One Parallel Language to Rule them All?

Chapel for HPC, Data Analytics, Machine Learning, …

Brad Chamberlain, Chapel Team, Cray Inc.

UW PLSE Research Retreat

September 12th, 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?

Chapel: A productive parallel programming language

- portable
- open-source (GitHub, Apache 2.0)
- extensible
- a collaborative effort
- a work-in-progress
- designed primarily for High Performance Computing (HPC)

Goals:

- Support general parallel programming
  - any parallel algorithm on any parallel hardware
- Make parallel programming far more productive
  - as programmable as Python
  - as fast as Fortran
  - as portable as C
  - as scalable as MPI
  - as fun as your favorite language

Sample Chapel Programs

Explicit parallelism and locality

```
coforall loc in Locales do
  on loc {
    const locTasks = here.maxTaskPar;
    coforall tid in 1..locTasks do
      writef("Hello from task %n of %n "+
             "running on %s\n",
             tid, locTasks, here.name);
  }
```

Abstract parallelism and locality

```
use CyclicDist;
config const n = 1000;
var D = {1..n, 1..n}
        dmapped Cyclic(startIdx = (1,1));
var A: [D] real;
forall (i,j) in D do
  A[i,j] = i + (j - 0.5)/n;
writeln(A);
```

Chapel for Data Analytics?

~4 years ago: Nah, seems like Hadoop is serving users well

Then, spoke to Hadoop programmers:

- Not as general, programmable, flexible as desired
- Wishlist matched Chapel well:
  - parallelism, scalability
  - large, distributed data structures
  - productivity-oriented features
- **Since then:** Spark also arrived on the scene

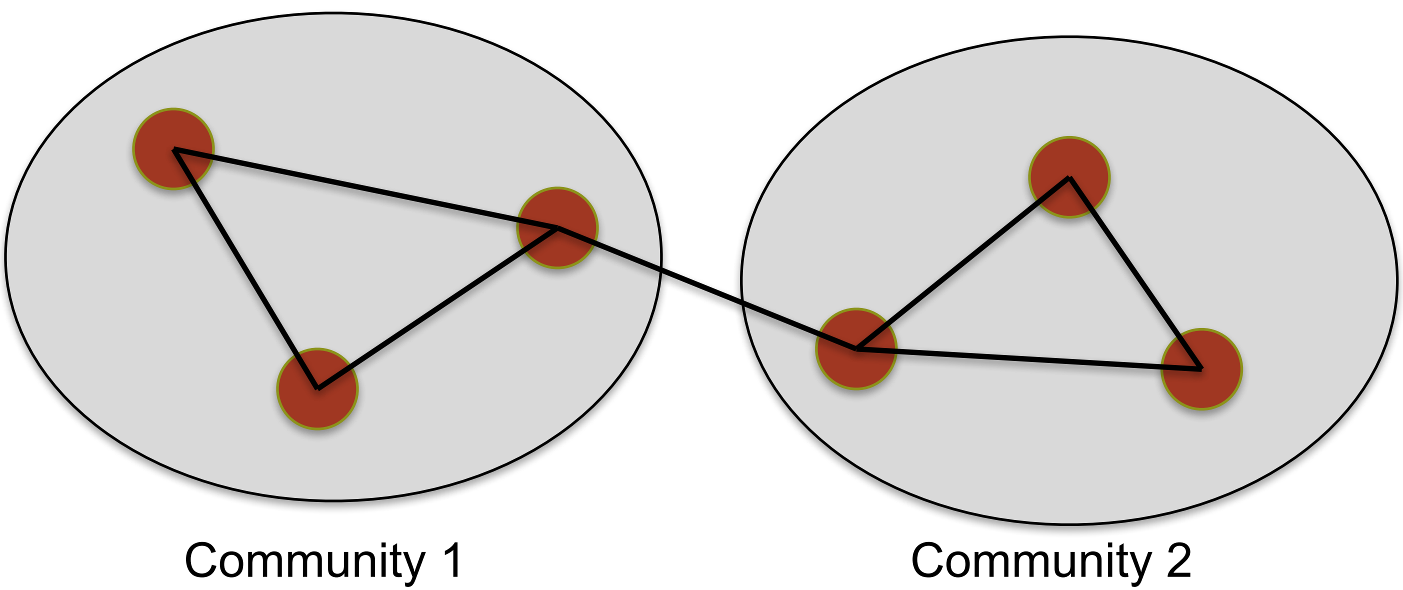
So:

- Began looking into data analytics within Chapel
- But, what to study…?

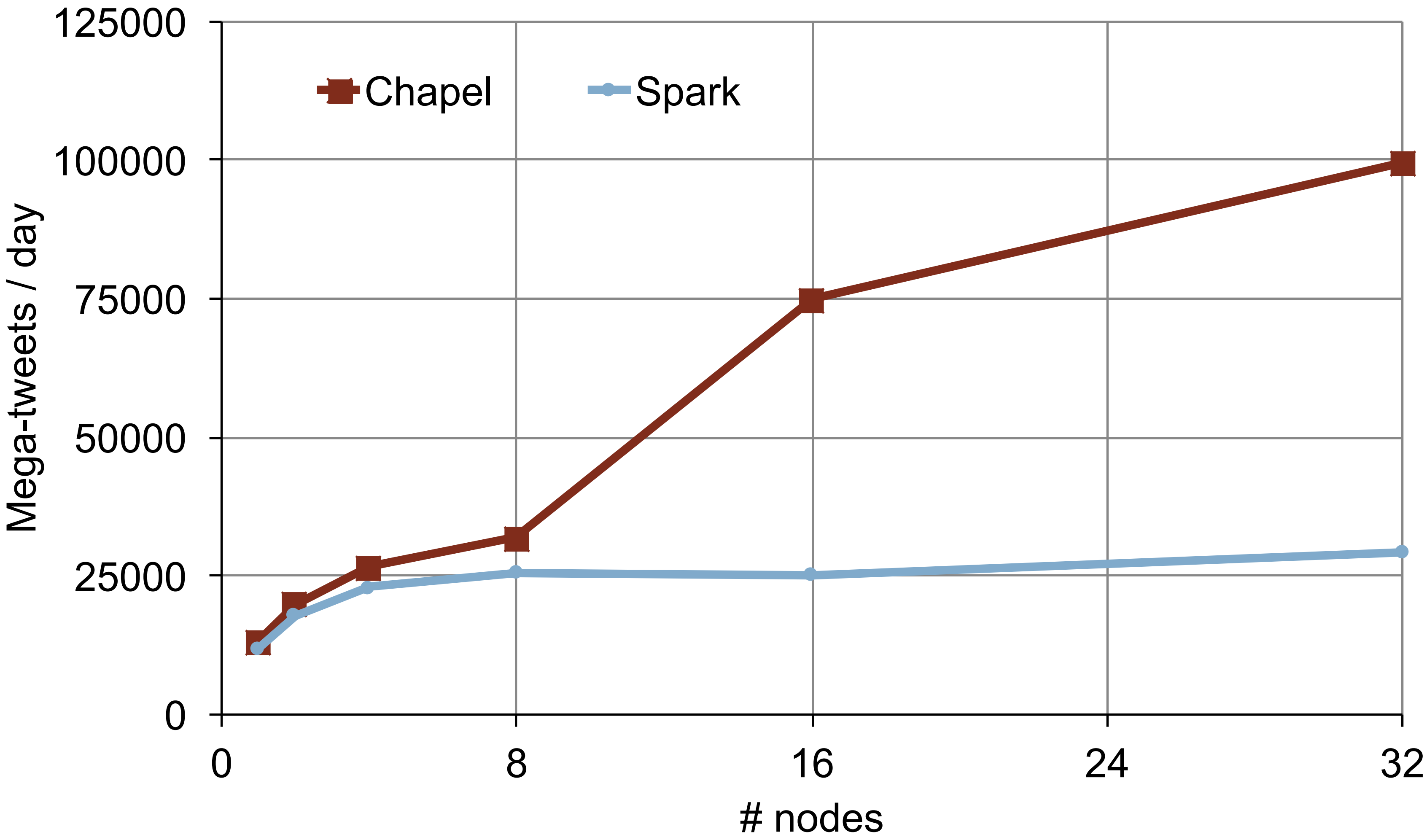
Twitter Community Detection Benchmark

Computation steps:

- Read in gzip files storing JSON-encoded tweets
- Find pairs of Twitter users that @mention each other
- Construct a graph from such users
- Run a label propagation algorithm on that graph
- Output the community structure resulting from label propagation



Twitter Graph Creation: Chapel vs Spark*



* Lots of caveats. Chapel and Spark implementations are not necessarily optimal. Computing mutual mentions only. 420 files, XC30 36-cores/locale, Chapel version used gasnet, fifo, gnu, fe29555c. Spark 1.5.2

Twitter study running out of steam… What's Next?

To make a splash in…

…data analytics
…machine learning
…your favorite parallel, scalable application area

…what features would a parallel language need?

…what killer apps / demonstrations should it pursue?

…what should we do with Chapel?

We're interested in collaborating with experts in such areas

One Parallel Language to Rule them All?

Chapel for HPC, Data Analytics, Machine Learning, …

Brad Chamberlain, Chapel Team, Cray Inc.

UW PLSE Research Retreat

September 12th, 2016

The Chapel Team at Cray (Summer 2016)

14 full-time employees + 2 summer interns + 1 visiting academic
(one of each started after this photo was taken)

Chapel Collaborations

HERIOT WATT UNIVERSITY

AMD

THE GEORGE WASHINGTON UNIVERSITY WASHINGTON, DC

WESTERN WASHINGTON UNIVERSITY

東京大学 THE UNIVERSITY OF TOKYO

Colorado State University

RICE

UNIVERSITY OF MARYLAND

BERKELEY LAB Lawrence Berkeley National Laboratory

Lawrence Livermore National Laboratory

Sandia National Laboratories

HAVERFORD COLLEGE

(your institution here?)

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

Questions?

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.

CRAY CHAPEL

CRAY

THE SUPERCOMPUTER COMPANY

Example Tweet in JSON format

- Tweets have ~63 fields stored in nested structures

{ "coordinates": null, "created_at": "Fri Oct 16 16:00:00 +0000 2015", "favorited": false, "truncated": false, "id_str": "28031452151", "entities": { "urls": [{ "expanded_url": null, "url": "http://chapel.cray.com", "indices": [69, 100] }], "hashtags": [ ], "user_mentions": [{ "name": "Cray Inc.", "id_str": "23424245", "id": 23424245, "indices": [25, 30], "screen_name": "cray" }] }, "in_reply_to_user_id_str": null, "text": "Let's mention the user @cray -- here is an embedded url .... http://chapel.cray.com", "contributors": null, "id": 28039652140, "retweet_count": null, "in_reply_to_status_id_str": null, "geo": null, "retweeted": false, "in_reply_to_user_id": null, "user": { "profile_sidebar_border_color": "C0DEED", "name": "Cray Inc.", "profile_sidebar_fill_color": "DDEEF6", "profile_background_tile": false, "profile_image_url": "http://a3.twimg.com/profile_images/2342452/icon_normal.png", "location": "Seattle, WA", "created_at": "Fri Oct 10 23:10:00 +0000 2008", "id_str": "23502385", "follow_request_sent": false, "profile_link_color": "0084B4", "favourites_count": 1, "url": "http://cray.com", "contributors_enabled": false, "utc_offset": -25200, "id": 23548250, "profile_use_background_image": true, "listed_count": 23, "protected": false, "lang": "en", "profile_text_color": "333333", "followers_count": 1000, "time_zone": "Mountain Time (US & Canada)", "verified": false, "geo_enabled": true, "profile_background_color": "C0DEED", "notifications": false, "description": "Cray Inc", "friends_count": 71, "profile_background_image_url": "http://s.twimg.com/a/2349257201/images/themes/theme1/bg.png", "statuses_count": 302, "screen_name": "gnip", "following": false, "show_all_inline_media": false }, "in_reply_to_screen_name": null, "source": "web", "place": null, "in_reply_to_status_id": null }

Reading JSON Tweets

// define Chapel records whose fields reflect only
// the portions of the JSON data we care about

```
record TweetUser {
  var id: int;
}
record TweetEntities {
  var user_mentions: list(TweetUser);
}
record User {
  var id: int;
}
record Tweet {
  var id: int,
      user: User,
      entities: TweetEntities;
}
```

```
proc process_json(…) {
  var tweet: Tweet;

  while true {
    // "%~jt" format string:
    //     j: JSON format
    //     t: any record
    //     ~: skip other fields
    got = logfile.readf("%~jt",
                        tweet,
                        error=err);
    if got && !err then
      handle_tweet(tweet);
    if err == EFORMAT then ...;
    if err == EEOF then break;
}
```

Processing Tweets: Productivity Comparison

Spark

- RDDs are immutable
- Algorithm written in terms of mapping a fn on data

Chapel

- Chapel arrays are mutable
- Algorithm written in terms of parallel loops