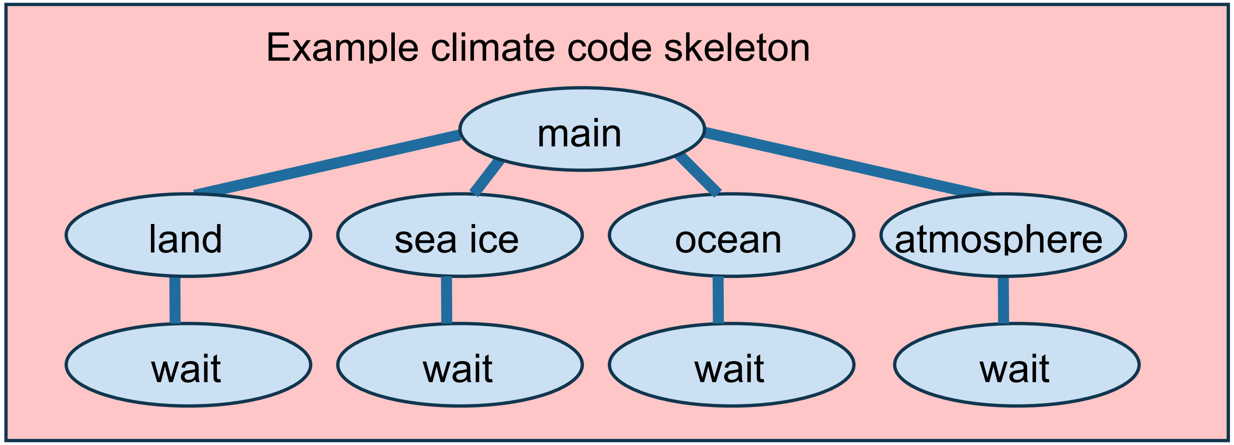
Note: higher is better

Goal: Automatic Scaling Analysis

- Pinpoint scalability bottlenecks
- Guide user to problems
- Quantify the magnitude of each problem
- Diagnose the nature of the problem

Challenges for Pinpointing Scalability Bottlenecks

- Parallel applications
  - modern software uses layers of libraries
  - performance is often context dependent



- Monitoring
  - bottleneck nature: computation, data movement, synchronization?
  - 2 pragmatic constraints (1) acceptable data volume, (2) low perturbation

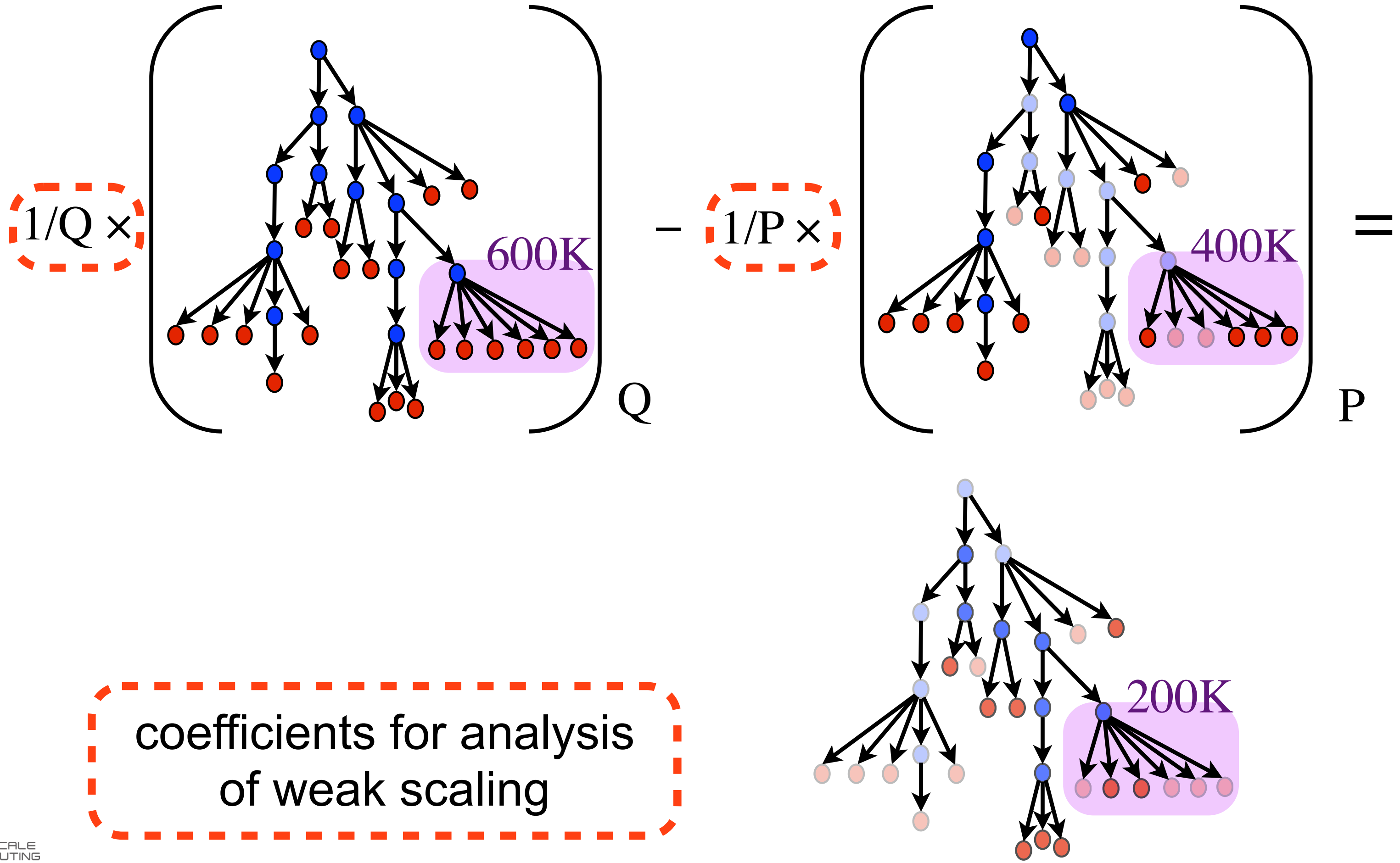
EXASCALE COMPUTING PROJECT

Scalability Analysis with Expectations

- **You have performance expectations for your parallel code**
  - strong scaling: linear speedup
  - weak scaling: constant execution time

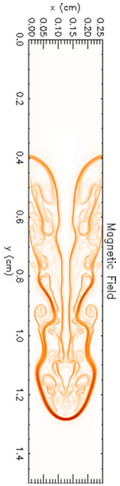
- **Put your expectations to work**
  - measure performance under different conditions
    - e.g., different levels of parallelism and/or different inputs
  - express your expectations as an equation
  - compute the deviation from expectations for each calling context
    - for both inclusive and exclusive costs
  - correlate the metrics with the source code
  - explore the annotated call tree interactively

Pinpointing and Quantifying Scalability Bottlenecks

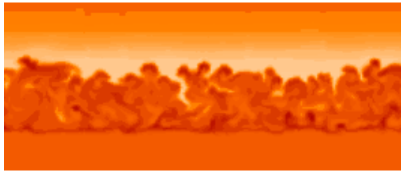


Scalability Analysis Demo

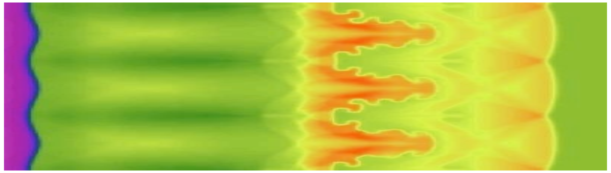
- Parallel, adaptive-mesh refinement (AMR) code
- Block structured AMR; a block is the unit of computation
- Designed for compressible reactive flows
- Can solve a broad range of (astro)physical problems
- Portable: runs on many massively-parallel systems
- Scales and performs well
- Fully modular and extensible: components can be combined to create many different applications



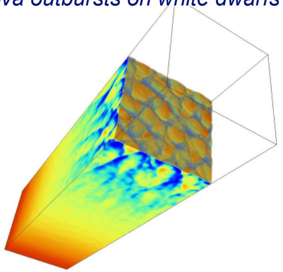
Magnetic Rayleigh-Taylor



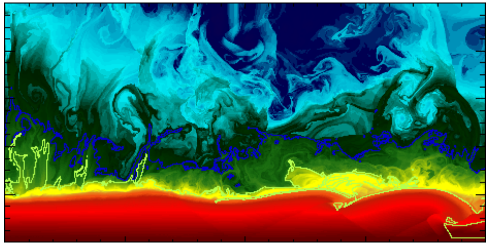
Nova outbursts on white dwarfs



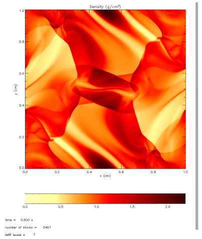
Laser-driven shock instabilities



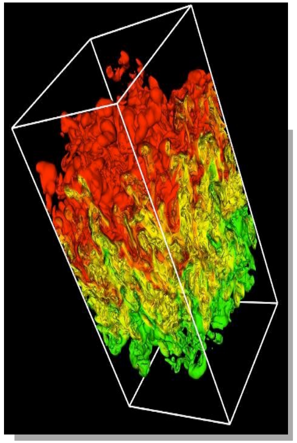
Cellular detonation



Helium burning on neutron stars



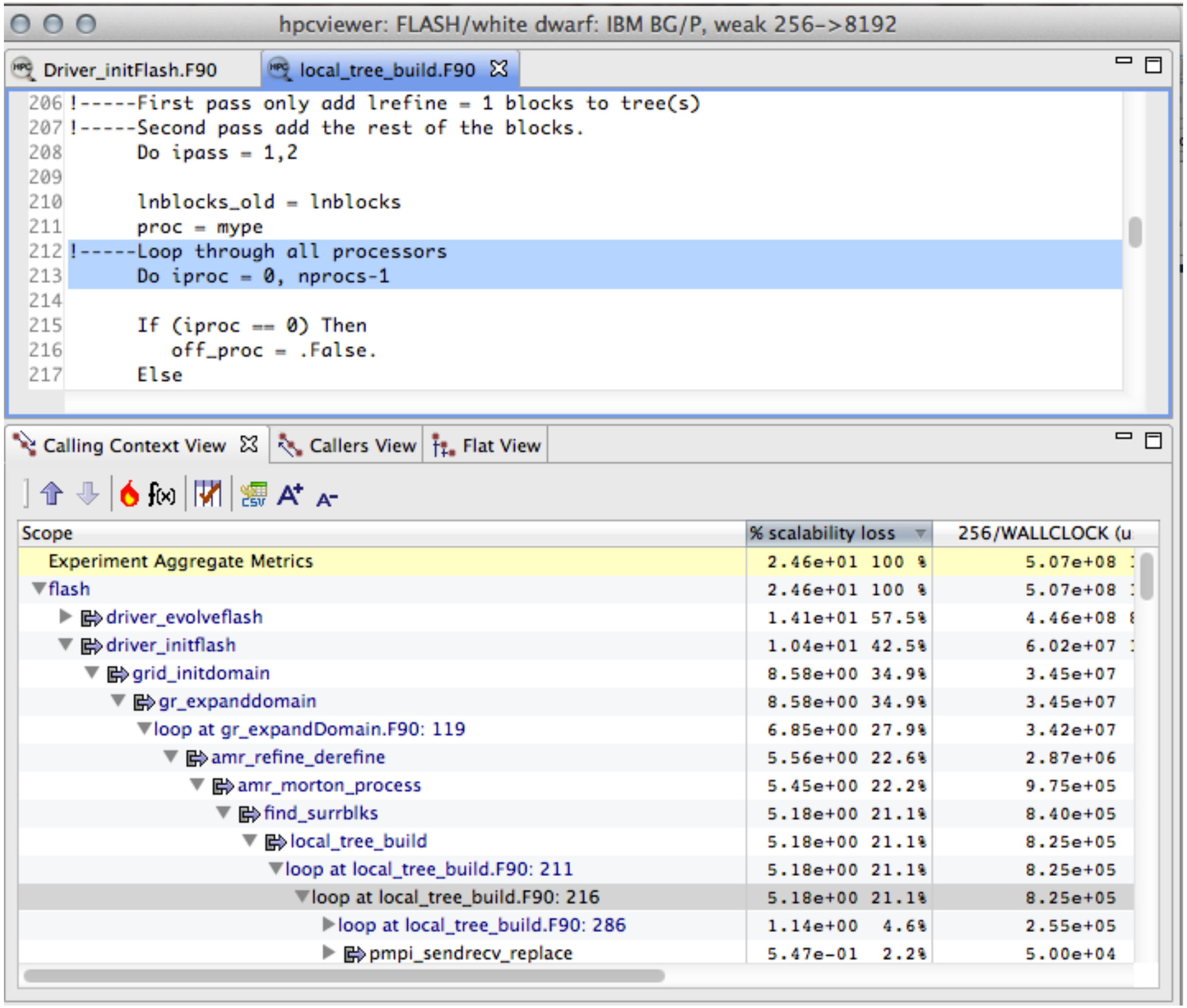
Orzag/Tang MHD vortex



Rayleigh-Taylor instability

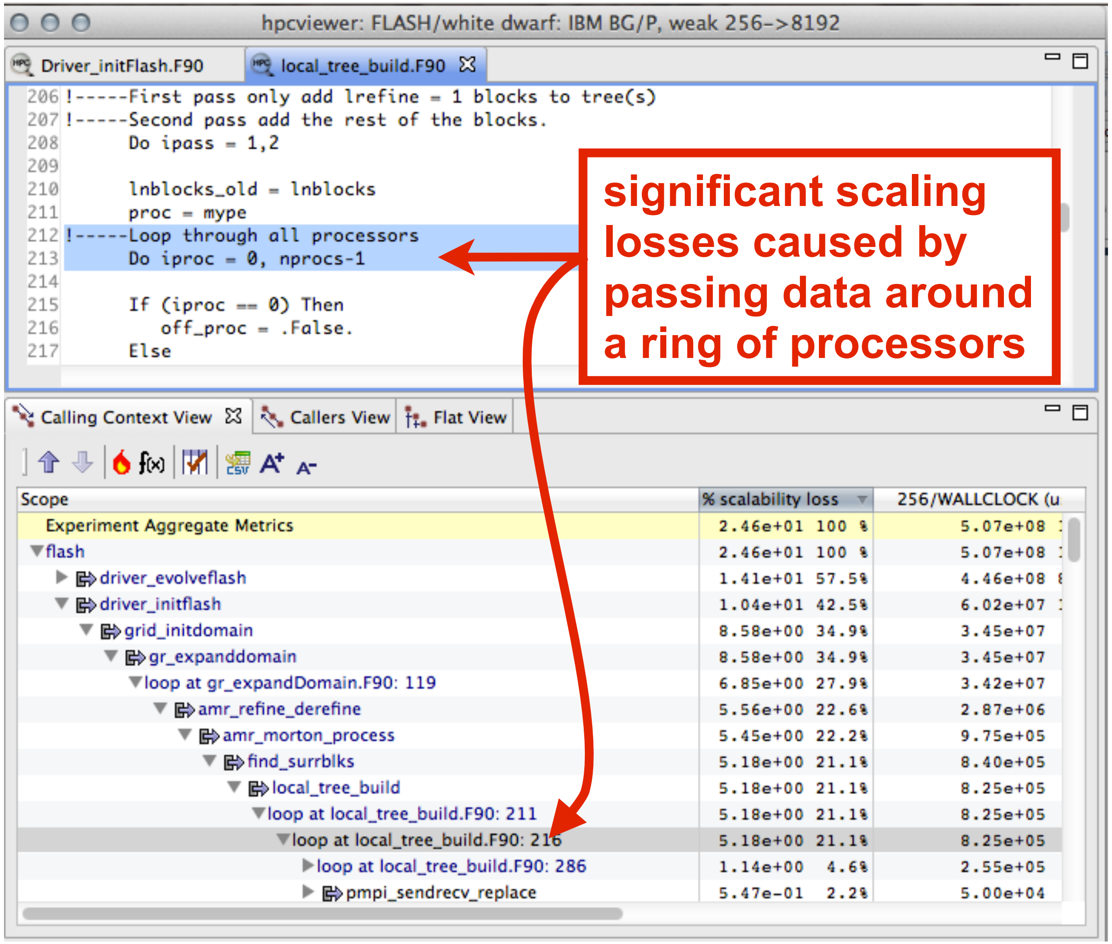
Figures courtesy of FLASH Team, University of Chicago

Scalability Analysis of Flash (Demo)

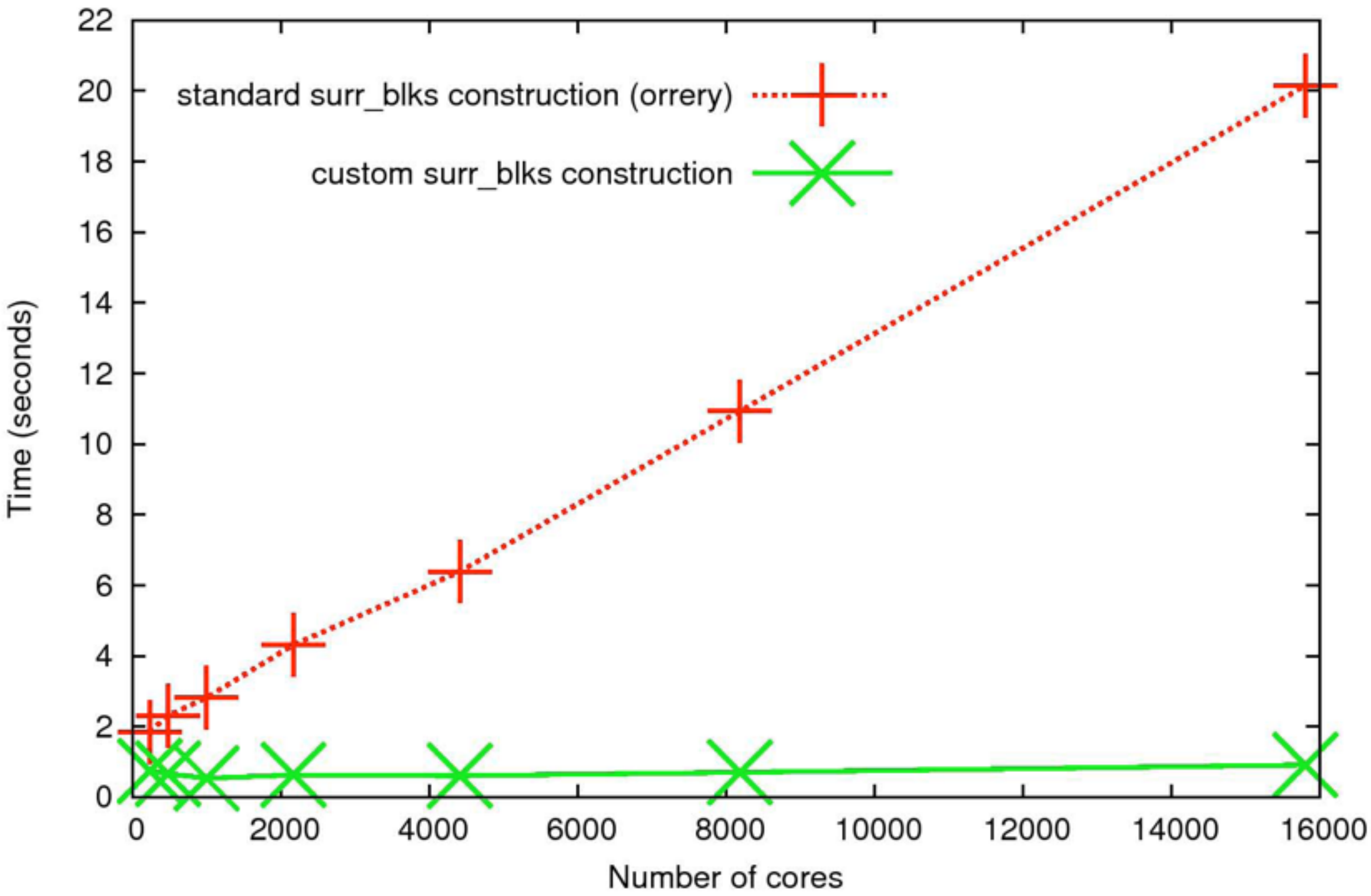


Scalability Analysis of FLASH

- Difference call path profile from two executions
  - different number of nodes
  - different number of threads
- Pinpoint and quantify scalability bottlenecks within or across nodes



Improved Flash Scaling of AMR Setup



Graph courtesy of Anshu Dubey, U Chicago

Scalability Loss in Practice

- **Try computing scaling losses across nodes as you increase the node count**
- **Try computing scaling losses within nodes as you increase the thread count**

- **How?**
  - use hpcrun to monitor an execution at scale P
    - e.g. mpirun -n 100 hpcrun -o foo.m …
  - use hpcrun to monitor an execution at scale Q
    - e.g. mpirun -n 1000 hpcrun -o bar.m …
  - use hpcstruct to analyze binaries in each directory
  - invoke hpcprof passing both of the directories
    - e.g., hpcprof foo.m bar.m
  - open the resulting database in hpcviewer and use the equation to analyze your losses!

References

- **Cristian Coarfa, John Mellor-Crummey, Nathan Froyd, and Yuri Dotsenko. 2007. Scalability analysis of SPMD codes using expectations. In Proceedings of the 21st annual International Conference on Supercomputing (ICS '07). Association for Computing Machinery, New York, NY, USA, 13–22. DOI:https://doi.org/10.1145/1274971.1274976**

- **Rice University, HPCToolkit Users Manual. http://hpctoolkit.org/manual/HPCToolkit-users-manual.pdf**

EXASCALE COMPUTING PROJECT