Benchmarks and Performance Results

Chapel Team, Cray Inc.

Chapel version 1.11

April 2, 2015

Safe Harbor Statement

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts. These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.

Executive Summary

- **Generally speaking, performance has improved with 1.11**
- **Previous slide decks have shown performance changes:**

  …due to vectorization
  …due to LICM improvements
  …due to ugni+muxed as default
  …due to parallel range iteration improvements
  …due to the local field pragma

- **These slides contain additional v1.11 performance results**
  - not tied to any specific effort, just comparisons across releases

Outline

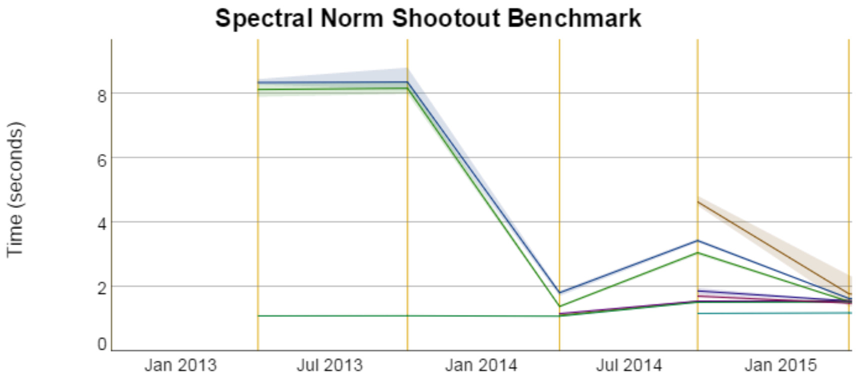
- Shootout Benchmark Status
- Single Locale Performance Trends
- Compiler Performance Trends
- Multi-locale Performance Trends
- ugni+qthreads Performance Trends
- Performance Scalability Study
- Performance Priorities and Next Steps

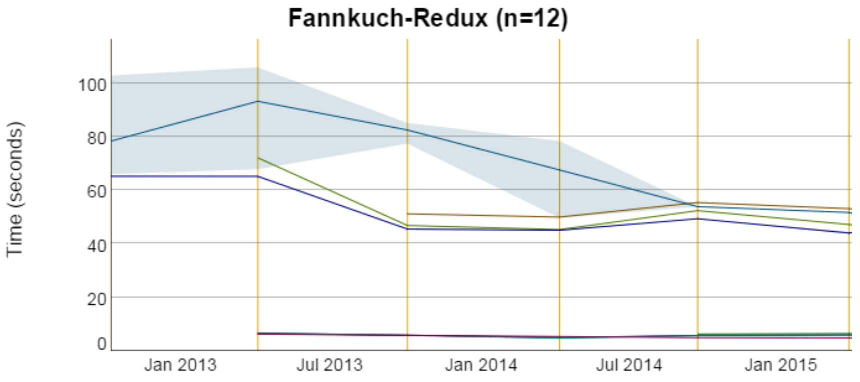
Shootout Benchmarks Status

Shootout Benchmark Summary

- **By design, not much effort put into shootouts for 1.11**
  - --no-local timings improved for some cases due to locality work
  - A few of our fastest versions improved, but most stayed the same
  - Several of our non-fastest versions also improved

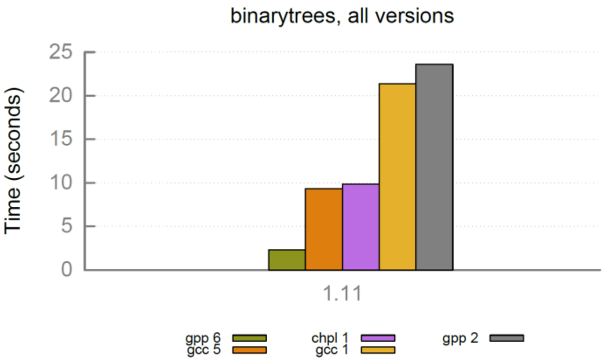
    ⇒ Chapel becoming less sensitive to writing in a specific style

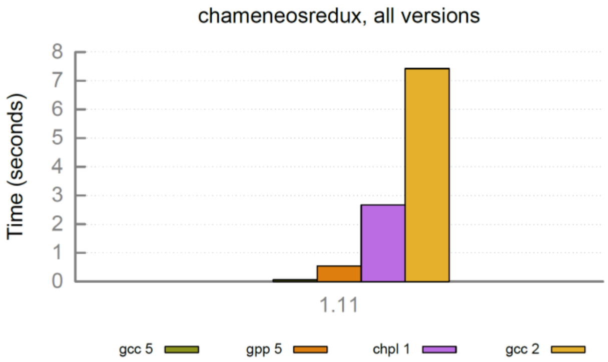


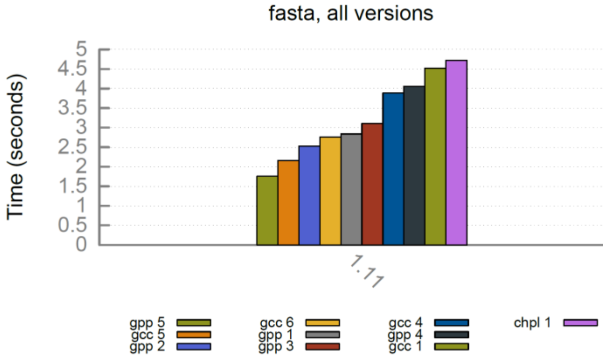


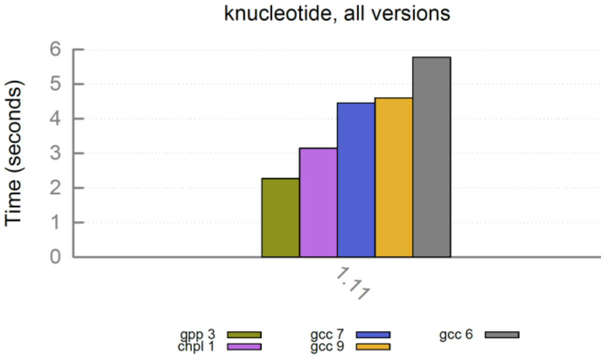
Shootout Performance Standings

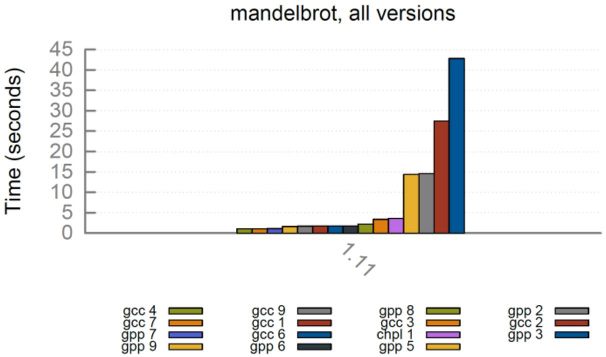
- **Chapel versions (purple) compared to C/C++ references**
  - Timings taken on 2x4-core Intel Xeon processors w/ gcc 4.7.2



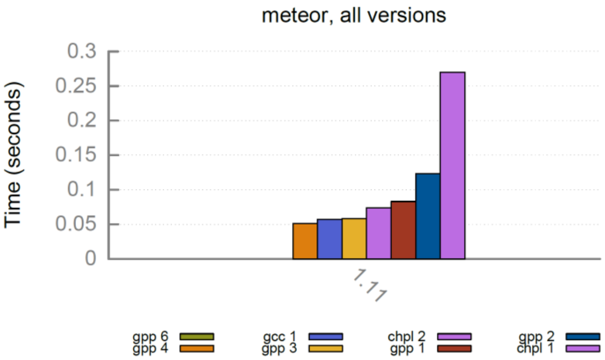




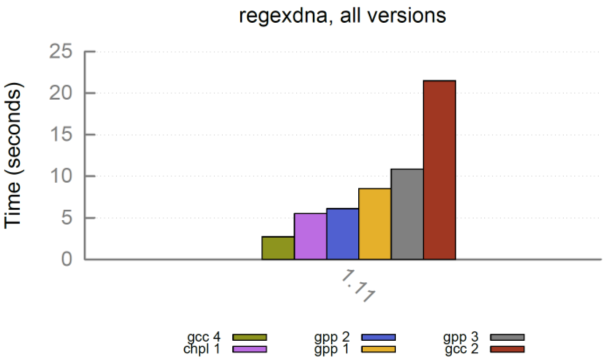


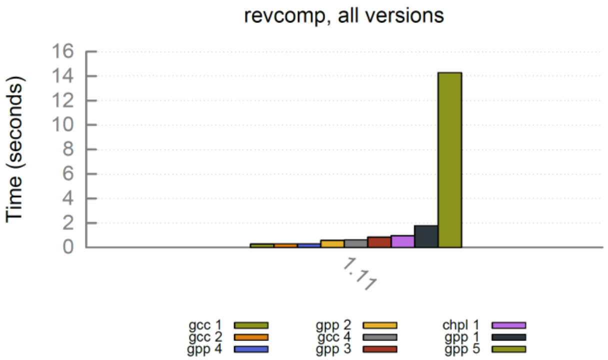


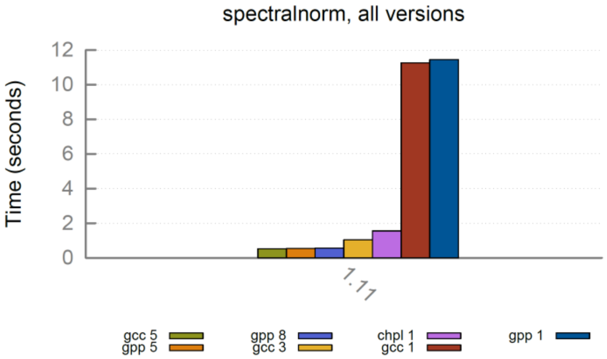
Shootout Performance Standings (continued)

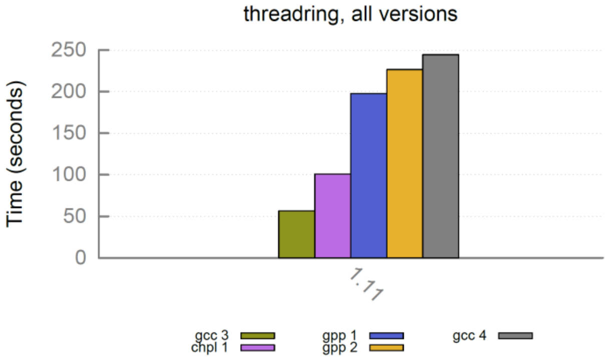


Note: two Chapel versions above, one faster, one more elegant









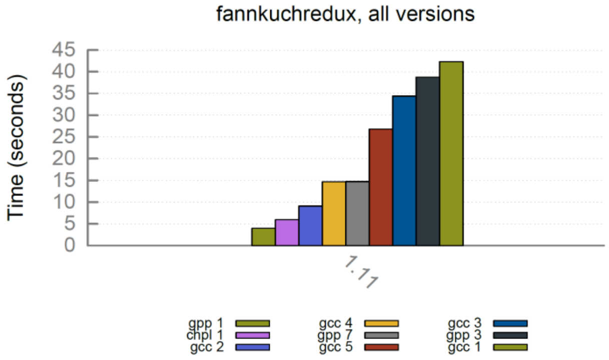
Shootout Performance Standings (continued)

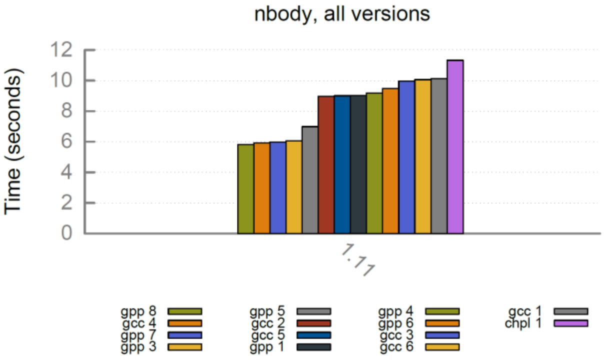
- **The following cases deserve additional notes:**

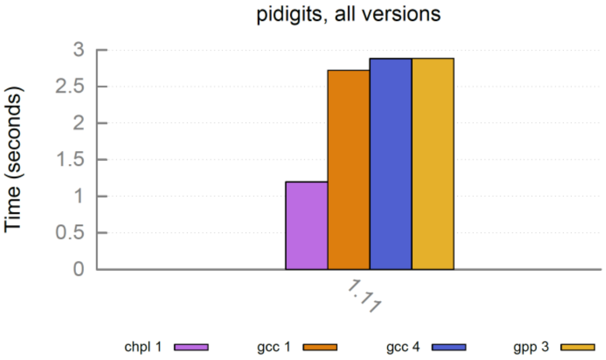
  fannkuch-redux:
  - reference versions hard-code #threads for the 4-core shootout system
  - Chapel doesn't, to its benefit on this 8-core system

  nbody:
  - the five fastest reference versions use vector intrinsics
  - the Chapel version vectorizes, yet not with gcc 4.7.2, so no benefit there

  pidigits:
  - the reference versions use an older system installation of GMP
  - Chapel uses a newer, bundled version that results in the difference



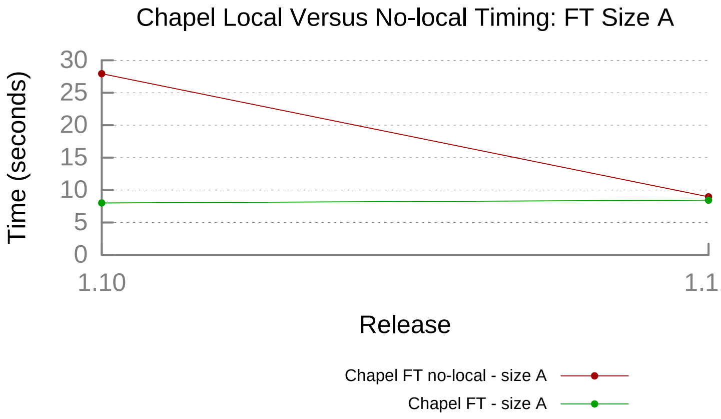


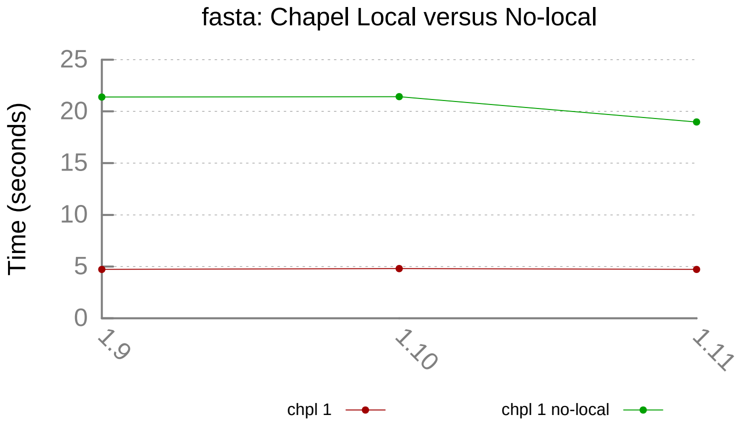


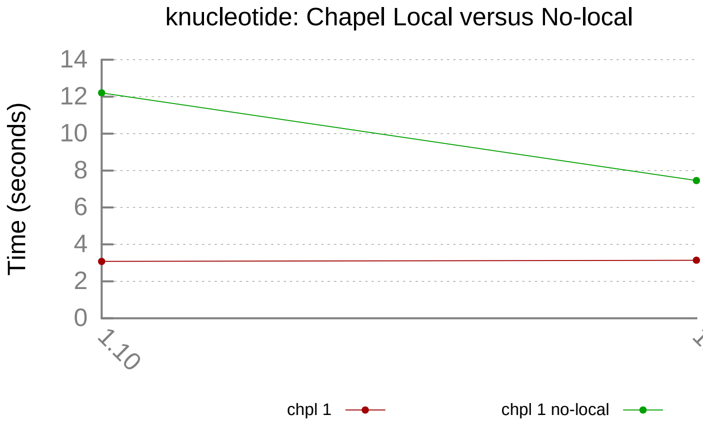
Single Locale Performance Trends

Single Locale Performance

- **No-local execution improved due to better local analysis**
  - More no-local executions complete without timing out in test system
    - e.g., FT size B, IS size A

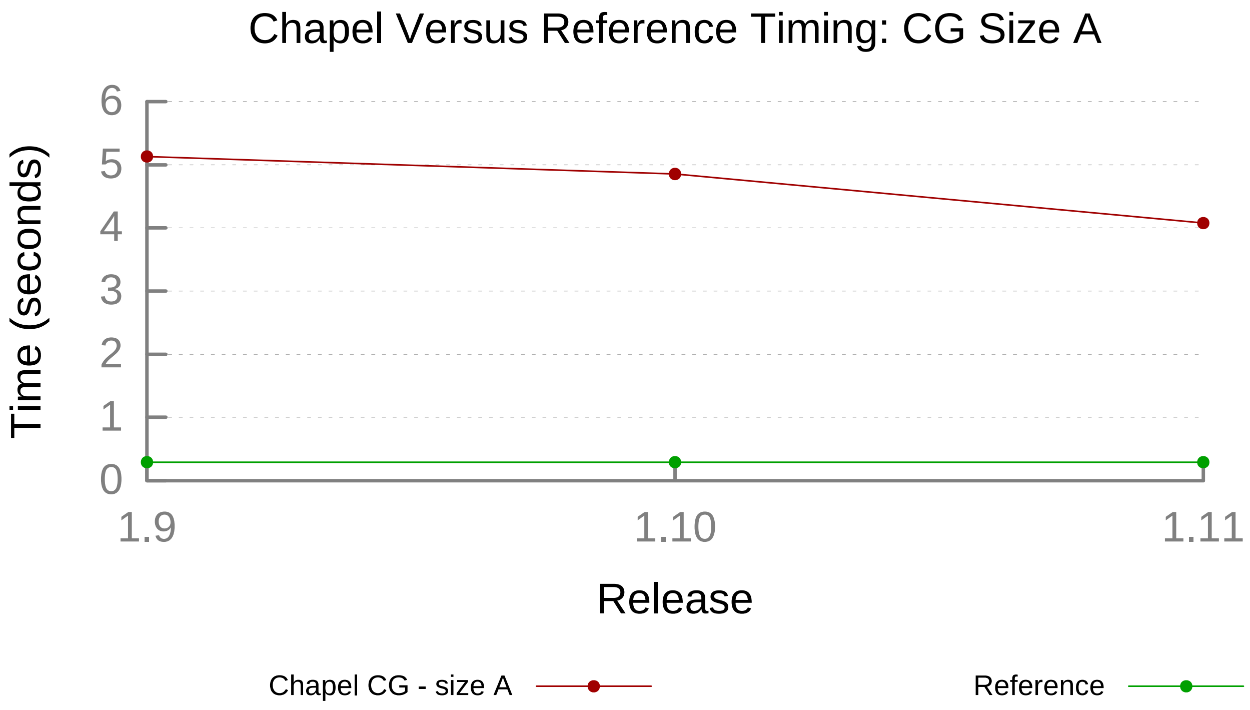






Single Locale Performance

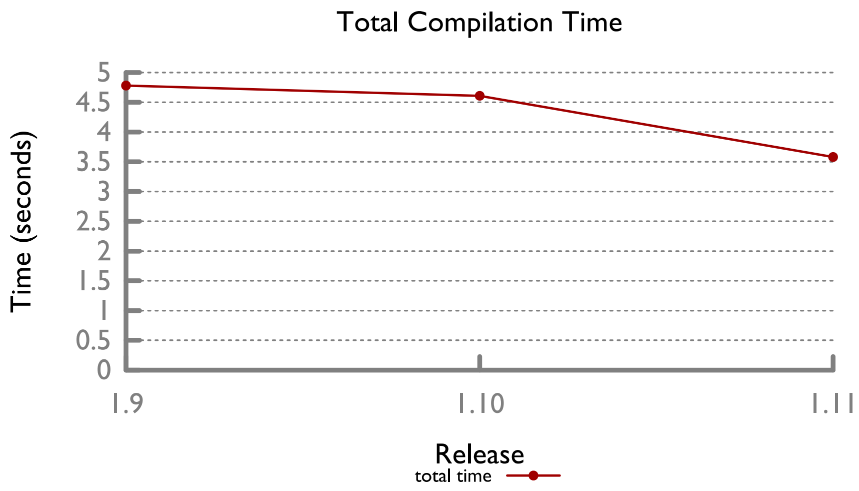
- Improvements to sparse iterators helped CG performance

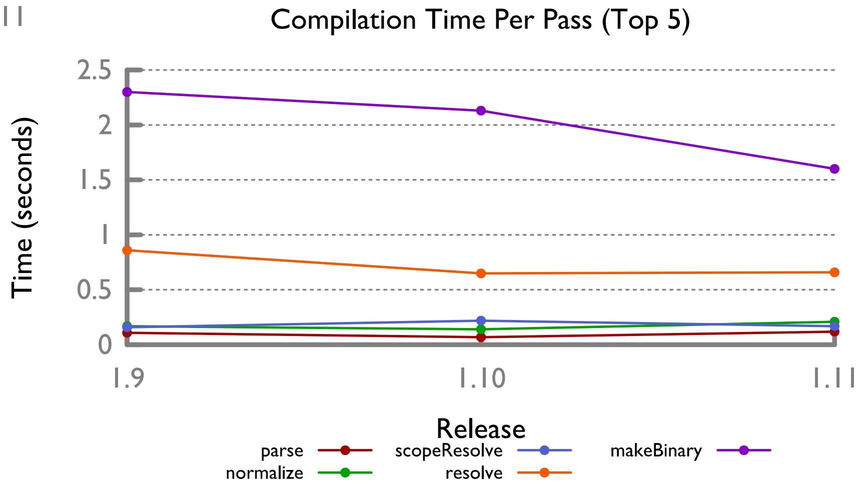


Compiler Performance Trends

Compiler Performance

- Compilation time has improved by ~1 second for all tests

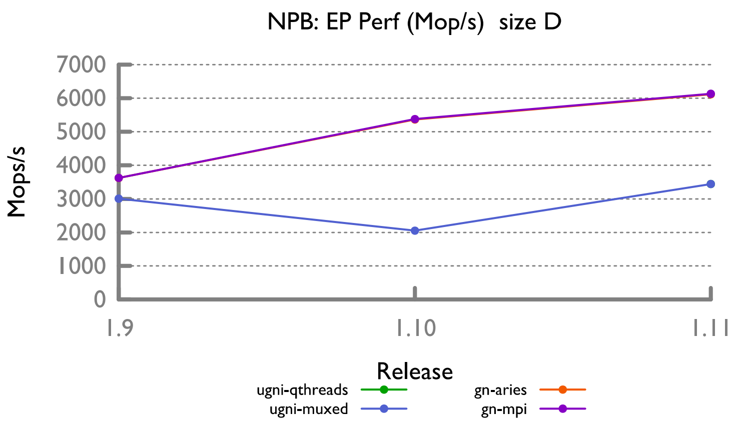


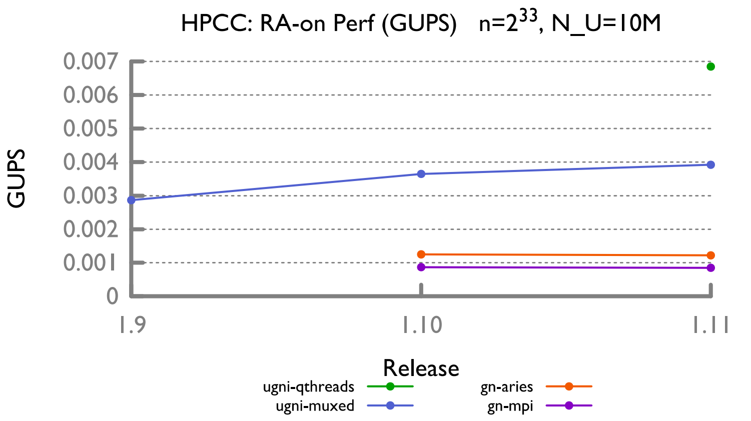


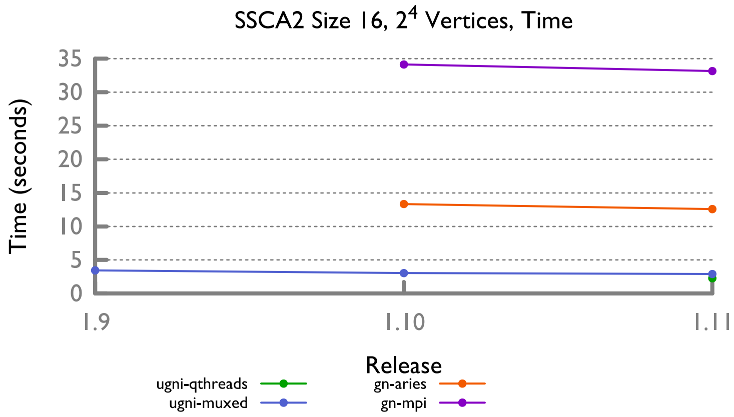
Multi-Locale Performance Trends

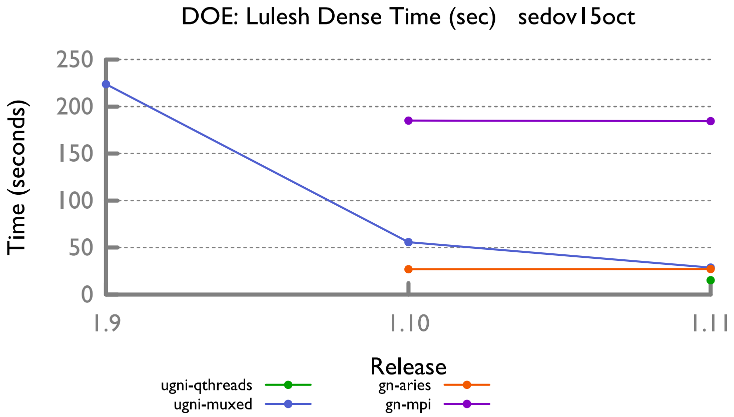
Multi-locale Performance

- Most benchmarks have remained the same or improved slightly


NPB: EP Perf (Mop/s) size D

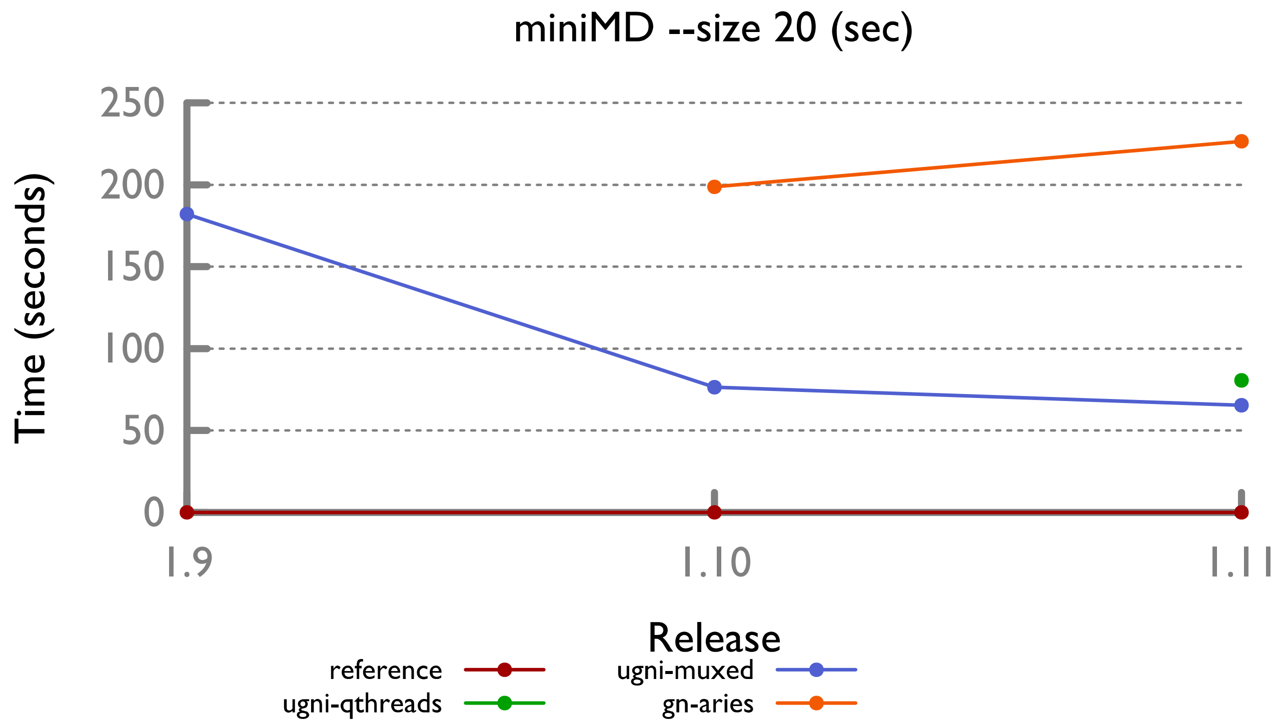

HPCC: RA-on Perf (GUPS) $n=2^{33}$, N_U=10M


SSCA2 Size 16, 2^4 Vertices, Time


DOE: Lulesh Dense Time (sec) sedov15oct

Multi-locale Performance

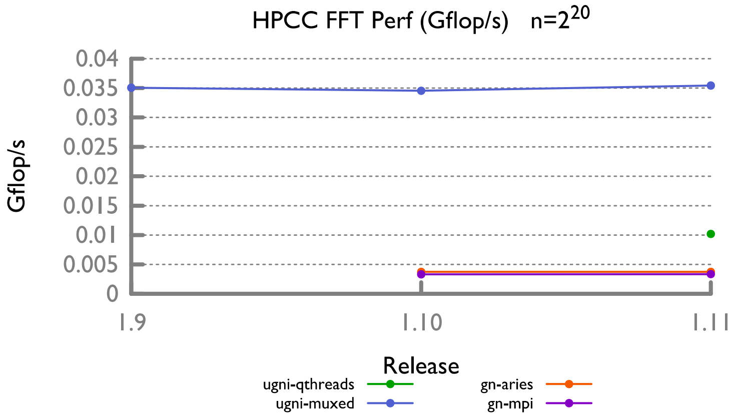
- **miniMD has gotten ~10% slower for gasnet-aries**
  - seemingly related to local class optimization
    - regression discovered while assembling this report
    - not yet sure what happened yet

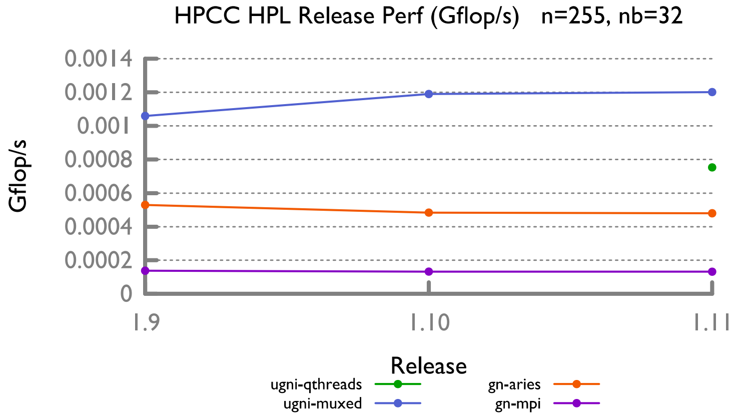


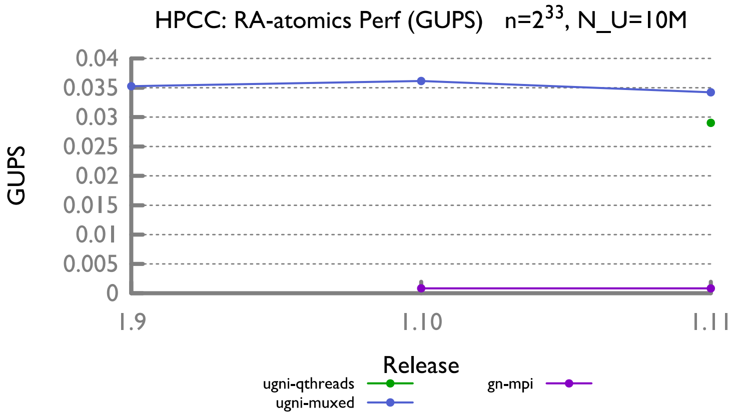
ugni+qthreads Performance Trends

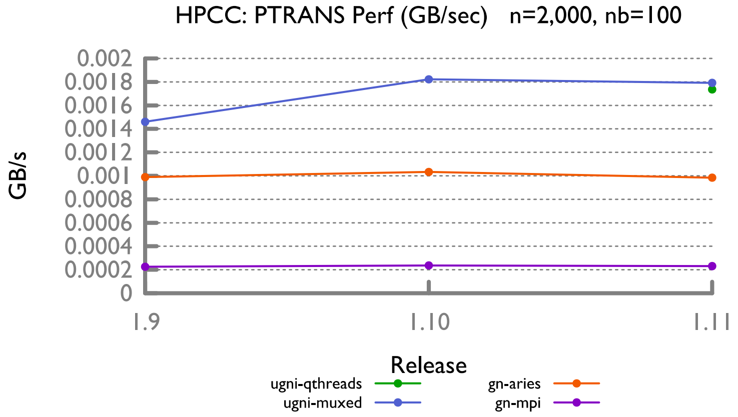
ugni+qthreads Performance

- ugni+qthreads is sometimes competitive with ugni+muxed


HPCC FFT Perf (Gflop/s) n=2^{20}

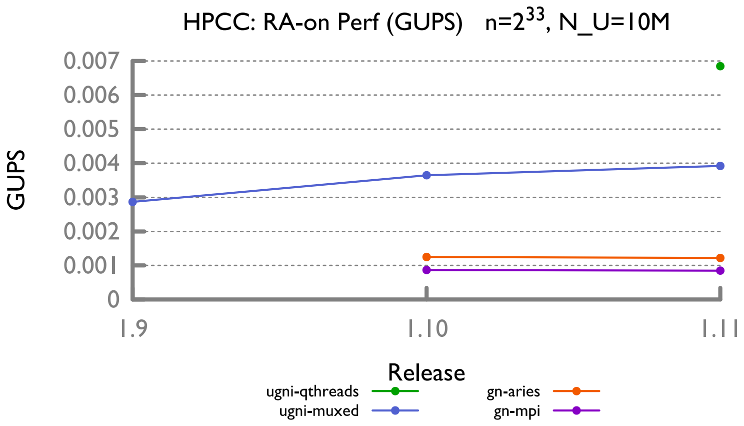

HPCC HPL Release Perf (Gflop/s) n=255, nb=32

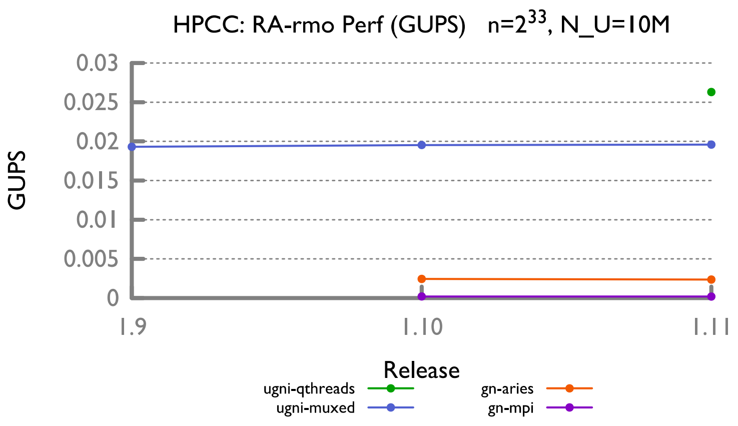

HPCC: RA-atomics Perf (GUPS) n=2^{33}, N_U=10M

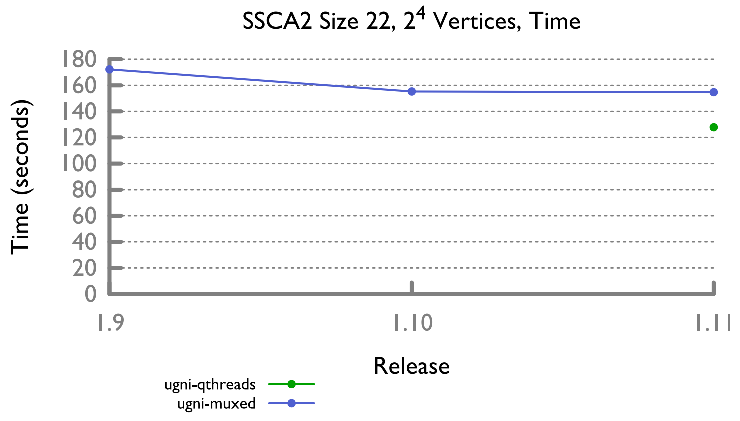

HPCC: PTRANS Perf (GB/sec) n=2,000, nb=100

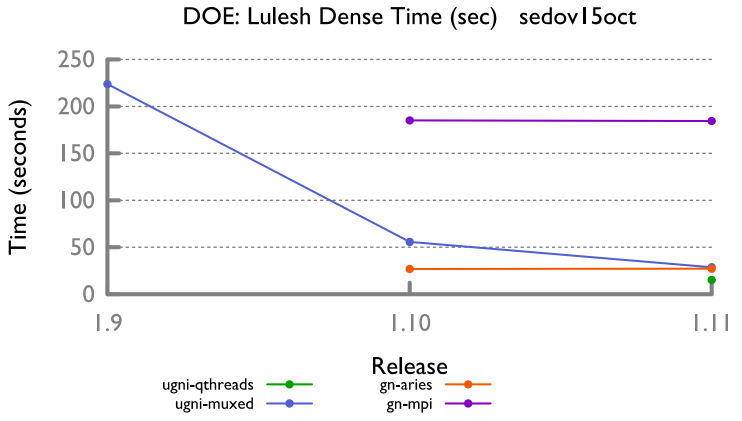
ugni+qthreads Performance

- **In other cases ugni+qthreads outperforms ugni+muxed**


HPCC: RA-on Perf (GUPS) n=2^{33}, N_U=10M


HPCC: RA-rmo Perf (GUPS) n=2^{33}, N_U=10M


SSCA2 Size 22, 2^4 Vertices, Time

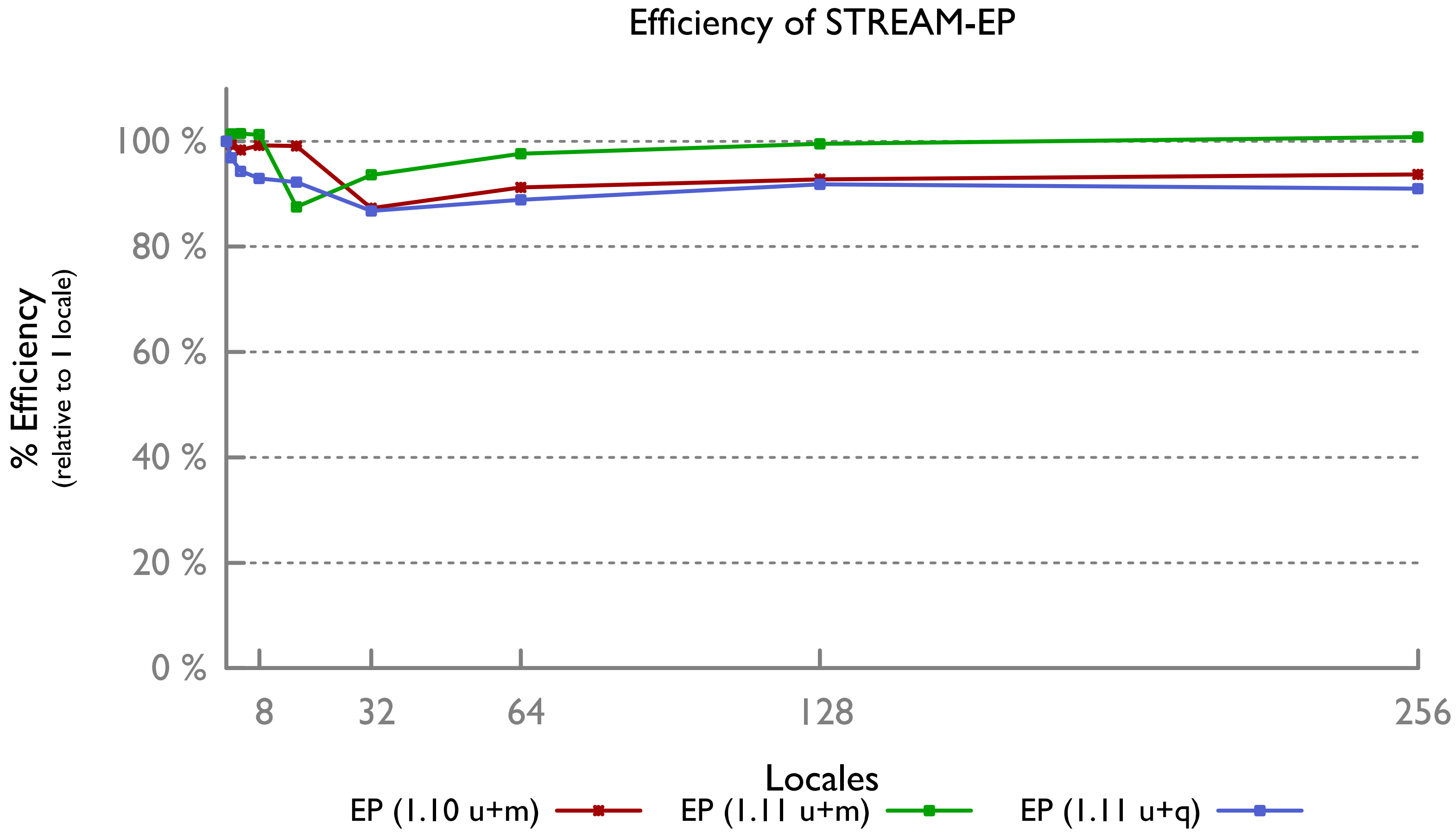

DOE: Lulesh Dense Time (sec) sedov15oct

Performance Scalability Study

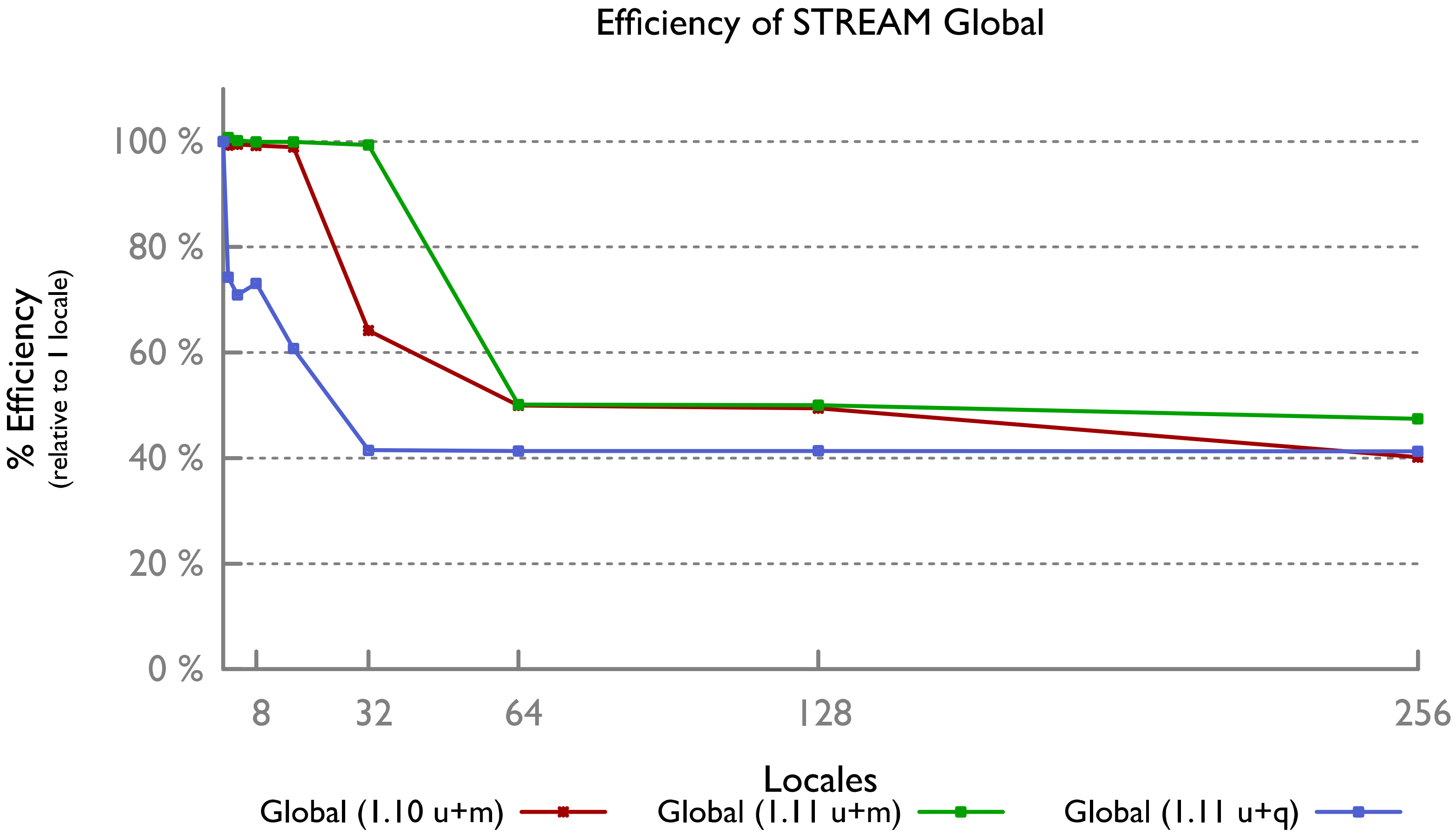
Scalability Study: Background

- **We continued the scalability study from last release**
  - HPCC Stream: EP and Global
  - HPCC RA: atomic, on-based, and remote memory operations (rmo)
    - these test network atomics, active messages, and puts/gets, respectively
  - Reduction of an array
- **All experiments shown here were performed on a Cray XC**
  - 1-256 locales (up from 1-64 from last release)
  - ugni+muxed and ugni+qthreads runtimes
- **The following slides highlight a few notable cases**

Scalability: STREAM-EP Efficiency

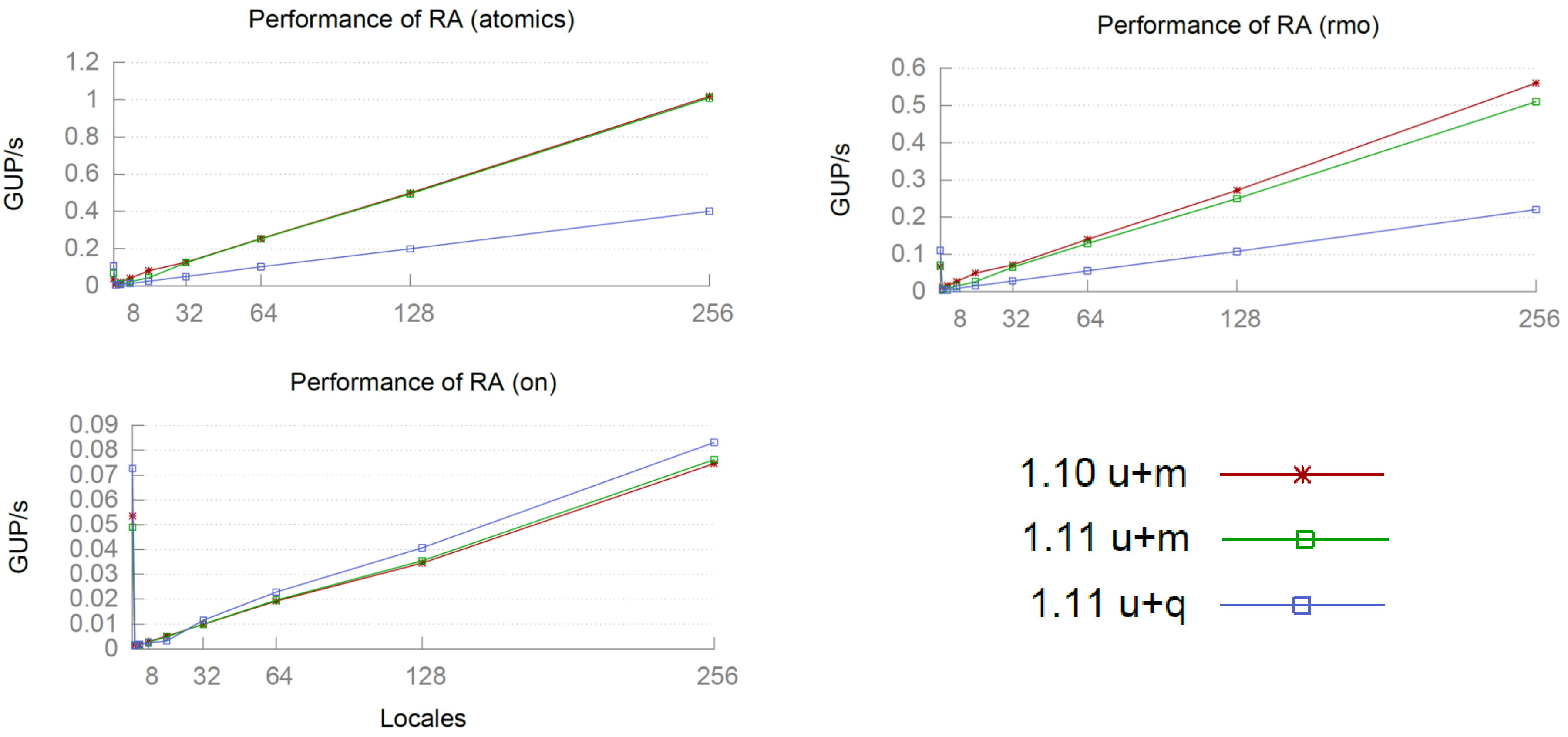


Scalability: STREAM Global Efficiency

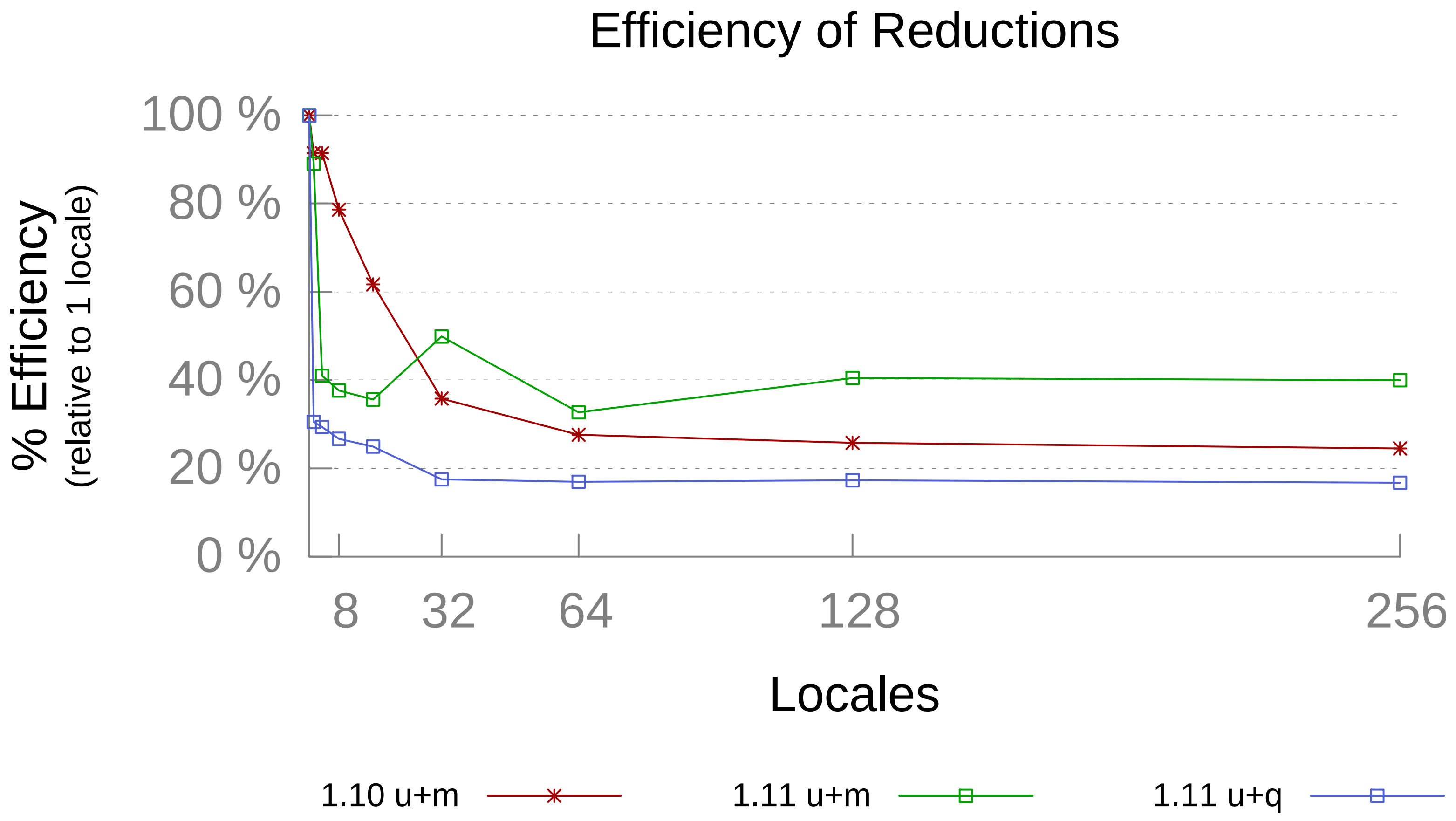


Scalability: RA Performance

- **for RA, ugni+muxed has not changed significantly**
  - More interesting is ugni+muxed vs. ugni+qthreads



Scalability: Reductions Efficiency



Performance Priorities and Next Steps

Performance Priorities and Next Steps

- **Continue to explore ugni+qthreads performance**
  - understand differences compared to ugni+muxed
    - if possible, close performance gap and retire muxed tasking
- **NUMA-aware performance**
  - more focus on NUMA locale model
    - particularly execution-time address representation
  - improve array initialization (parallel, appropriate first-touch)
    - currently gated by constructor/default init/noinit capabilities
  - explore the impact of using NUMA by default
- **Continue scalability studies**
  - Reduce unnecessary communication code
  - Improve implementation of reductions

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA, and YARCDATA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

Copyright 2014 Cray Inc.

CRAY CHAPEL

CRAY

THE SUPERCOMPUTER COMPANY

http://chapel.cray.com chapel_info@cray.com http://sourceforge.net/projects/chapel/