Performance Optimizations

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

COMPUTE | STORE | ANALYZE

Safe Harbor Statement

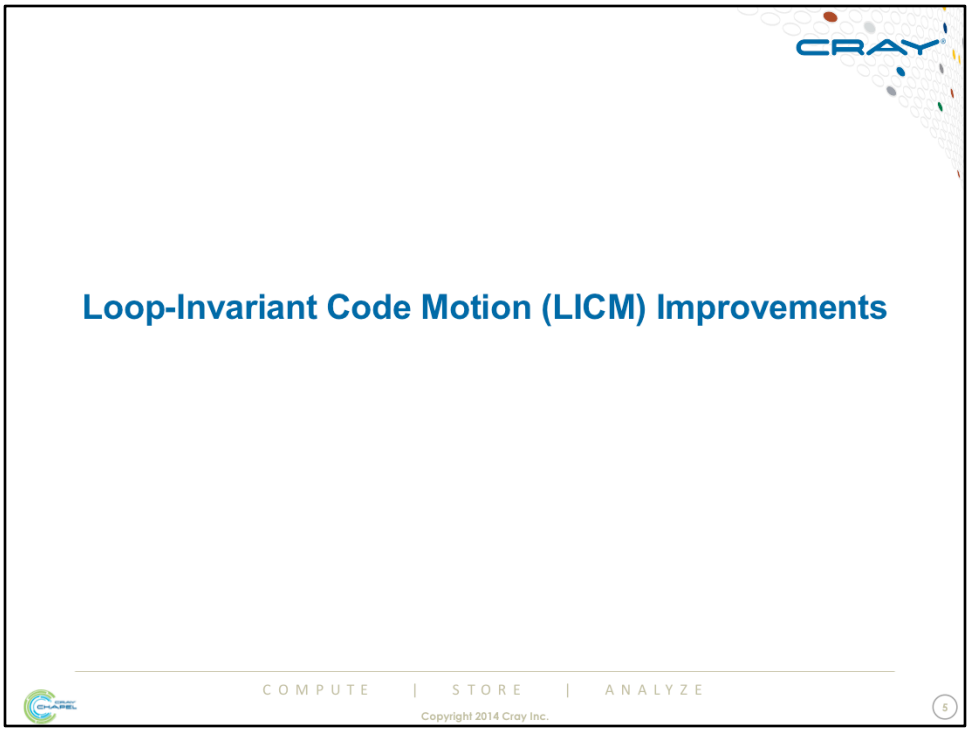
This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction 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.

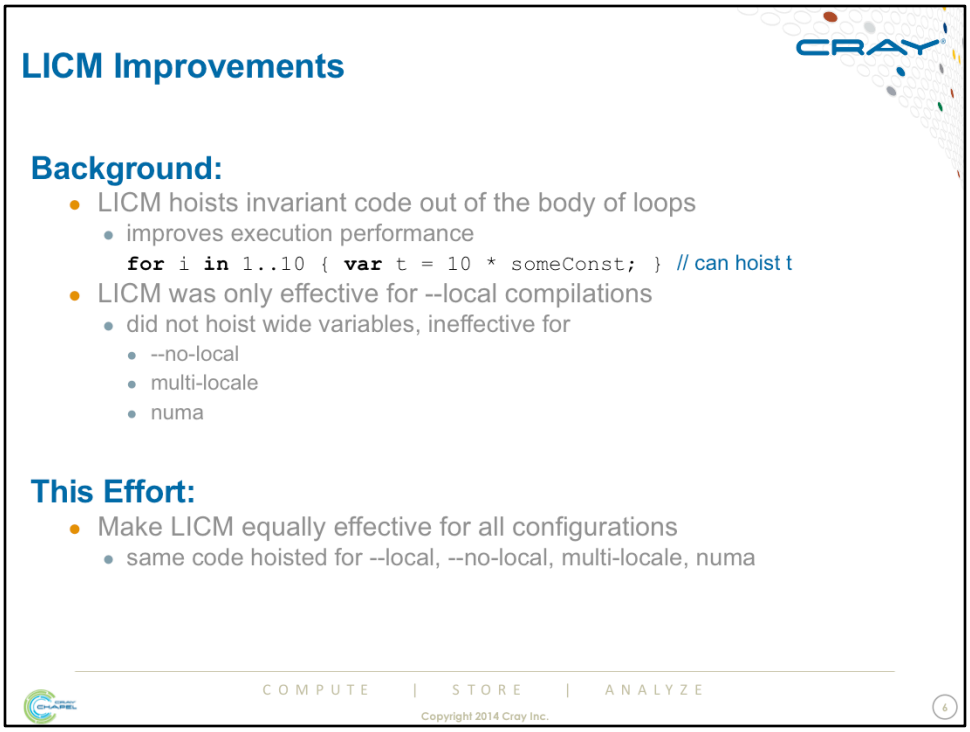
Executive Summary

- **Performance optimizations did not receive a large effort in this release cycle**
  - Some changes that we expected to benefit performance did not
    - e.g., improvements to generated code (see later deck)
  - Yet, a few focused changes had a significant impact in key cases

Outline

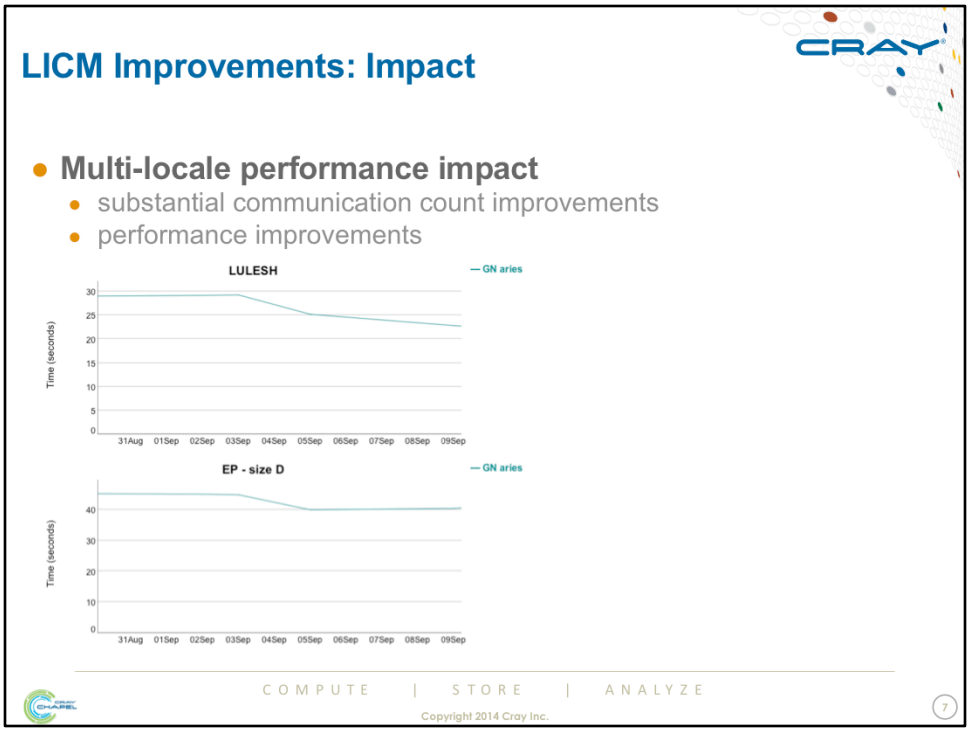
- Loop-Invariant Code Motion (LICM) Improvements
- CHPL_TARGET_ARCH and --specialize
- Other Performance Optimizations


Loop-Invariant Code Motion (LICM) Improvements
COMPUTE | STORE | ANALYZE
Copyright 2014 Cray Inc.
5



Checks in loop invariant code motion accidently prevented wide variables form being hoisted, which meant LICM was largely ineffective for everything except for --local.

To fix this, the check that prevented hoisting of wide variables was removed, and two latent bugs were discovered and fixed.



There were some substantial communication (comm) count improvements. Most notably tests in test/performance/ferguson. These test the comm counts for some basic chapel communication idioms:

```
remote-array-read   : #gets 200011 => 100012
remote-array-write  : #gets 100011 => 12
remote-class-read   : #gets 700011 => 600012
remote-class-write  : #gets 400011 => 300012
remote-record-read  : #gets 400011 => 300012
remote-record-write : #gets 100011 => 12
remote-tuple-read   : #gets 400011 => 300012
remote-tuple-write  : #gets 100011 => 12
```

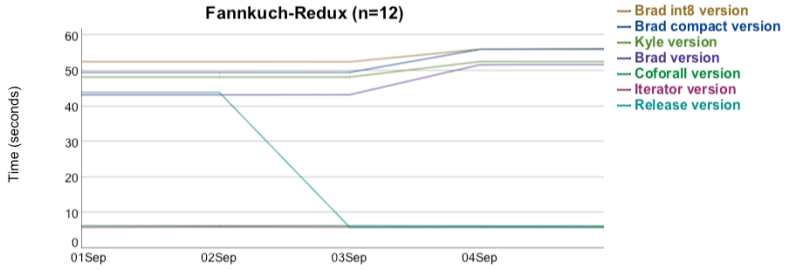
```
There were a few other improvements in real tests such as SSCA

ssca2 kernel #3     : #gets 239 => 158
```

```
There would have been more comm count improvements but LICM is
not very effective with bounds-checks on, and tests in test/
```
performance/ferguson, SSCA, and the other improvements happen to throw –fast

LICM Improvements: Impact (continued)

- **Single locale performance impact**
  - some variations of Fannkuch took a performance hit
    - result of a latent bug that was discovered and fixed as part of this work
    - did not affect the fastest versions



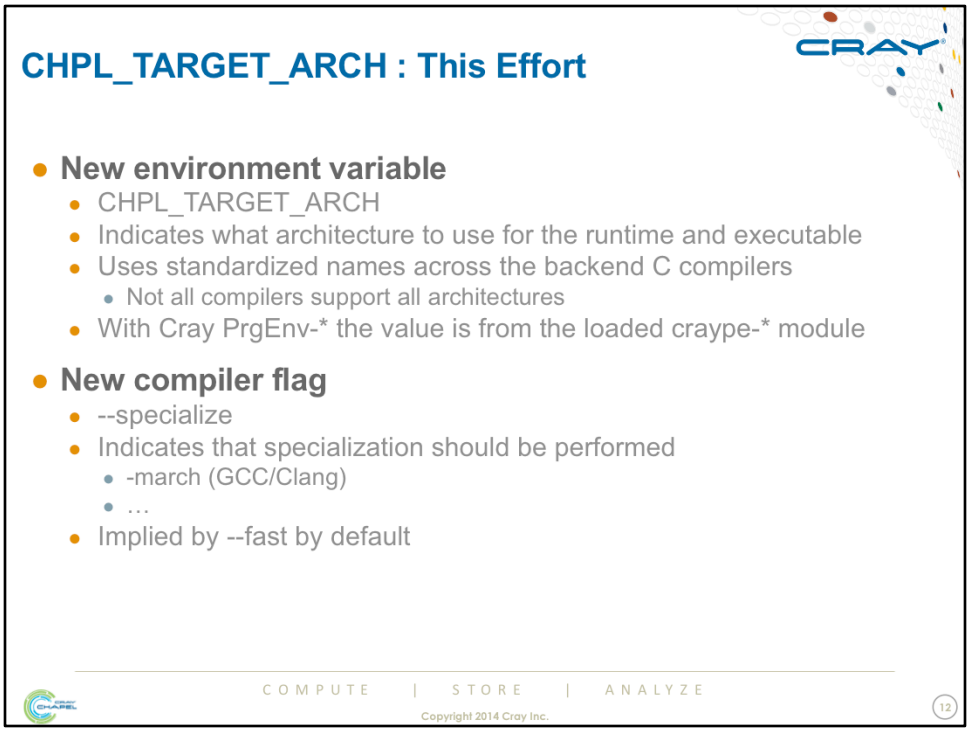
LICM Improvements: Next Steps

- **Fix Fannkuch performance regression**
  - source of regression has been identified
  - fix would have come too late in the release cycle
    - was not a high priority since fastest versions were unaffected

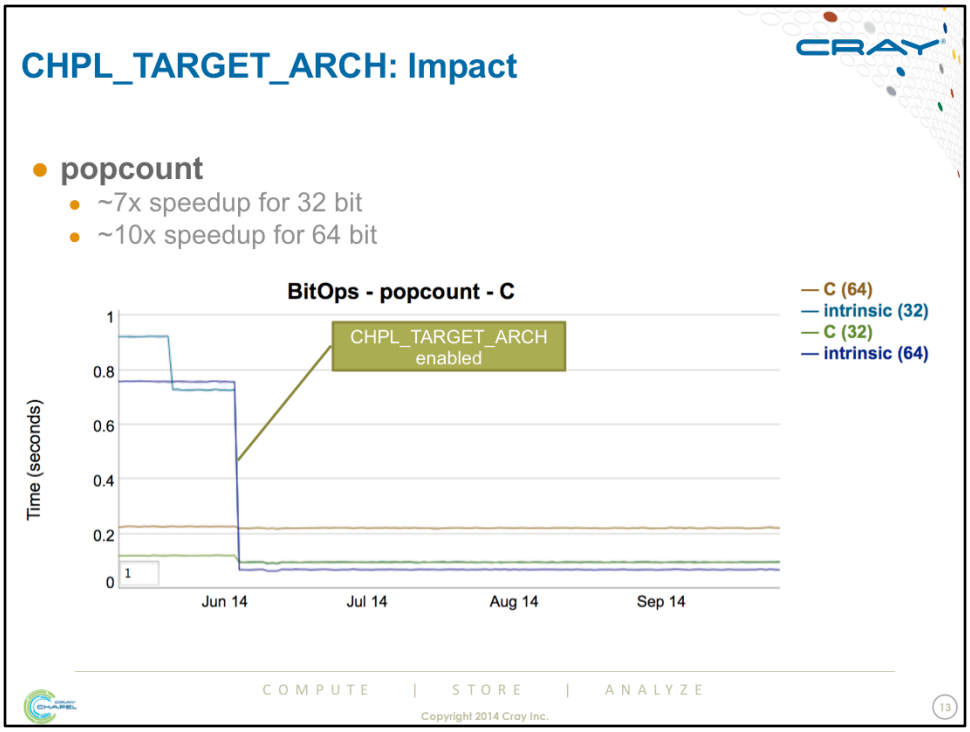

CHPL_TARGET_ARCH and --specialize
COMPUTE | STORE | ANALYZE
Copyright 2014 Cray Inc.
10

CHPL_TARGET_ARCH : Background

- **C compilers can optimize for the target architecture**
  - We needed some way to expose this in Chapel
- **Some operations based on intrinsics are slow**
  - By default, C compilers won't emit instructions added by recent ISAs
- **A target architecture is required to get good vectorization**
  - No AVX or SSE newer than SSE2 without specifying the architecture



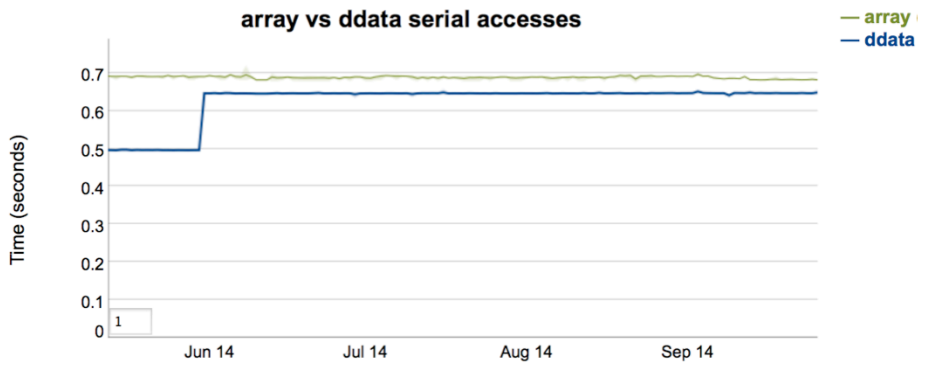
See README.chplenv for more details on the supported options for CHPL_TARGET_ARCH



GCC's __builtin_popcount is slower than the straight C version without allowing specialization.

CHPL_TARGET_ARCH: Impact

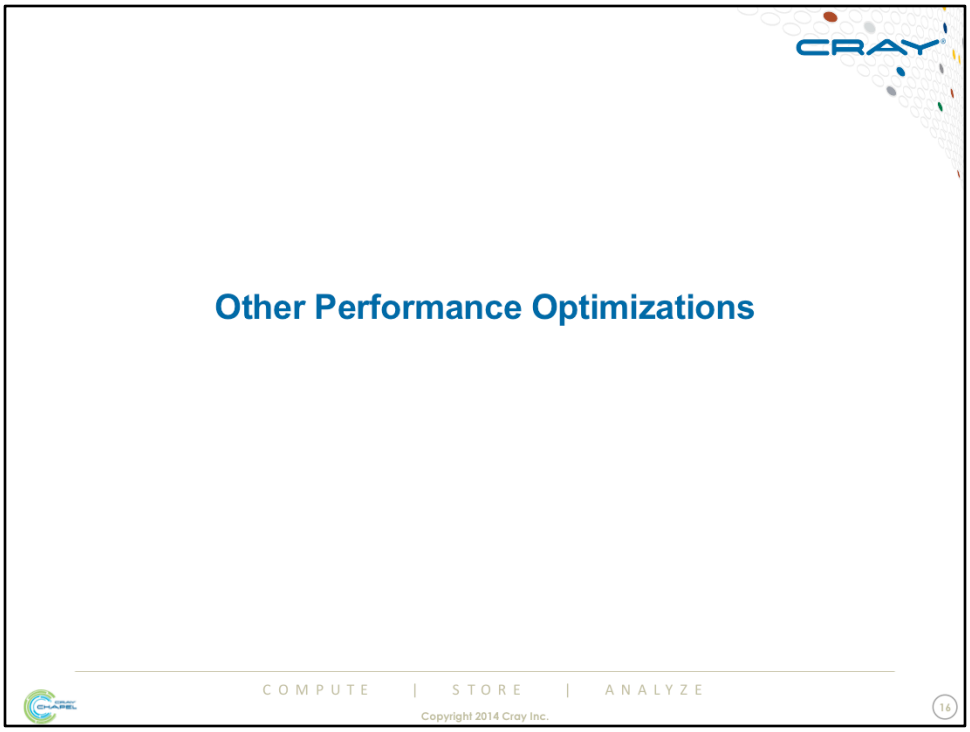
- **array vs ddata serial accesses**
  - ~1.4x slowdown for ddata
  - Not clear why this happens



Little to no impact on other tests.

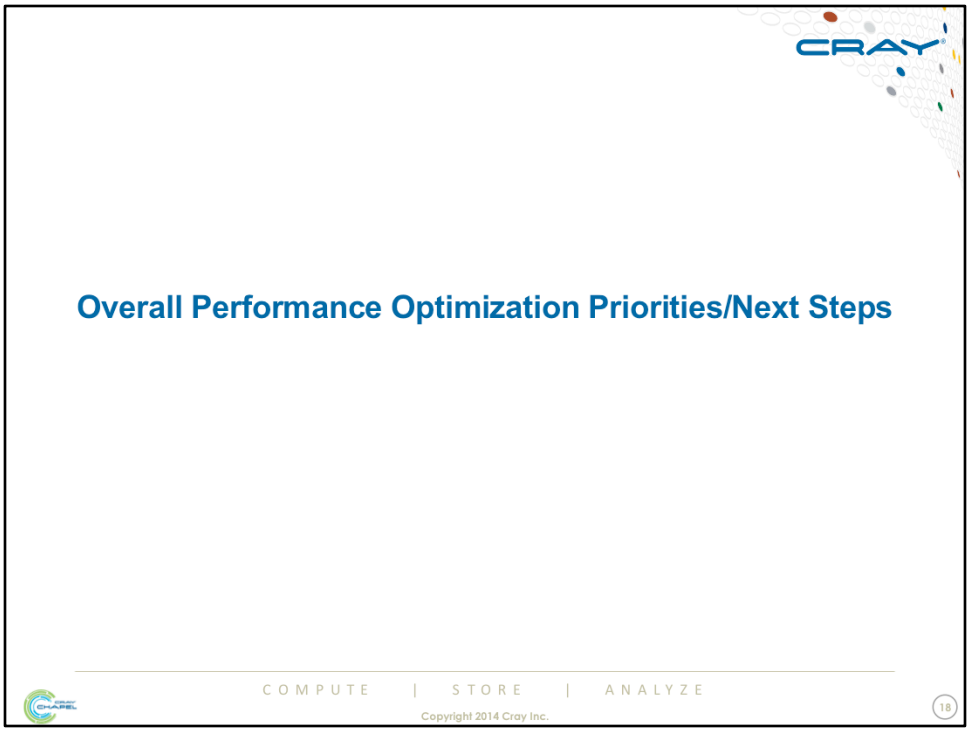
CHPL_TARGET_ARCH: Next Steps

- Investigate why the timings went up for ddata


Other Performance Optimizations
COMPUTE | STORE | ANALYZE
Copyright 2014 Cray Inc.
16

Other Performance Optimizations

- **Switch to Qthreads tasking layer (see 'Runtime' slides)**
  - and switch in thread counts from logical to physical cores by default
- **Improved the performance of the 1D array serial iterator**
  - previously, we were using a more expensive (strided) idiom
- **Made task counters use network atomics when available**
  - previously, they used processor atomics and active messages
- **Improved the performance of readstring()**
- **Reduced communication counts due to other refactorings**


Overall Performance Optimization Priorities/Next Steps
COMPUTE | STORE | ANALYZE
Copyright 2014 Cray Inc.
18

Overall Perf. Opt. Priorities/Next Steps

- Focus on SIMD-ization of generated code
- Reduce amount of unnecessary communication code
  - Relates to earlier item about --local vs. --no-local
- Add support for standalone parallel iterators
- Optimize reductions
- Improve LICM for cases like Fannkuch that backslid

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/