Multilocale Performance Trends

Chapel Team, Cray Inc.
Chapel version 1.9 summary
April 17th, 2014 (released) / May 2014 (documented)

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.

Copyright 2014 Cray Inc.

Caveats for this slide deck

- **As noted earlier, our recent focus has been primarily on single-locale performance**
  - to date, we have not tracked multi-locale performance automatically
  - thus, much of the data in this deck was generated for these slides
    - ⇒ an attempt to retroactively understand the past six months of progress
    - ⇒ not as much ongoing analysis and tracking effort as single-locale cases
    - ⇒ not all performance changes are understood at present
- **Work is currently underway to enable multi-locale testing**
  - this should ease much of the above pain next time around
- **Multi-locale performance has improved in v1.9 in spite of lack of focused attention recently**

Copyright 2014 Cray Inc.

Multilocale Performance on Cray XC

Benchmarks:
- HPC Challenge: Stream, RA, FFT, HPL, PTRANS
- NAS Parallel Benchmarks: EP
- SSCA#2

System: Cray-internal Cray XC “tiger”
- 44 compute nodes total, of various types
  - here, 16 homogeneous nodes were used for performance studies
    - Sandy Bridge processors, 16 cores, 2.7GHz, 32GB memory
- in most cases, reported the best of 6 or more tries

Runtime Layers:
- ugni (+ muxed): native runtime layer developed for Cray systems
- GASNet/aries (+ fifo): GASNet’s conduit, tuned for Cray systems
- GASNet/mpi (+ fifo): GASNet’s portability conduit

COMPUTE | STORE | ANALYZE

Copyright 2014 Cray Inc.

For each benchmark, we present major contributions to performance improvements.

Most graphs plot performance – bigger is better. Exceptions as indicated.

Note that the ungi + muxed option is only available within the Chapel module, not part of the open-source release

Note that GASNet+mpi is not a combination that one would want to use for performance reasons; it exists primarily as a robust but slow portability option. We present results for it here simply as a point of comparison (the other options should outperform it)

Two types of graphs in this presentation

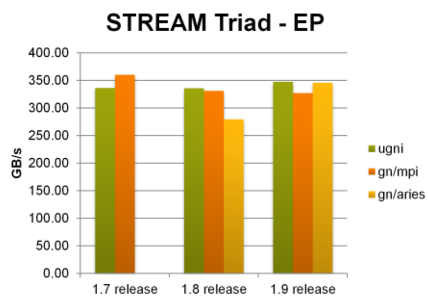
1. **Historical:**
   - shows performance for each release, for the benchmark at that time
     - in most cases, we've gathered these numbers retroactively and recently
   - tracks improvements to both the benchmark and to Chapel
   - these benchmarks are unchanging, so this is equivalent to the release-over-release graphs from the single-locale runs

2. **Timeline:**
   - shows performance relative to revision number
   - was gathered via sampling, so may not tell the whole story

Note that, unlike the single-locale graphs previously, these are generally performance graphs (higher is better)

Copyright 2014 Cray Inc.

HPCC STREAM EP: historical

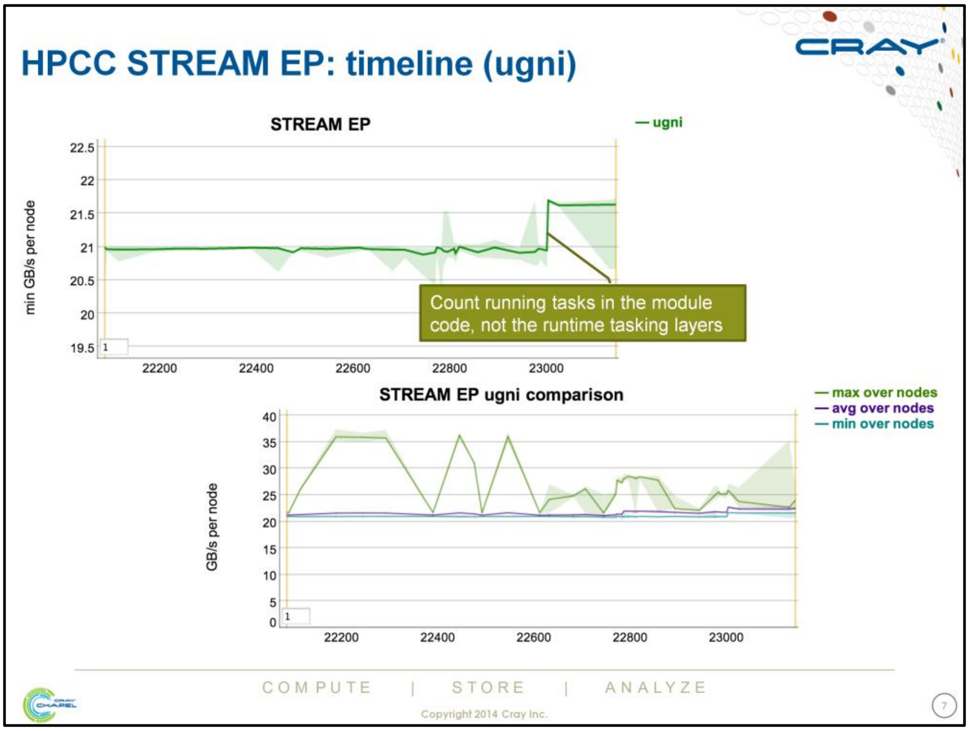


COMPUTE | STORE | ANALYZE

Copyright 2014 Cray Inc.

test/release/examples/benchmarks/hpcc/stream-ep.chpl

m=357,739,200 per node



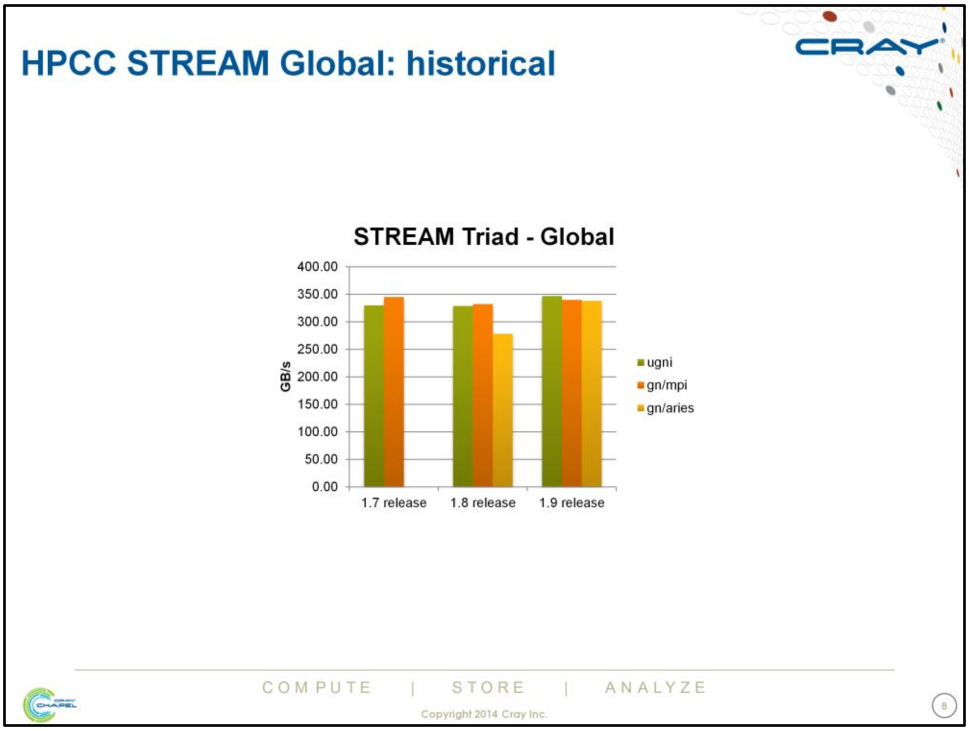
The **first graph**: for each run, pick the performance of the slowest node. Then, take the average over 6 runs. Shaded areas show the spread over 6 runs.

The improvement is: 23004 Count running tasks in the module code, not the runtime tasking layers (Greg).

The **second graph**: for each run, pick the fastest node (highest line), average node (middle line), slowest node (lowest line). We believe NUMA effects make individual node performance very variable. We ignore the best-case performance because it is not reproducible.

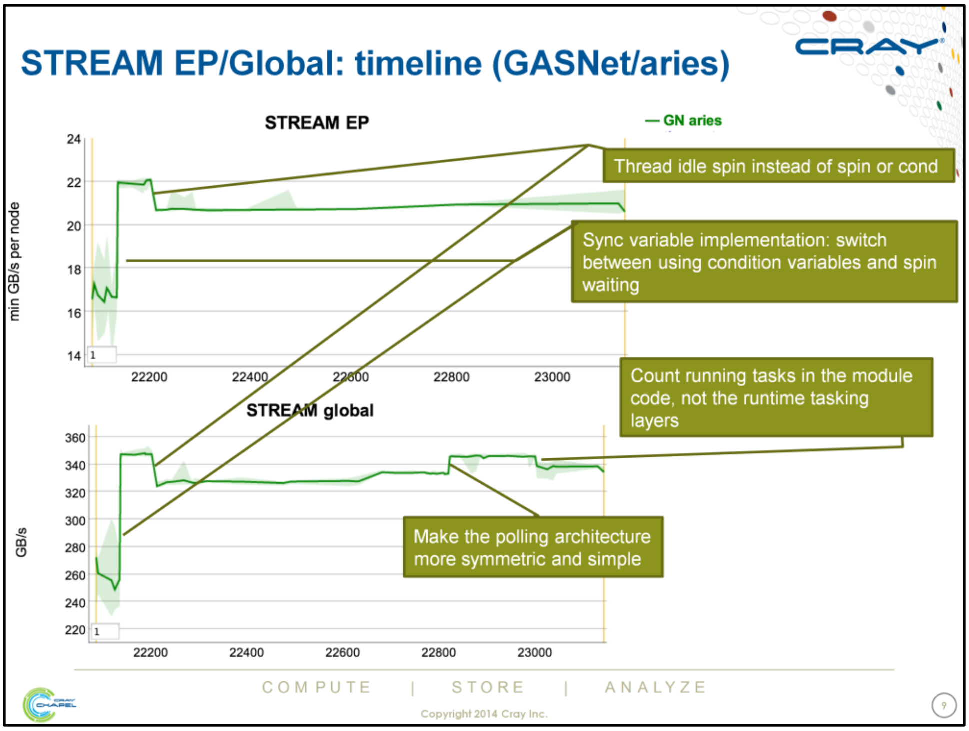
Yellow vertical bars indicate Releases 1.8 and 1.9. X axis shows revision numbers.

Count running tasks in the module code, not the runtime tasking layers: r23004.



test/release/examples/benchmarks/hpcc/stream.chpl

m=357,739,200 per node, 5.7G total



The first graph measures per-node performance – the slowest node in a run is taken. The second graph measures the whole-system performance (16 nodes).

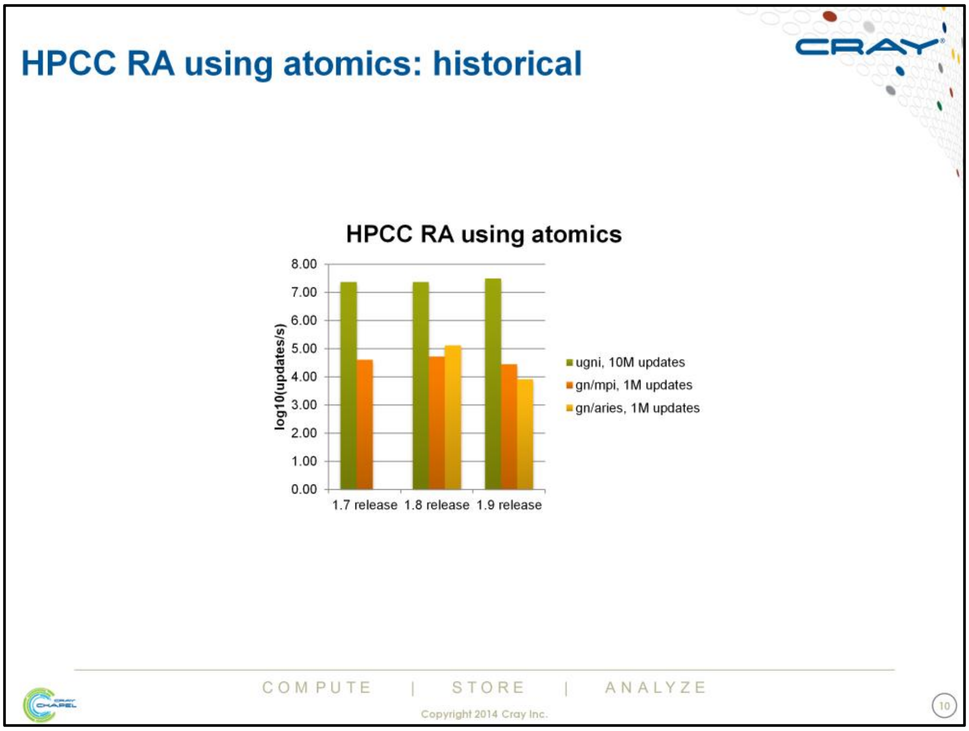
Yellow vertical bars indicate Releases 1.8 and 1.9. X axis shows revision numbers.

Sync variable implementation: switch between using condition variables and spin waiting: r22137.

Thread idle spin instead of spin or cond: r22210.

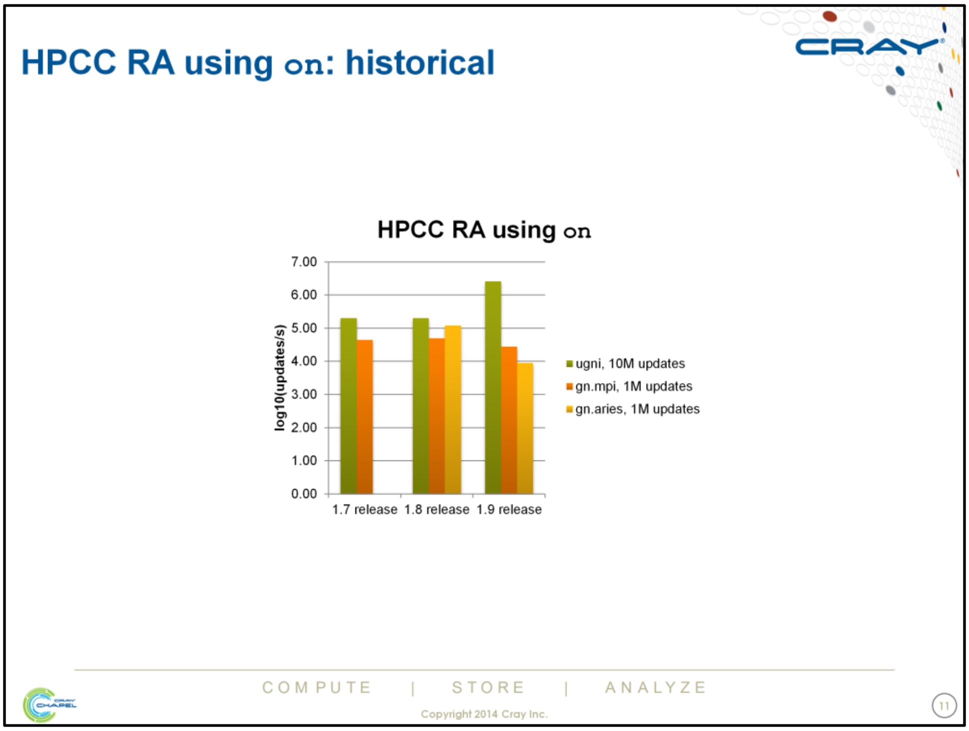
Make the polling architecture more symmetric and simple: r22823.

Count running tasks in the module code, not the runtime tasking layers: r23004.



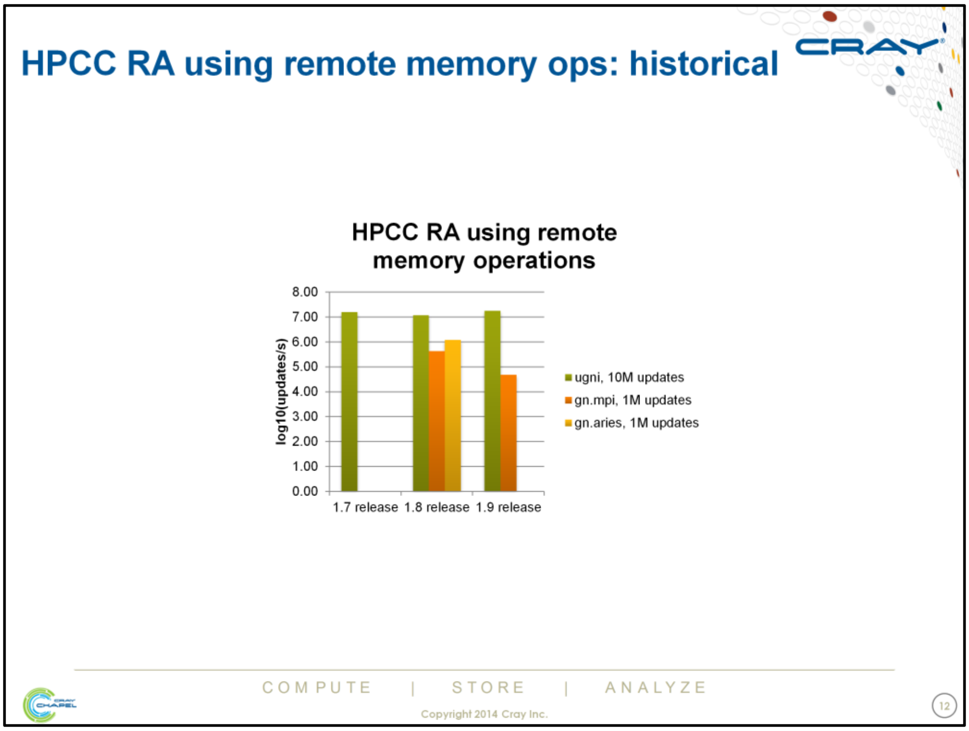
test/release/examples/benchmarks/hpcc/ra-atomics.chpl

n=8,589,934,592 (2^33), 10M updates (ugni), 1M updates (GASNet/mpi, GASNet/aries)



test/release/examples/benchmarks/hpcc/ra.chpl useOn=true

n=8,589,934,592 (2^33), 10M updates (ugni), 1M updates (GASNet/mpi, GASNet/aries)



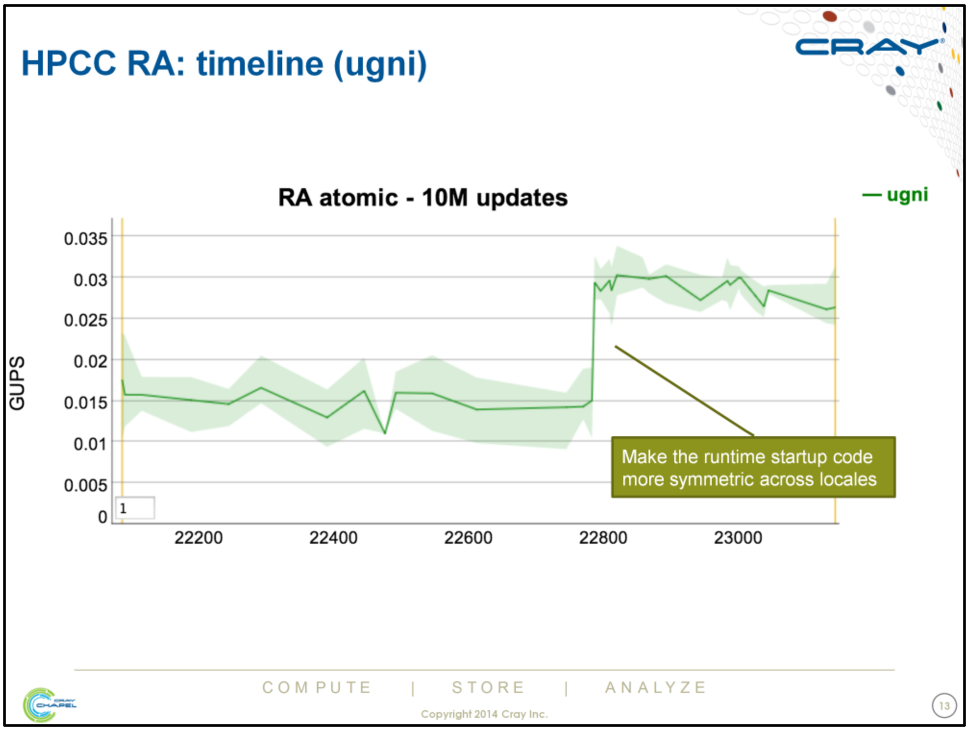
test/release/examples/benchmarks/hpcc/ra.chpl useOn=false

n=8,589,934,592 (2^33), 10M updates (ugni), 1M updates (GASNet/mpi, GASNet/aries)

GASNet/aries was not available for 1.7

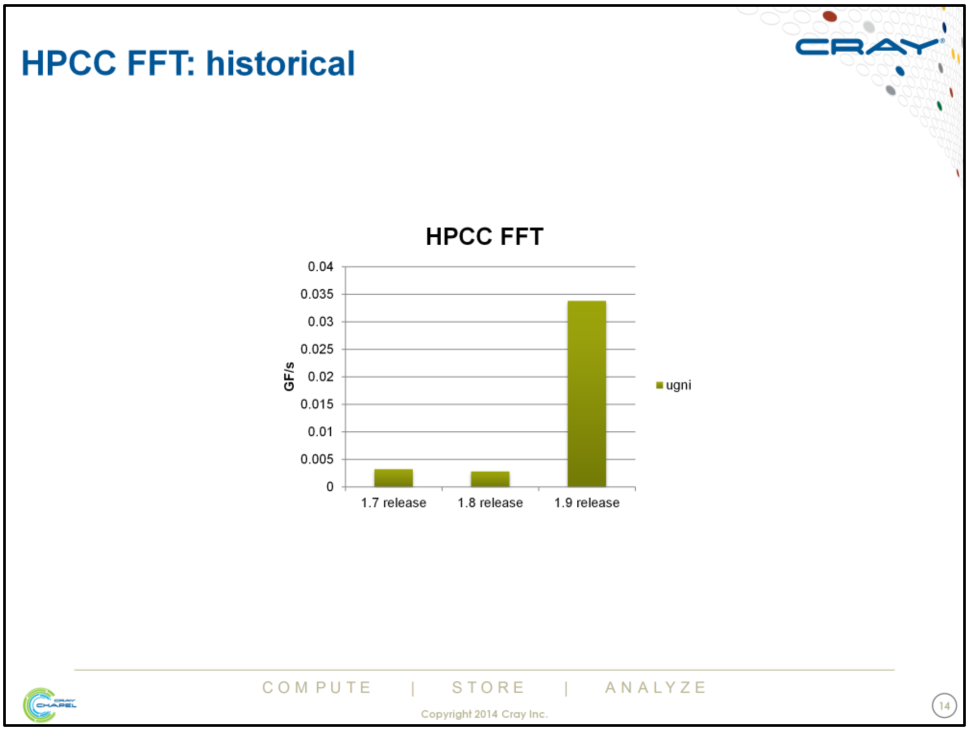
RA under GASNet/mpi times out for 1.7

RA under GASNet/aries crashes at run time for 1.9



Yellow vertical bars indicate Releases 1.8 and 1.9. X axis shows revision numbers.

Make the runtime startup code more symmetric across locales: r22787, r22809, r22813.



test/release/examples/benchmarks/hpcc/fft.chpl

n=4,194,304 (2^22)

HPCC HPL: historical (release version)

- **uses expensive high-level array/domain operations**
  - e.g. extensive use of slices
- **ugni status:**
  - v1.8 times out, v1.9 does not
  - Improvements due to
    - Making the runtime startup code more symmetric across locales
    - Reducing the number of formal temps inserted by the compiler
- **GASNet/mpi status**
  - all versions time out
- **GASNet/aries**
  - 3x improvement v1.8 → v1.9
  - Improvements due to:
    - Hybrid sync variable implementation
    - Switched to GASNet 1.22.0
    - Reduce the number of formal temps inserted by the compiler

COMPUTE | STORE | ANALYZE

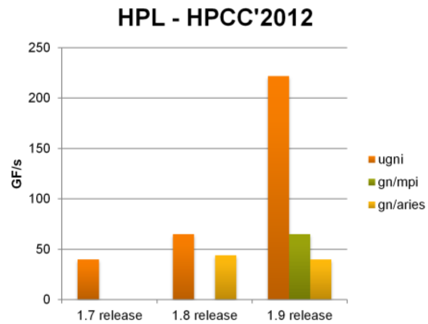
Copyright 2014 Cray Inc.

test/release/examples/benchmarks/hpcc/hpl.chpl

n=1023, nb=32

The formal temp reduction change was r22900

HPCC HPL: historical (HPCC'2012 version)



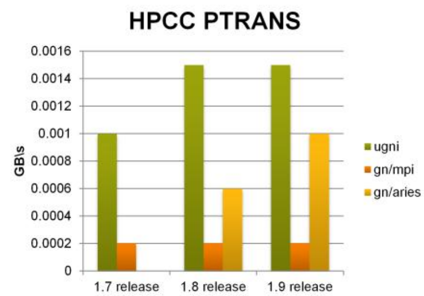
- **uses a low-level coding style**
- **ugni improvements due to:**
  - Making the runtime startup code more symmetric across locales
  - Replacing trivial record assignments with verbatim copies
- **GASNet/mpi improvements due to:**
  - Replacing trivial record assignments with verbatim copies
  - Counting running tasks in the module code, not in the runtime

Copyright 2014 Cray Inc.

test/studies/hpcc/HPL/vass/hpl.hpcc2012.chpl

n=31,999, nb=200

HPCC PTRANS: historical



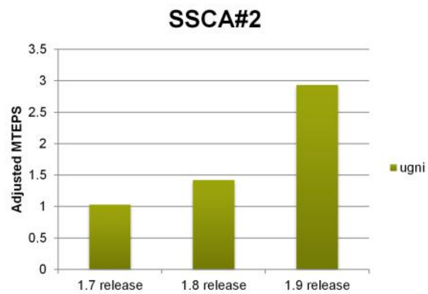
- **GASNet/aries improvements due to:**
  - Switching to GASNet 1.22.0

Copyright 2014 Cray Inc.

test/release/examples/benchmarks/hpcc/ptrans.chpl

n=2,000; nb=100

SSCA#2: historical



- GASNet/mpi, aries not reported due to execution time > 10 minutes

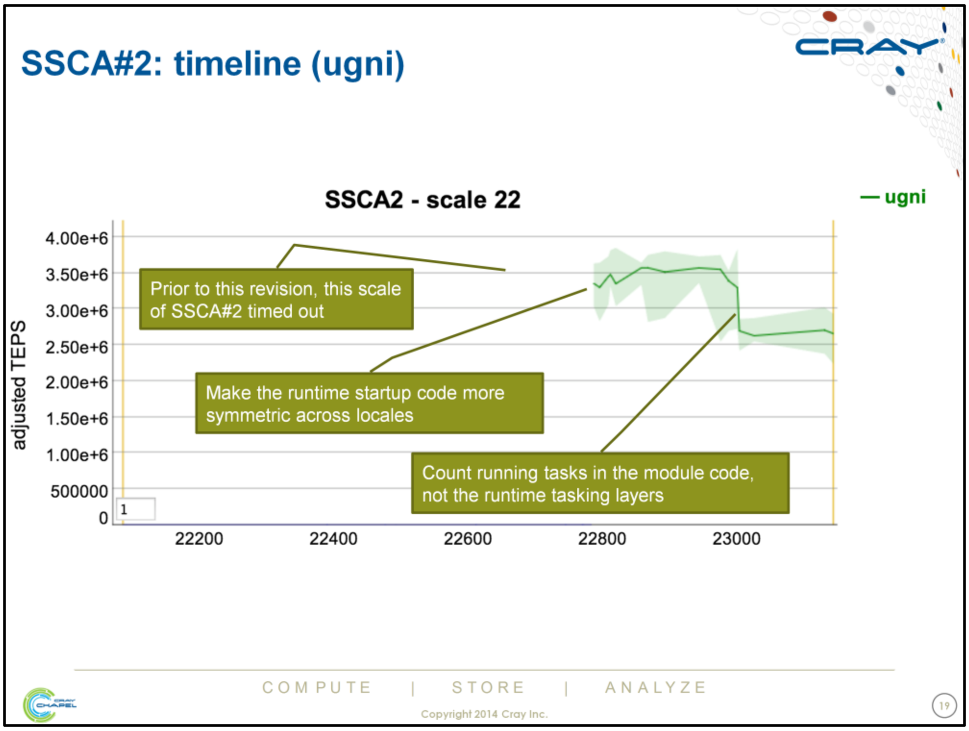
Copyright 2014 Cray Inc.

test/release/examples/benchmarks/ssca2/SSCA2_main.chpl

scale=4,194,304 (2^22), start verts=64 (2^6)

for v1.9, start verts=16

We also have performance data for toy problem sizes. We are not reporting them here because they are not significant.

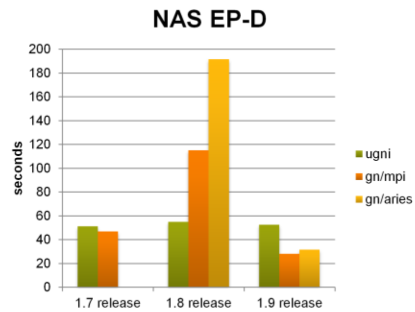


Yellow vertical bars indicate Releases 1.8 and 1.9. X axis shows revision numbers.

Make the runtime startup code more symmetric across locales: r22787, r22809, r22813.

Count running tasks in the module code, not the runtime tasking layers: r23004.

NPB EP: historical



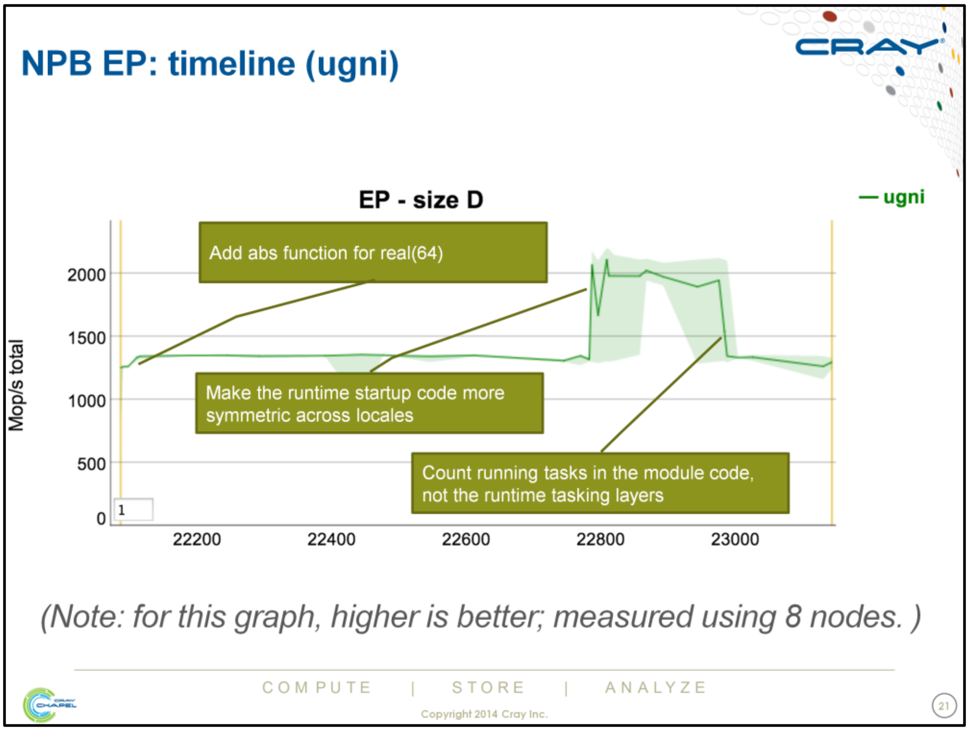
(Note: for this graph, lower bars are better)

- **GASNet/aries improvements due to:**
  - Hybrid sync variable implementation
- **GASNet/mpi improvements due to the above, plus:**
  - Reduced sublocale overheads at execution time for fifo tasks

Copyright 2014 Cray Inc.

test/npb/ep/mcahir/ep.chpl

Class D



Large spread makes precise analysis difficult.

Yellow vertical bars indicate Releases 1.8 and 1.9. X axis shows revision numbers.

Add abs function for real(64): r22102.

Make the runtime startup code more symmetric across locales: r22787, r22809, r22813.

Count running tasks in the module code, not the runtime tasking layers: r23004.

Multilocale Performance Impacts (part 1)

The most influential commits, in chronological order.

r22137: Sync variable implementation

- STREAM (ep+g), HPL (r) – aries
- NPB EP – mpi, aries

r22150..r22160: switch to GASNet 1.22.0

- HPL (r), PTRANS - aries

r22210: Thread idle spin instead of spin|cond

- slowdown STREAM (ep+g), HPL (r) – aries

r22787, r22809, r22813: Make the runtime startup code more symmetric across locales

- STREAM (g), RA (a+o+r), FFT, HPL (r+hpcc12), SSCA#2 - ugni

Copyright 2014 Cray Inc.

Multilocale Performance Impacts (part 2)

r22845+r22846: replace trivial record assignments with verbatim copies
- HPL (hpcc12) – ugni, mpi

r22900: Reduce the number of (unnecessary) formal temps inserted by the compiler
- HPL (r) – ugni, aries

r23004: Count running tasks in the module code, not the runtime tasking layers
- STREAM (ep+g) – ugni
- slowdown SSCA#2 (ugni)
- HPL (hpcc12) – mpi
- FFT - aries

Copyright 2014 Cray Inc.

Multilocale Next Steps

- **Enable regular (nightly/weekly) multilocale testing**
  - using a Cray module built against trunk/HEAD
  - correctness testing
  - performance testing
    - goal: automatically generate historical record as in single-locale world
  - possibly on a cluster as well (resources permitting)

- **Track dynamic communications as performance tests**
  - benefit: more stable/reproducible than timings for most benchmarks
  - currently only done via correctness testing
    - makes it difficult to get long-term sense of how things are trending

Copyright 2014 Cray Inc.

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/