Performance Results

Chapel Team, Cray Inc.
Chapel version 1.10
October 2nd, 2014

COMPUTE | STORE | ANALYZE

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

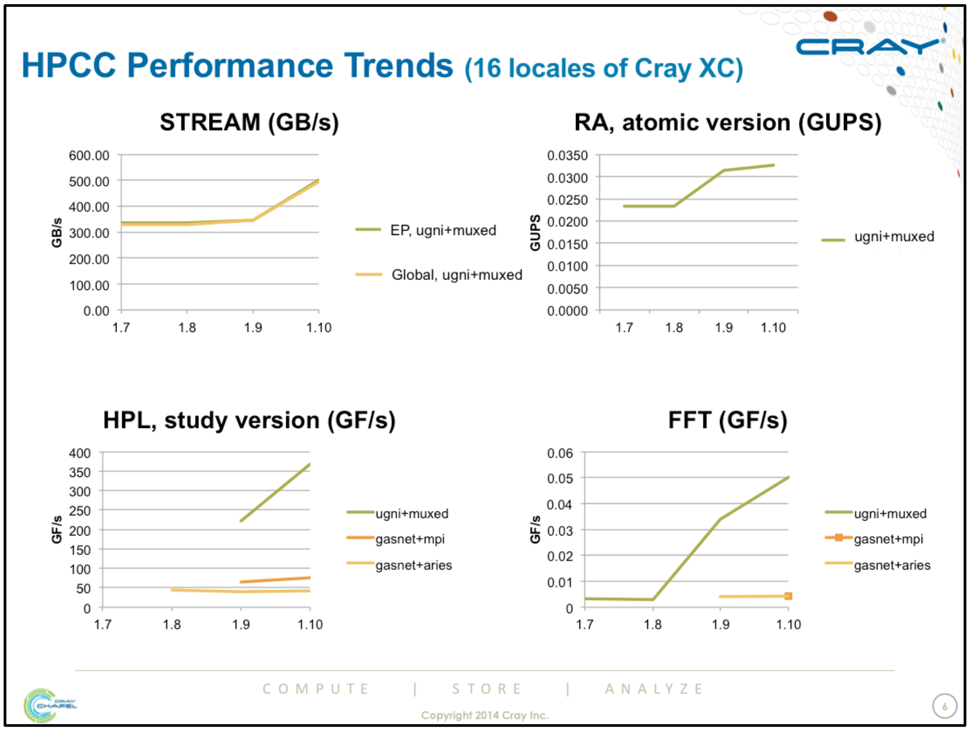
- **Generally speaking, performance has improved with 1.10**
- **Previous slide decks have shown performance changes:**

  ….in the shootout benchmarks
  …due to threading mechanism and policy changes
  …due to LICM improvements
  …due to the addition of CHPL_TARGET_ARCH/--specialize
  …due to generated code improvements
- **These slides contain additional v1.10 performance results**
  - not tied to any specific effort, just comparisons across releases

Outline

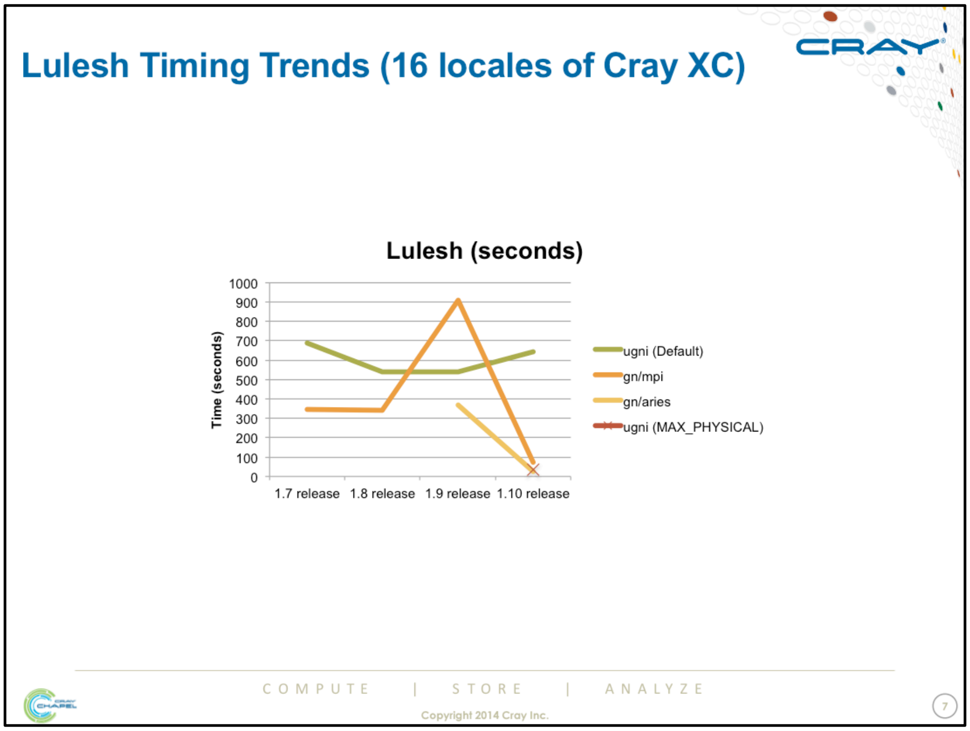
- Multi-locale Performance Trends
- Release-over-Release Nightly Testing Improvements
- Performance Scalability Study
- Performance Issue: Slurm/uGNI Conflict

Multi-locale Performance Trends



Higher is better in all these graphs.

Note that GASNet over MPI completes for the first time as of 1.10 for FFT.



This is our "study" version of LULESH, running with the sedov15oct input deck

gn = GASNet

Lower is better here.

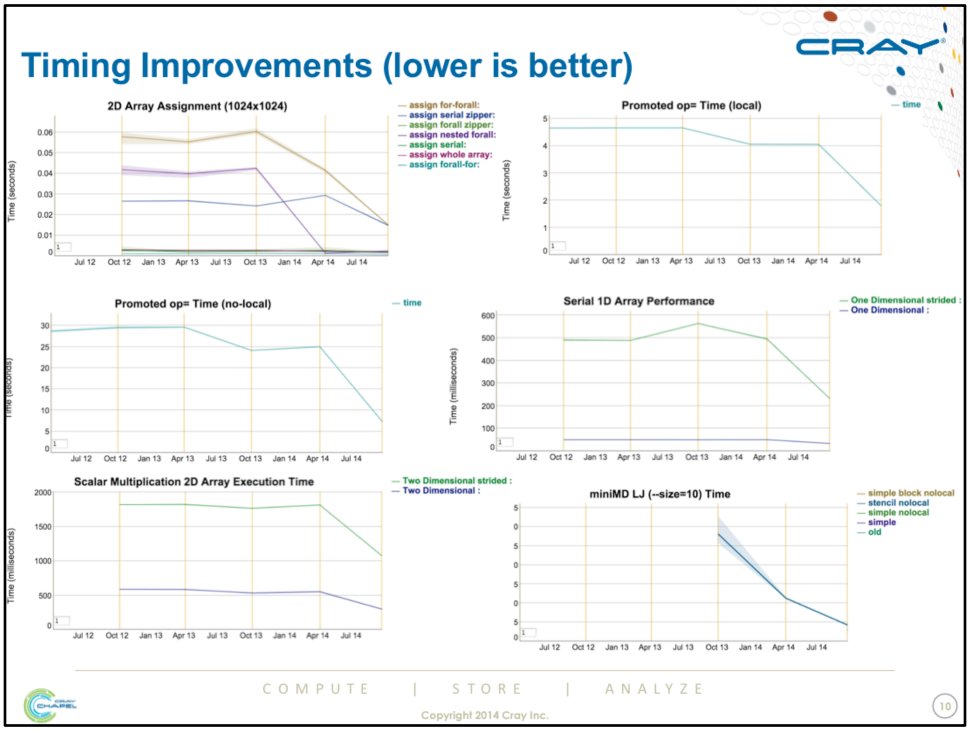
Note that LULESH strongly prefers using a number of threads equal to the number of physical cores for ugni/muxed (shown with the single 'X' point gathered with 1.10, which competes with the gasnet/aries results).

Release-over-Release Nightly Testing Improvements

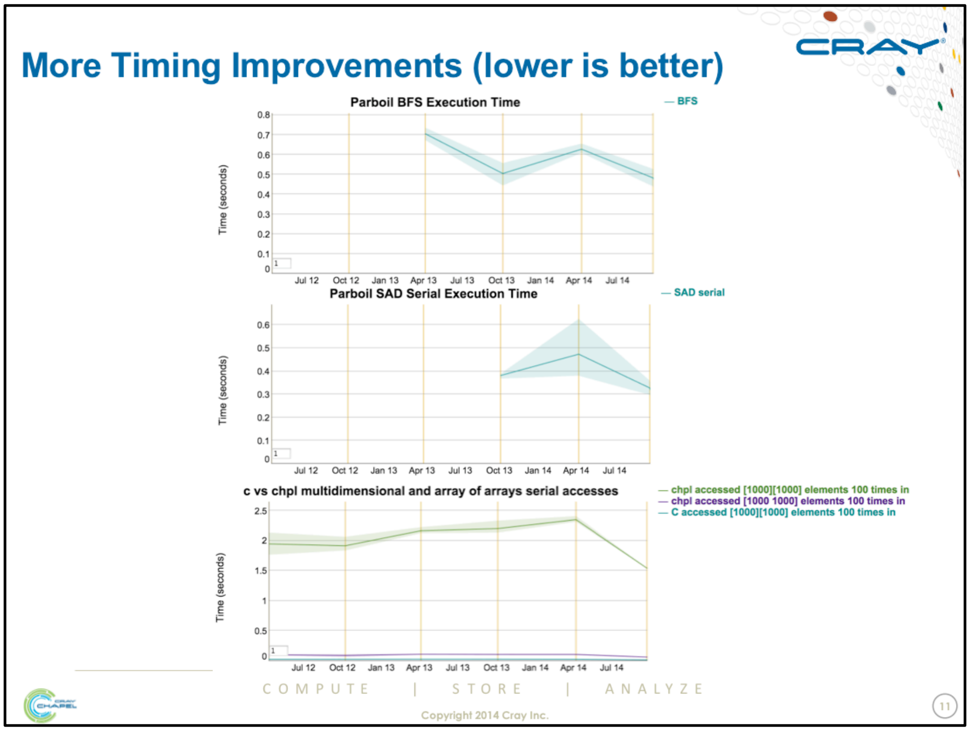
Release-over-Release Improvements

- **The following slides summarize release-over-release improvements across several releases**
  - today's benchmark codes
  - today's OS, gcc, etc.
  - retroactively run on past releases

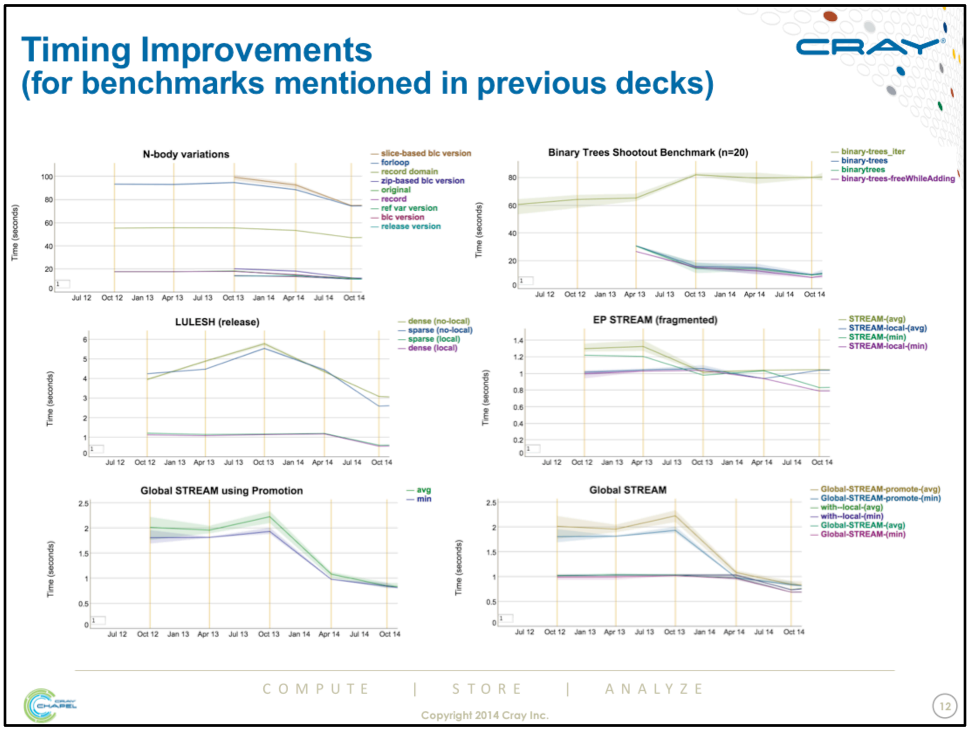
- **These were collected on chap04**
  - 64-bit Linux
  - 2 quad-core Intel Xeon processors with hyper-threading
    - 16 logical cores
  - 48 GB RAM
  - Can be viewed interactively at: http://chapel.sourceforge.net/perf/



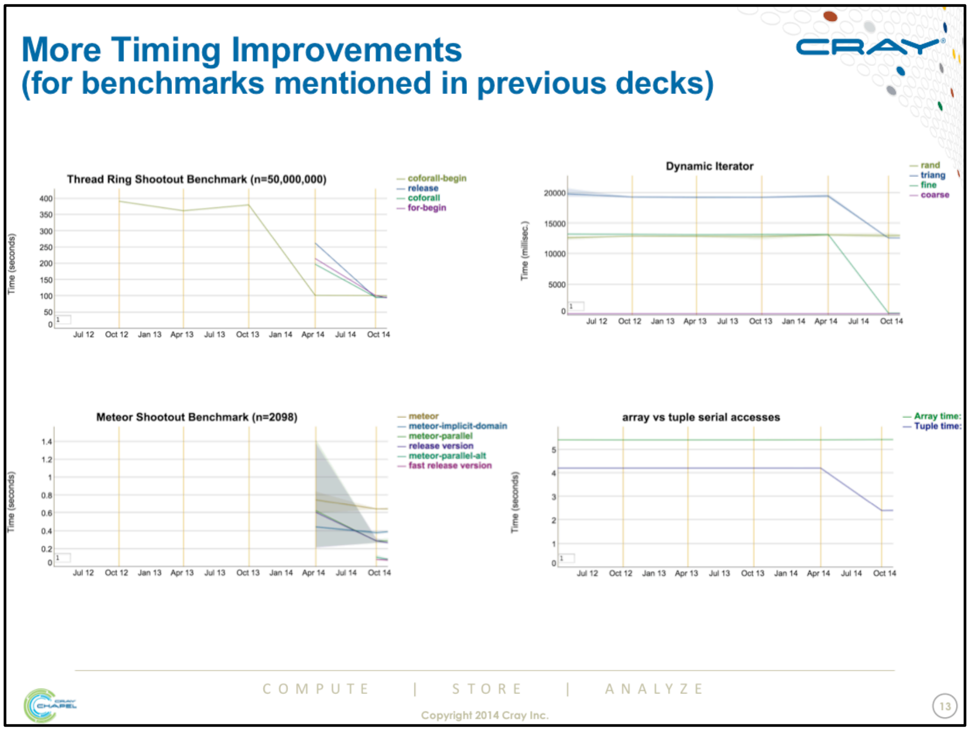
The vertical yellow lines correspond to the past several releases (every six months), with 1.10 being the rightmost one.



The vertical yellow lines correspond to the past several releases (every six months), with 1.10 being the rightmost one.



The vertical yellow lines correspond to the past several releases (every six months), with 1.10 being the rightmost one.



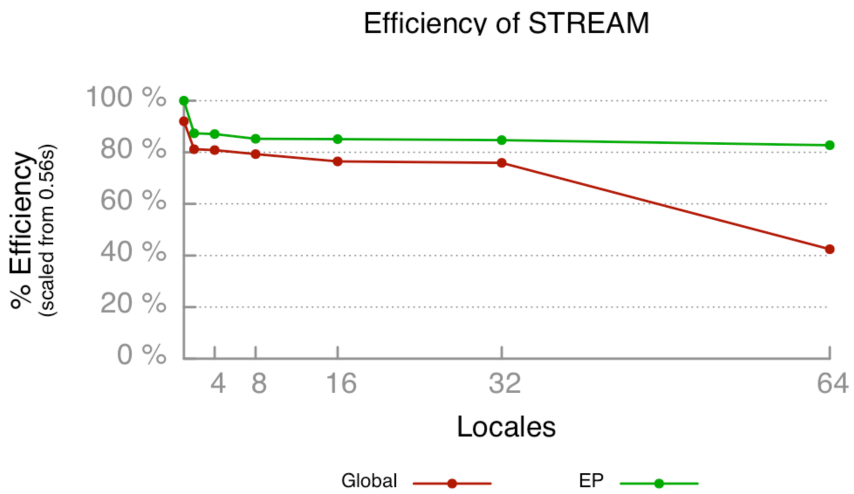
The vertical yellow lines correspond to the past several releases (every six months), with 1.10 being the rightmost one.

Performance Scalability Study

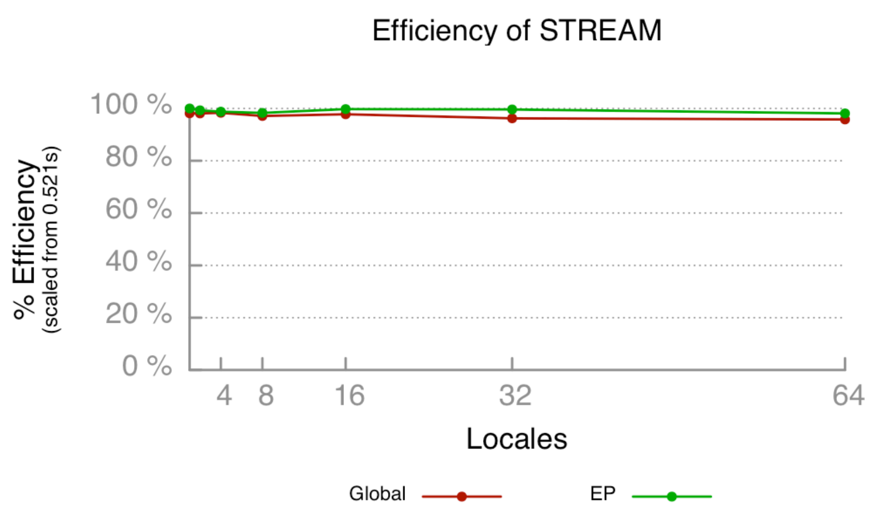
Scalability Study: Background

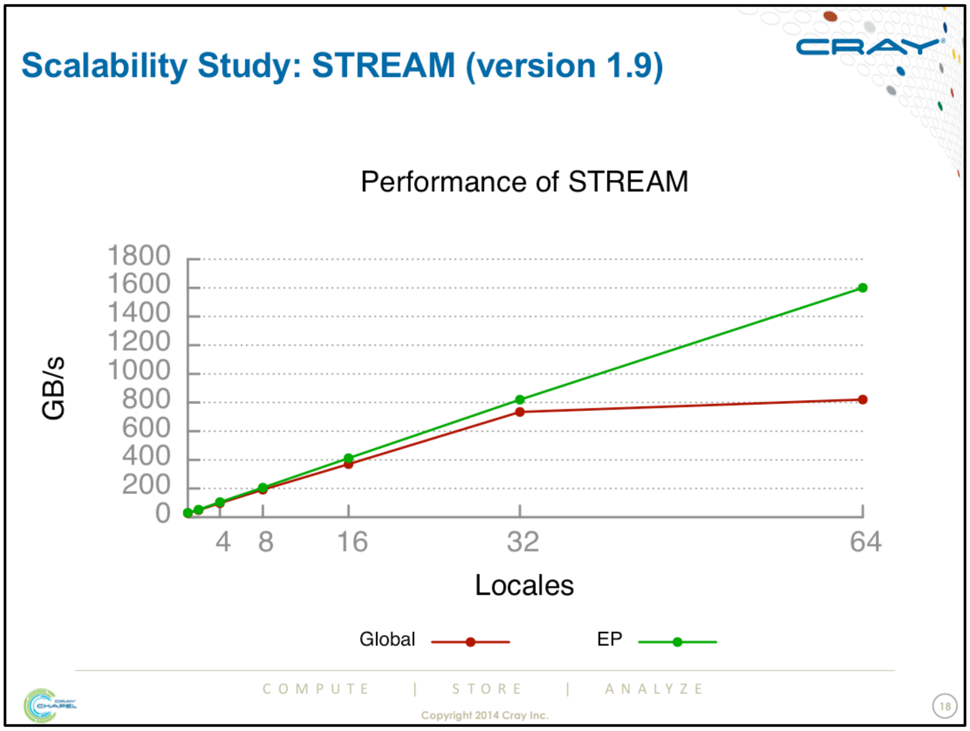
- **For this release, we performed a scalability study for some basic benchmarks**
  - HPCC Stream: EP and Global
  - HPCC RA: atomic, on-based, and remote memory operations (rmo)
    - these test network atomics, active messages, and puts/gets, respectively
  - Reduction of an array
- **All experiments shown here were performed on a Cray XC**
  - 1-64 locales
  - ugni+muxed runtime
- **Each of the following pairs of slides shows 1.9 vs. 1.10**
  - Note that some graphs are efficiency, some are performance
  - Also note that the scales of the graph change between versions
    - version 1.10 performs strictly better for these benchmarks on 1 locale

Scalability Study: STREAM (version 1.9)



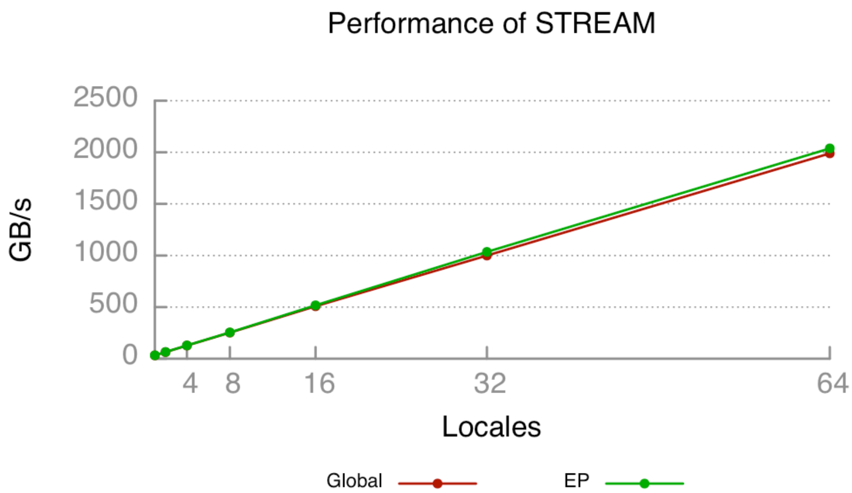
Scalability Study: STREAM (version 1.10)

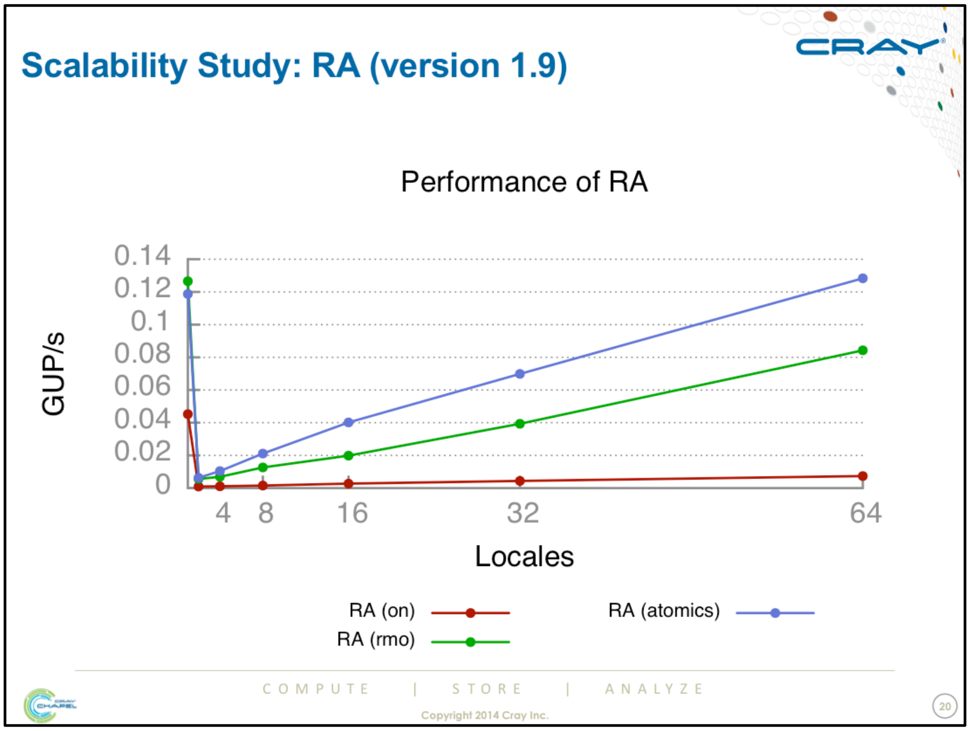




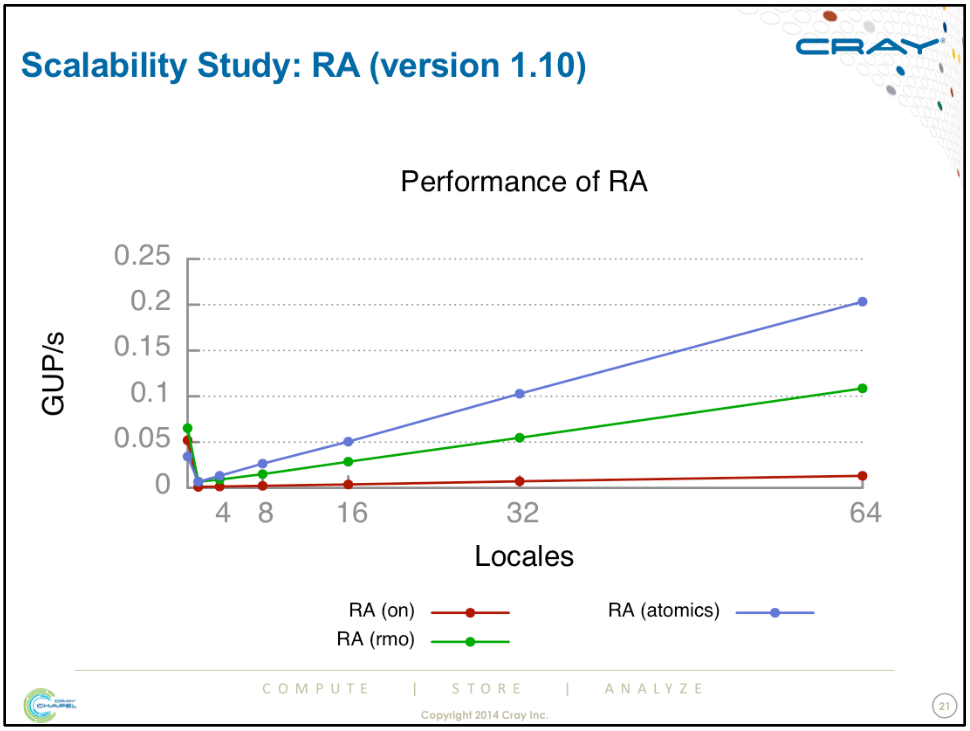
This pair of graphs is just a different way of looking at the data in the previous ones

Scalability Study: STREAM (version 1.10)

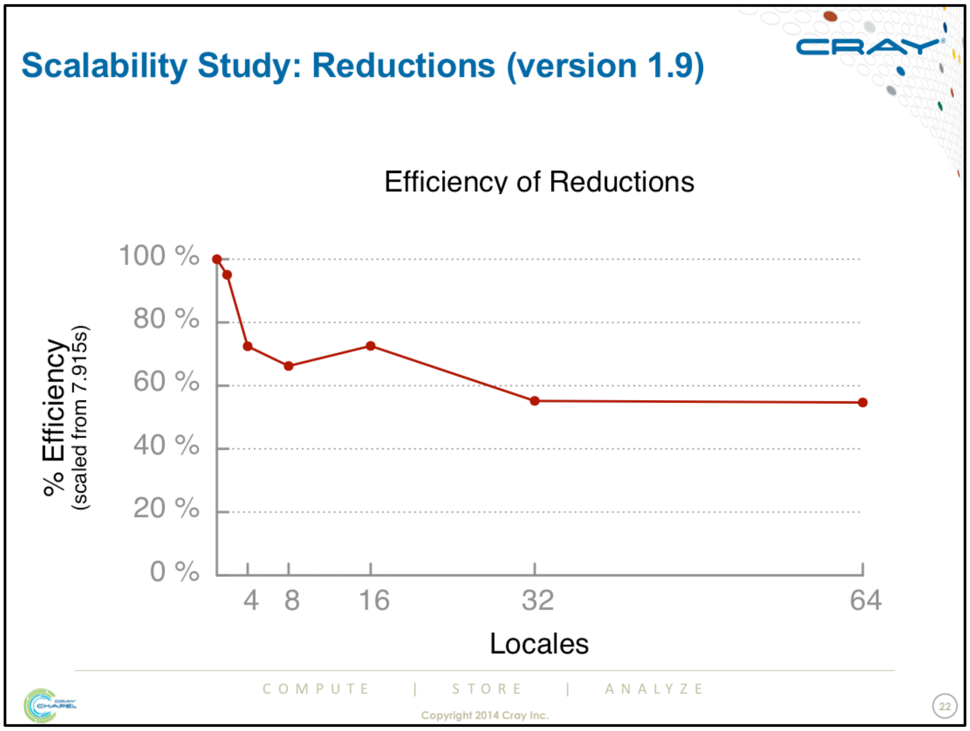




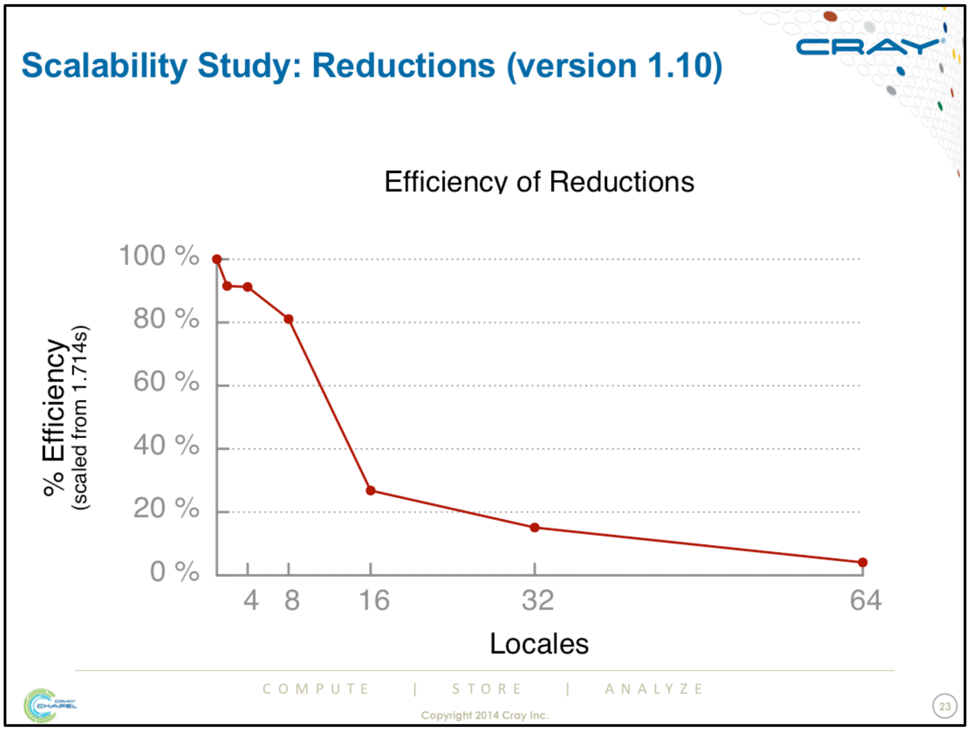
For RA, the drop in performance when moving from 1 to 2 locales is expected because you move from a world in which all updates are local to one in which half are remote. Note that our version does not do bucketing, so tends to do worse on 2 locales than the reference version.



Note that as of 1.10, the parallel performance crosses over the 1-locale performance at a much lower locale count than in 1.9



This test creates an array requiring ¼ of the available memory across all locales and measures the time required to compute a sum-reduce over it.



It's not clear to use yet what is making reductions scale worse in version 1.10. Interestingly, setting CHPL_RT_NUM_THREADS_PER_LOCALE='MAX_PHYSICAL' causes scalability to be much more similar to the 1.9 version.

Scalability Study: Next Steps

- **Scale studies to higher locale counts**
  - This study limited by availability/size of systems we could access
  - Have since gotten accounts on larger/more available external systems
- **Improve NUMA mapping for stream**
- **Investigate scalability limiters for on-based RA for ugni**
- **Tune reductions**
  - Investigate scalability challenges
  - Re-implement for performance
    - Ideally by moving implementation from compiler to modules
    - Approach would benefit from (ongoing) data parallel task intents work
- **Compare across more communication layers, benchmarks**

Performance Issue: Slurm/uGNI Conflict

Slurm/uGNI conflict

Background: Slurm limits CHPL_COMM=ugni throughput

- comm=ugni uses multiple uGNI communication domains (NIC cores)
  - default: # comm domains = # pthreads
  - minimizes thread contention, maximizes NIC throughput
  - program memory must be registered separately in each comm domain
- but slurm limits per-process Aries resources, to allow for node sharing
  - effectively limits total memory registration per process to about 240 GiB
- for example, cannot have 32 GiB heap and 8 comm domains

This Effort: Under slurm, throttle to fit within its limits

- reduce default heap size to 16 GiB (v. ALPS: nearly all free memory)
- given heap size, limit # comm domains so total registration < 240 GiB
  - 16 GiB: 15 comm domains; 32 GiB: 7 comm domains; etc.

Impact: May not reach full Aries performance with slurm

Next Steps: Requires network-exclusive mode from Slurm

Performance Priorities and Next Steps

Performance Priorities and Next Steps

- **consider ugni/muxed as default runtime on Crays**
  - understand/improve cases like RA using ons
  - improve coverage, reporting, and graphing of automated testing
  - investigate support for ugni with qthreads and/or gasnet with muxed
- **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
  - consider using NUMA by default (?)
- **SIMD-ization of generated code**
- **Continue scalability studies**
  - Reduce unnecessary communication
  - Improve implementation of reductions

COMPUTE | STORE | ANALYZE

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.

CRAY CHAPEL

CRAY

THE SUPERCOMPUTER COMPANY

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