Chapel: A Productive Parallel Programming Language

Lydia Duncan, Chapel Team, Cray Inc.

Women Techmakers: Community Tech-Talks

January 19th, 2016

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.

What is Chapel?

- **An emerging parallel programming language**
  - Design and development led by Cray Inc.
    - in collaboration with academia, labs, industry; domestically & internationally
- **An open-source (Apache license) project on GitHub**
- **A work-in-progress**
- **Goal: Improve productivity of parallel programming**

Chapel is Portable

- **Chapel's design is intended to be hardware-independent**

- **The current release requires:**
  - a C compiler
  - a *NIX environment
  - POSIX threads
  - (for distributed execution): support for RDMA, MPI, or UDP

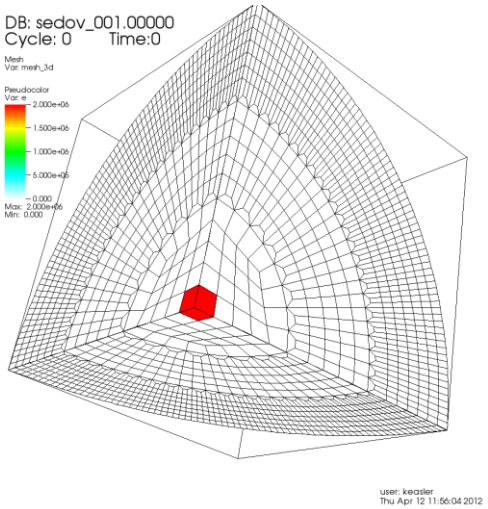
- **Chapel can run on…**

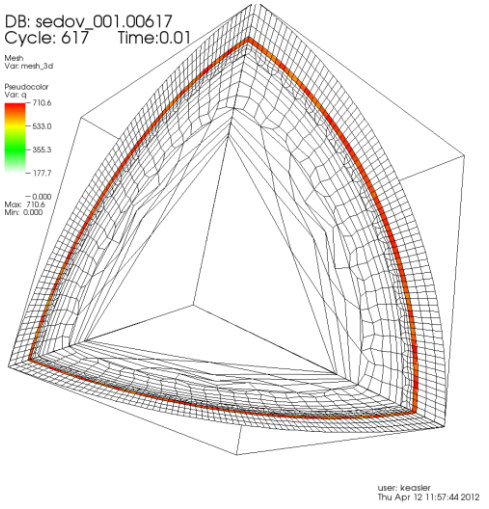
  …laptops and workstations
  …commodity clusters
  …the cloud
  …HPC systems from Cray and other vendors
  …modern processors like Intel Xeon Phi, GPUs*, etc.

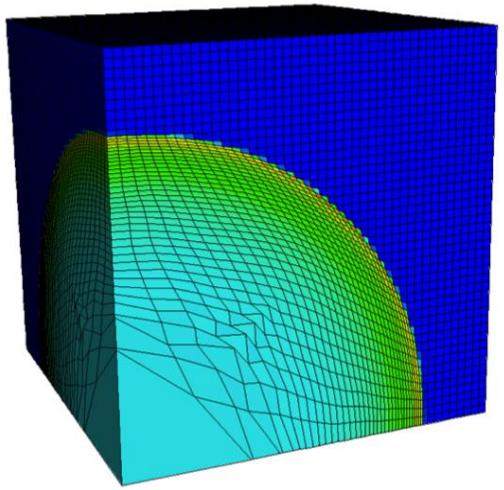
  * = academic work only; not yet supported by the official release

LULESH: a DOE Proxy Application

Goal: Solve one octant of the spherical Sedov problem (blast wave) using Lagrangian hydrodynamics for a single material



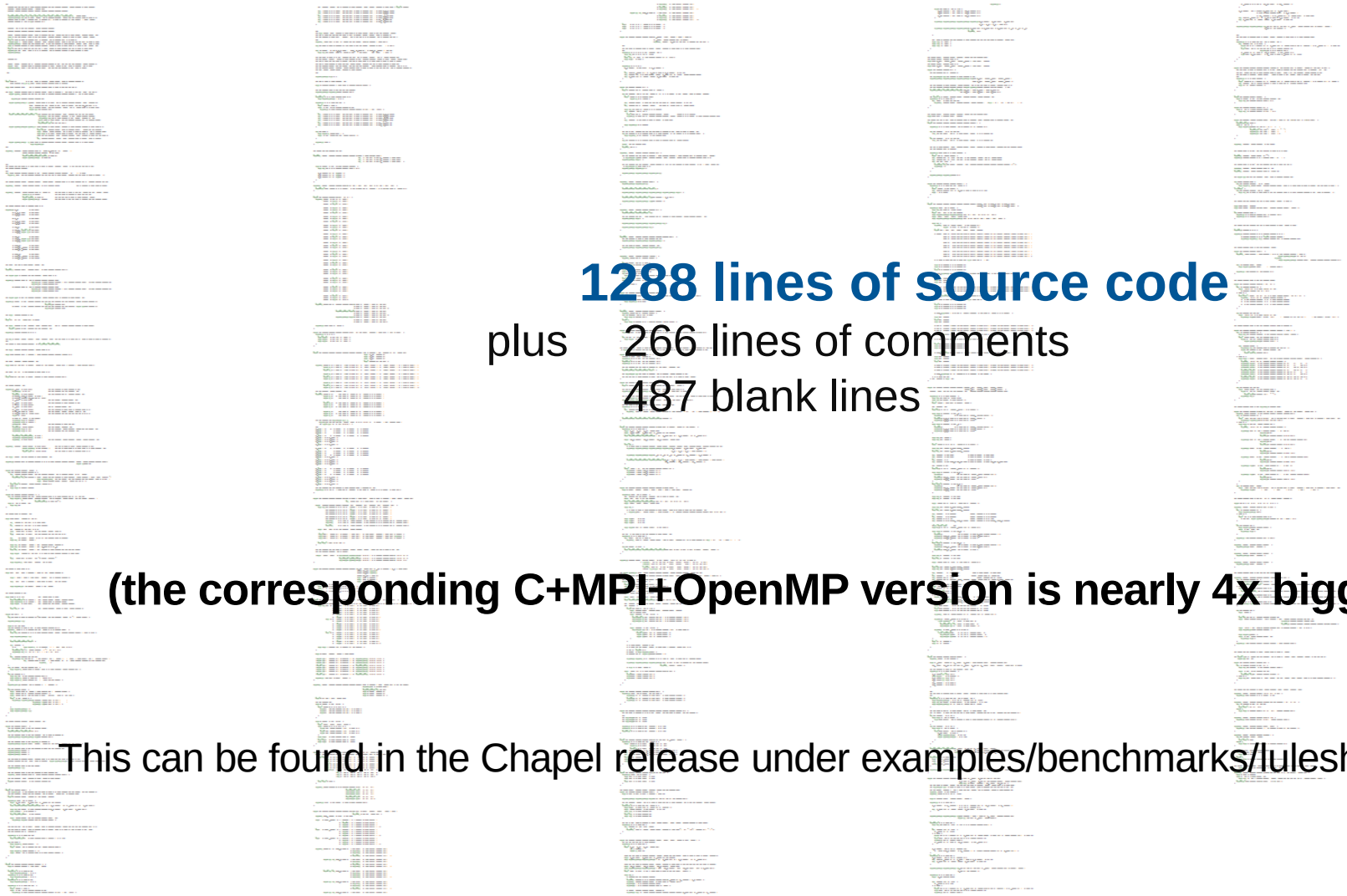




pictures courtesy of Rob Neely, Bert Still, Jeff Keasler, LLNL

LULESH in Chapel

LULESH in Chapel



1288 lines of source code

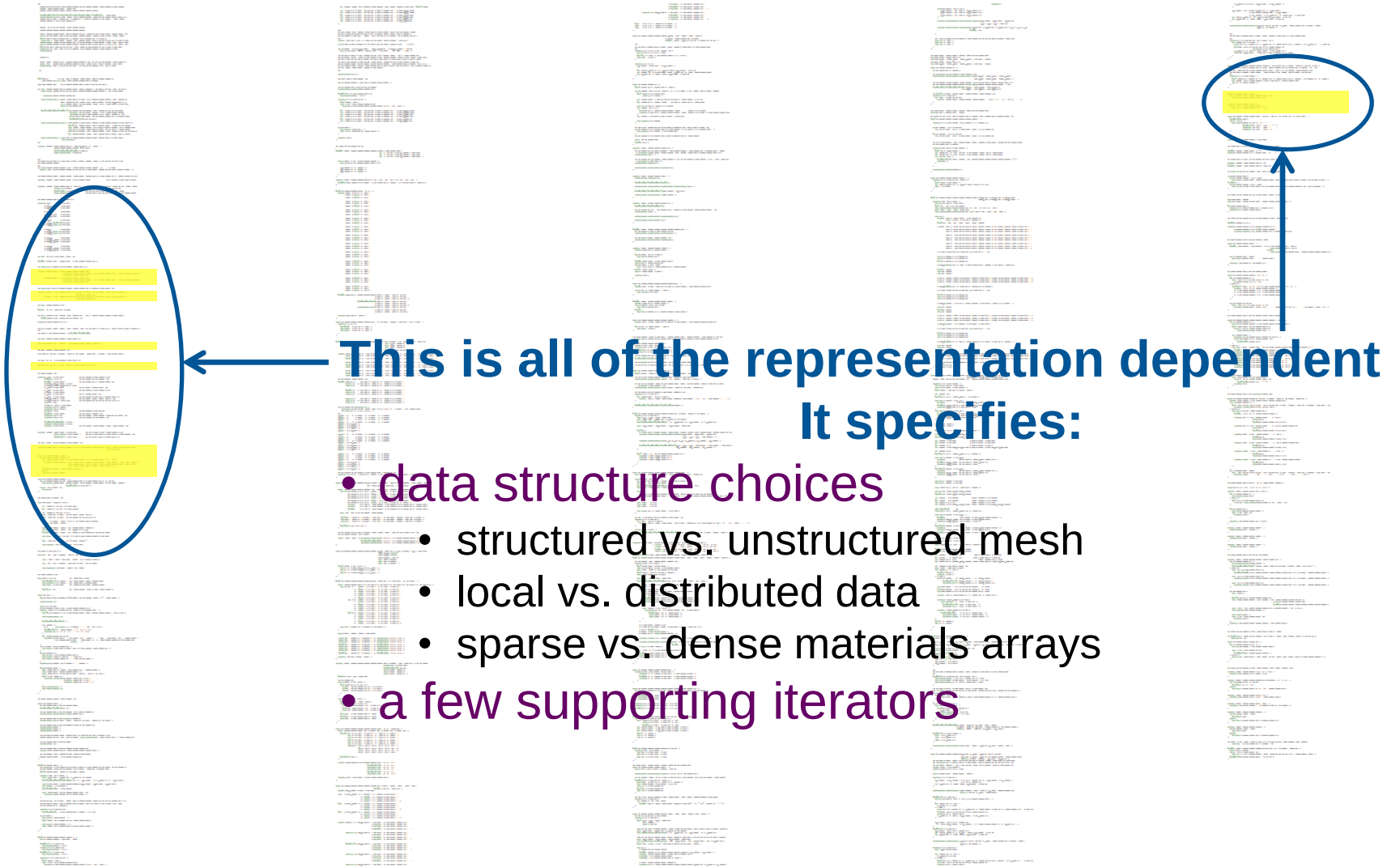
plus 266 lines of comments

487 blank lines

(the corresponding C+MPI+OpenMP version is nearly 4x bigger)

This can be found in the Chapel release under examples/benchmarks/lulesh/*.chpl

LULESH in Chapel

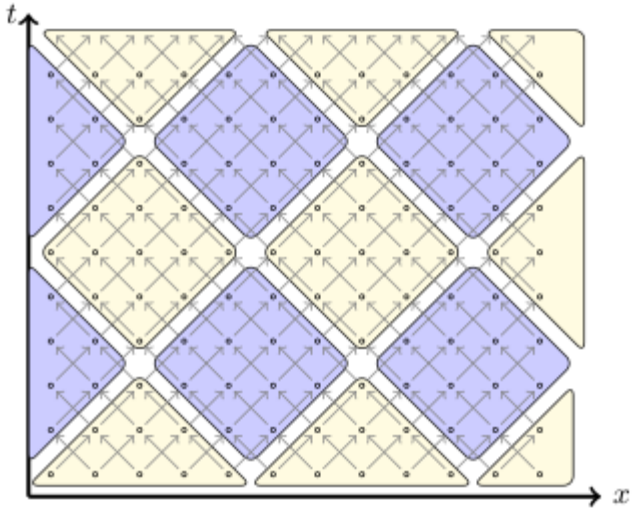


This is all of the representation dependent code. It specifies:

- data structure choices
  - structured vs. unstructured mesh
  - local vs. distributed data
  - sparse vs. dense materials arrays
- a few supporting iterators

Other Uses of Chapel: Diamond Tiling

- **Tiling – strategy to reduce mem. bandwidth pressure**
  - Improves performance and scalability of benchmark
  - Usually must be done by hand
  - Traditionally:
    - Difficult to write
    - Difficult to maintain
    - Not especially portable



Source: Bertolacci et. al. "Parameterized Diamond Tiling for Stencil Computations with Chapel parallel iterators." ACM International Conference on Supercomputing, Newport Beach, CA. June 8-11, 2015. Conference Presentation/Paper.

Other Uses of Chapel: Diamond Tiling

```
// Loop over tile wavefronts.
for (kt=ceild(3,tau)-3; kt<=floord(3*T,tau); kt++) {
  // The next two loops iterate within a tile wavefront.
  int k1_lb = ceild(3*Lj+2+(kt-2)*tau,tau*3);
  int k1_ub = floord(3*Uj+(kt+2)*tau,tau*3);
  int k2_lb = floord((2*kt-2)*tau-3*Ui+2,tau*3);
  int k2_ub = floord((2+2*kt)*tau-3*Li-2,tau*3);
  // Loops over tile coordinates within a parallel wavefront of tiles.
  #pragma omp parallel for ...
  for (k1 = k1_lb; k1 <= k1_ub; k1++) {
    for (x = k2_lb; x <= k2_ub; x++) {
      k2 = x - k1;
      // Removing k1 term from k2 upper and lower bounds enables collapse(2).
      // Loop over time within a tile.
      for (t = max(1, floord(kt*tau-1, 3)); t < min(T+1, tau + floord(kt*tau, 3)); t++) {
        write = t & 1;
        // equivalent to t mod 2
        read = 1 - write;
        // Loops over the spatial dimensions within each tile.
        for (i =  max(Li,max((kt-k1-k2)*tau-t, 2*t-(2+k1+k2)*tau+2)); i <= min(Ui,min((1+kt-k1-k2)*tau-t-1,
        2*t-(k1+k2)*tau)); i++) {
          for (j =  max(Lj,max(tau*k1-t, t-i-(1+k2)*tau+1)); j <= min(Uj,min((1+k1)*tau-t-1, t-i-k2*tau));
          j++) {
            A[write][x][y] = (A[read][x-1][y] + A[read][x][y-1] + ... ; } } } } } }
```

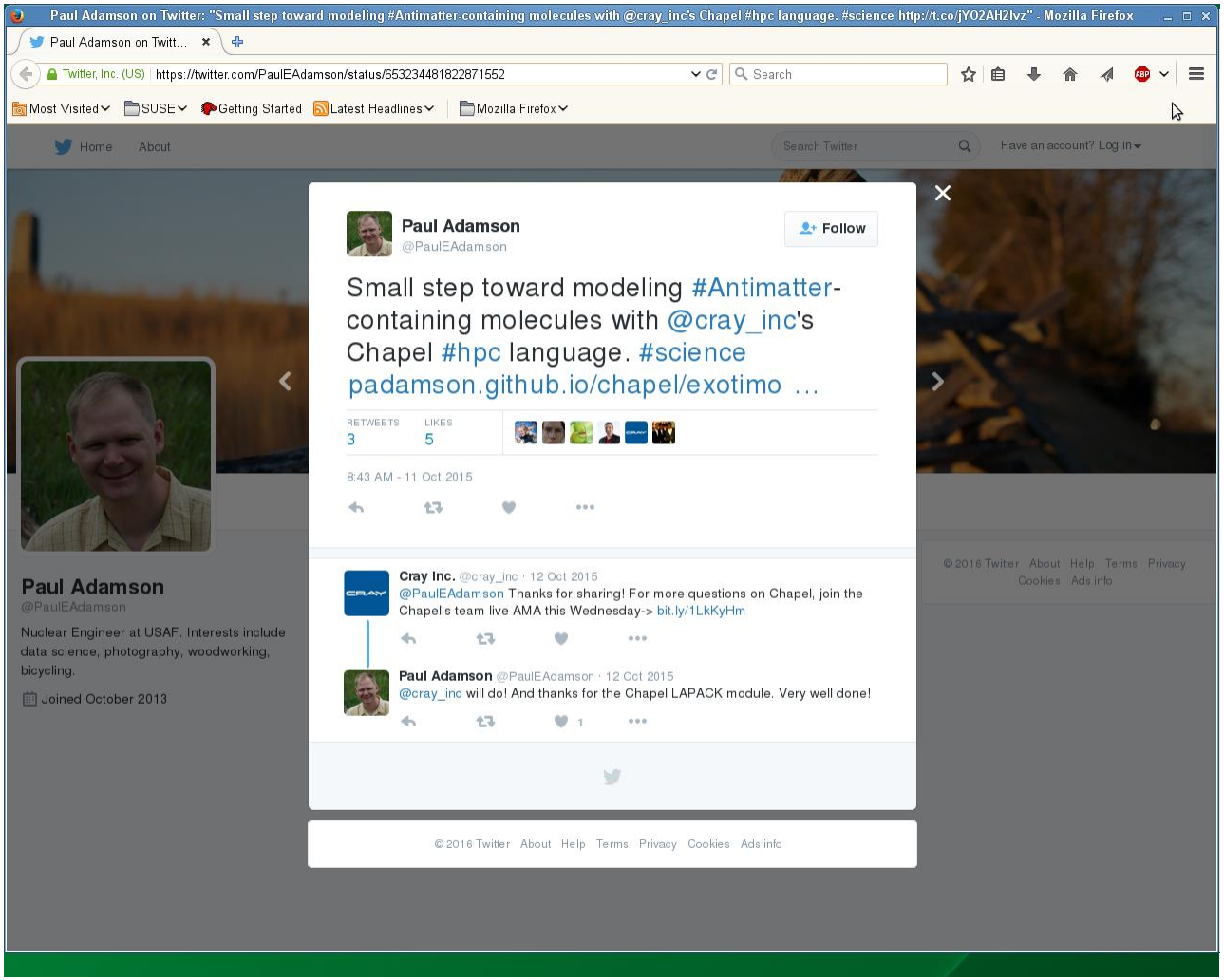
Source: Bertolacci et. al. "Parameterized Diamond Tiling for Stencil Computations with Chapel parallel iterators." ACM International Conference on Supercomputing, Newport Beach, CA. June 8-11, 2015. Conference Presentation/Paper.

Other Uses of Chapel: Diamond Tiling

```
forall (read, write, x ,y) in DiamondTileIterator(L, U, T, tau) {
  A[write, x, y] = (A[read,x-1,y] + A[read,x,y-1] + A[read,x  ,y] +
        A[read,x,y+1] + A[read,x+1,y]) / 5;
}
```

Source: Bertolacci et. al. “Parameterized Diamond Tiling for Stencil Computations with Chapel parallel iterators.” ACM International Conference on Supercomputing, Newport Beach, CA. June 8-11, 2015. Conference Presentation/Paper.

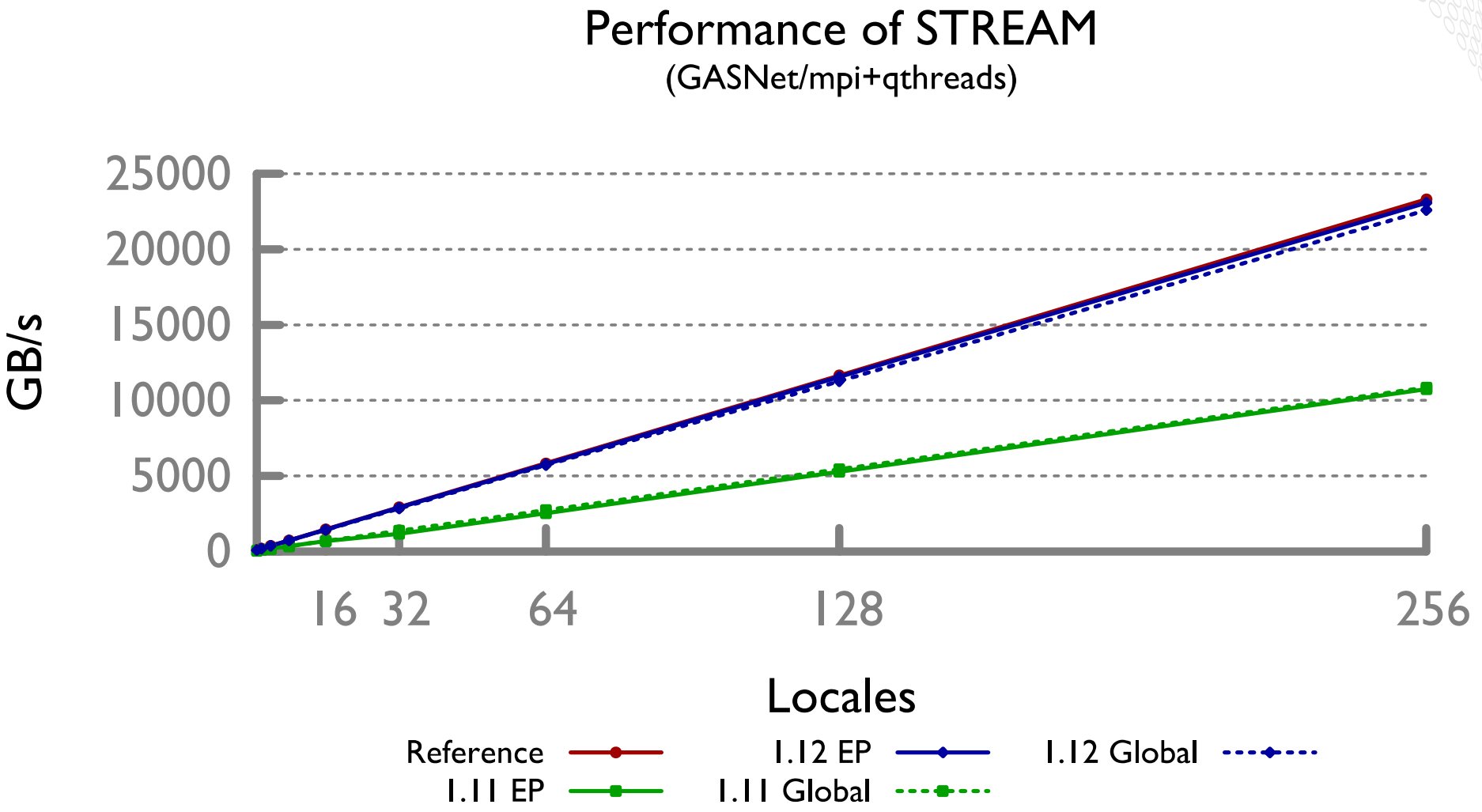
Other Uses of Chapel



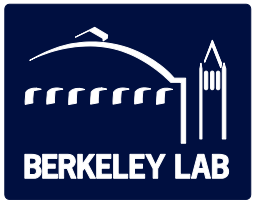
Chapel is a Work-in-Progress

- **Currently being picked up by early adopters**
  - Users who try it generally like what they see
  - Last release got 1400+ downloads over six months
- **Most features are functional and working well**
  - some areas need improvements: strings, object-oriented features
- **Performance is hit-or-miss**
  - shared memory performance is often competitive with C+OpenMP
  - distributed memory performance needs more work
- **We are actively working to address these lacks**

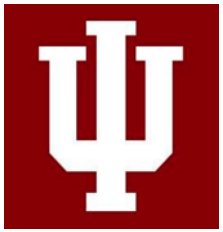
STREAM Scalability



Chapel is a Collaborative Effort



Lawrence Berkeley
National Laboratory



(and many others as well…)

http://chapel.cray.com/collaborations.html

Chapel is Open-Source

- **Chapel's development is hosted at GitHub**
  - https://github.com/chapel-lang
- **Chapel is licensed as Apache v2.0 software**
- **Download/install online**
  - see http://chapel.cray.com/download.html for instructions

Online Resources

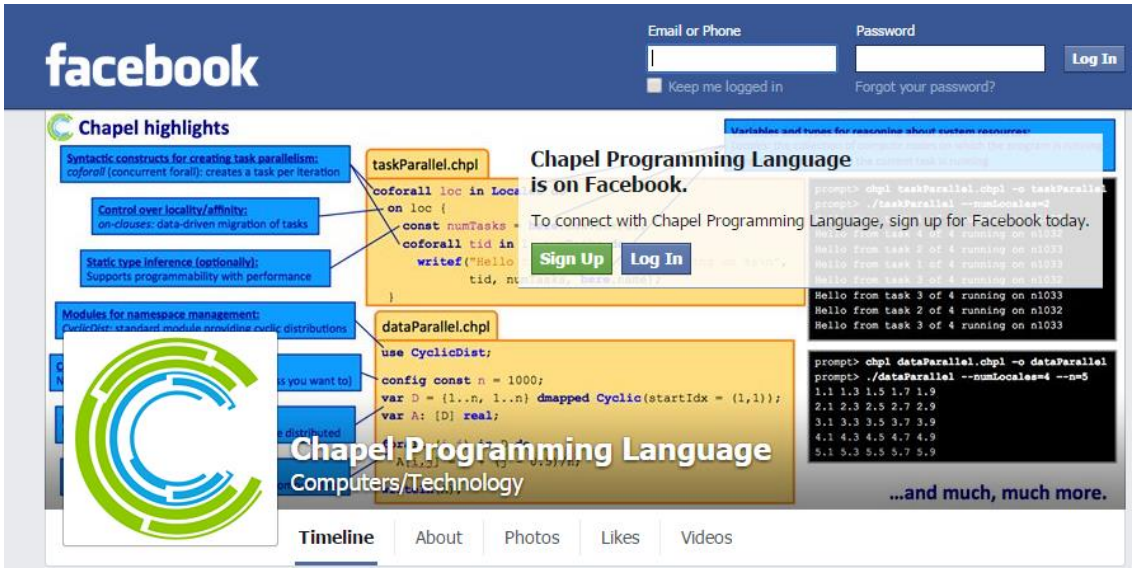
Project page: http://chapel.cray.com

- overview, papers, presentations, language spec, …

GitHub page: https://github.com/chapel-lang

- download Chapel; browse source repository; contribute code

Facebook page: https://www.facebook.com/ChapelLanguage



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, and URIKA. 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.

http://chapel.cray.com chapel_info@cray.com http://github.com/chapel-lang/chapel/