Tools Updates

Chapel Team, Cray Inc.

Chapel version 1.12

October 1st, 2015

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.

Outline

- chplvis: a Tool for Performance Visualization
- 'chpldoc' Improvements

chplvis: a Tool for Performance Visualization

chplvis: Background

- **Performance debugging tool support for Chapel is limited**
  - No support for tools like Tau, ParaView, and VisIt
  - Basic support for tools like valgrind and gdb
  - Parallel performance debugging is not easy

- **Chapel runtime uses tasks and locales**
  - Direct support for these features is important
  - Other tools do not support this exact paradigm
    - e.g., provide thread-based information rather than task-based

    ⇒ Challenging to integrate support for Chapel in other tools

chplvis: A Communication and Task Viewer

- **A new visualization tool for…**
  …Inter-locale communication
  …Task concurrency within a locale
  …CPU and clock time

- **Support provided via a standard module: `VisualDebug'**
  - User calls module functions to indicate program segments to visualize
    - http://chapel.cray.com/docs/latest/modules/standard/VisualDebug.html
  - Runtime collects data about runtime events, creates data files
  - Chplvis program displays data from files visually

chplvis: VisualDebug.chpl module

startVdebug("dataDirName")

- Starts collecting runtime events
- Creates named directory for data collection
- Each locale writes a separate data file

stopVdebug()

- Stops all data collection, closes all data files

tagVdebug("tagName")

- Inserts tag in the data, allows partial run visualizations
- Chplvis tool uses tag name to control displayed data

pauseVdebug()

- Temporarily suspend collection of runtime events

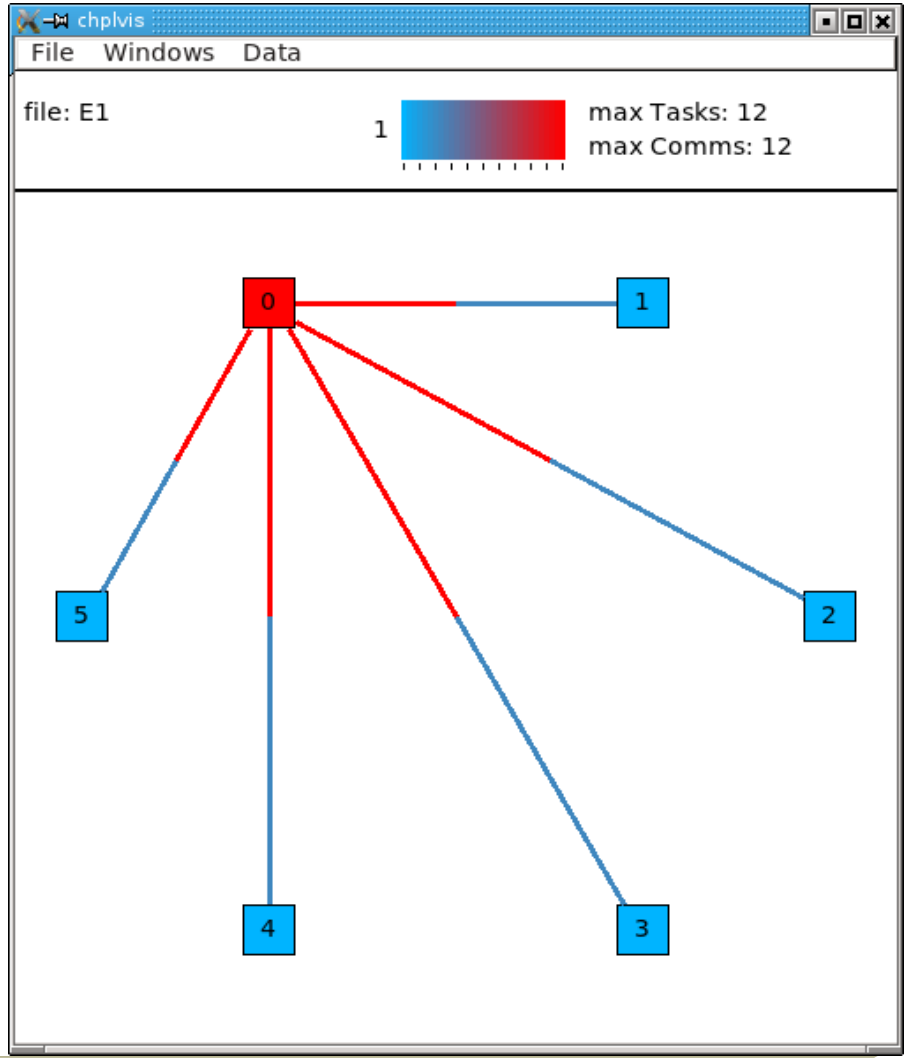
chplvis: Simple Example

Simple coforall and on clause:

```
use VisualDebug;

startVdebug("E1");

coforall loc in Locales do
   on loc do
     writeln("Hello from locale " + here.id + ".");

stopVdebug();
```

chplvis: Simple Example display

- **Simple coforall and on-clause display, 6 locales**
  - Boxes are locales
  - Lines are communication
  - Color shows data value
    - Segment adjacent to box indicates number of communications *into* locale
- **Interactive display**
  - Click on boxes
  - Click on lines
  - Data menu changes display



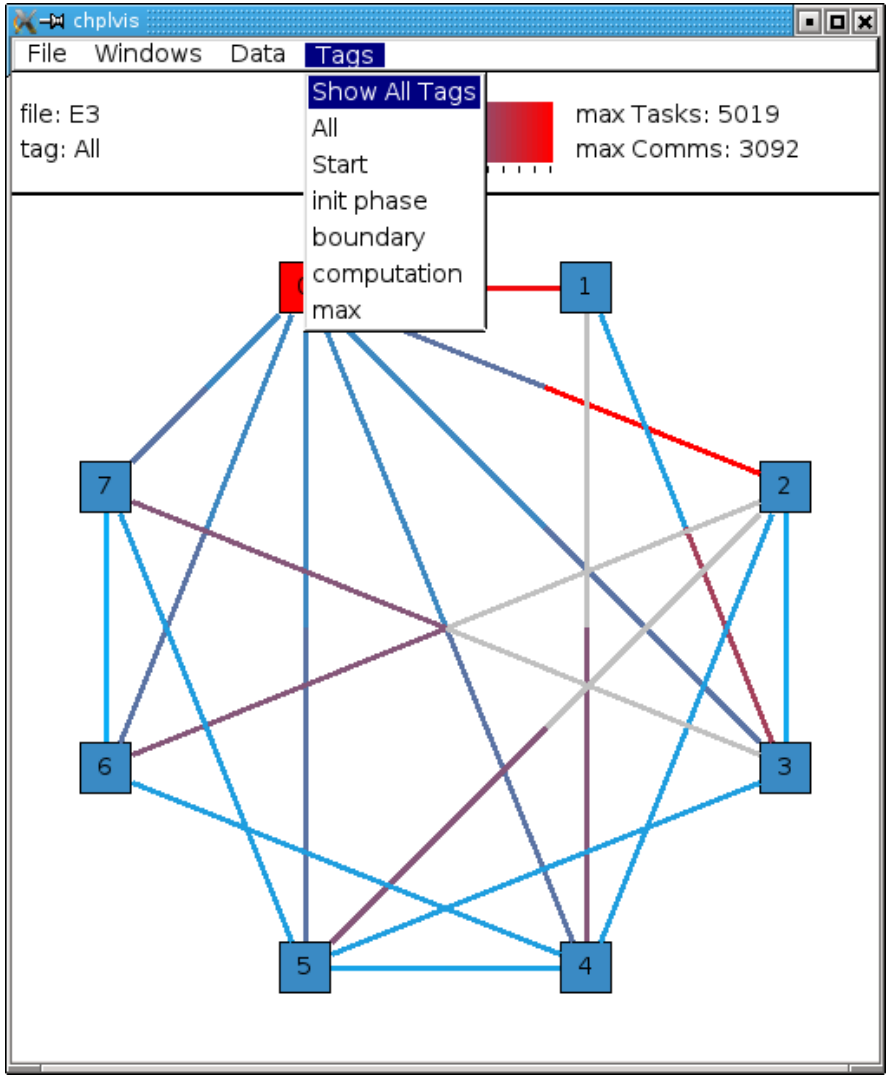
chplvis: Jacobi Example

Jacobi computation loop:

```
while (delta > epsilon) {
  tagVdebug("computation");  // Tag the computation part of this loop
   for t in 1..compLoop {
     forall (i,j) in R do A[i,j] = Temp[i,j];
     forall (i,j) in R do
       Temp[i,j] = (A[i-1,j]+A[i+1,j]+A[i,j-1]+A[i,j+1])/4;
   }

   tagVdebug("max");  // tag the reduction part of this loop.
   forall (i,j) in R do
     Diff[i,j] = abs(Temp[i,j] - A[i,j]);
   delta = max reduce Diff;

   pauseVdebug();
   iteration += compLoop;
 }
```

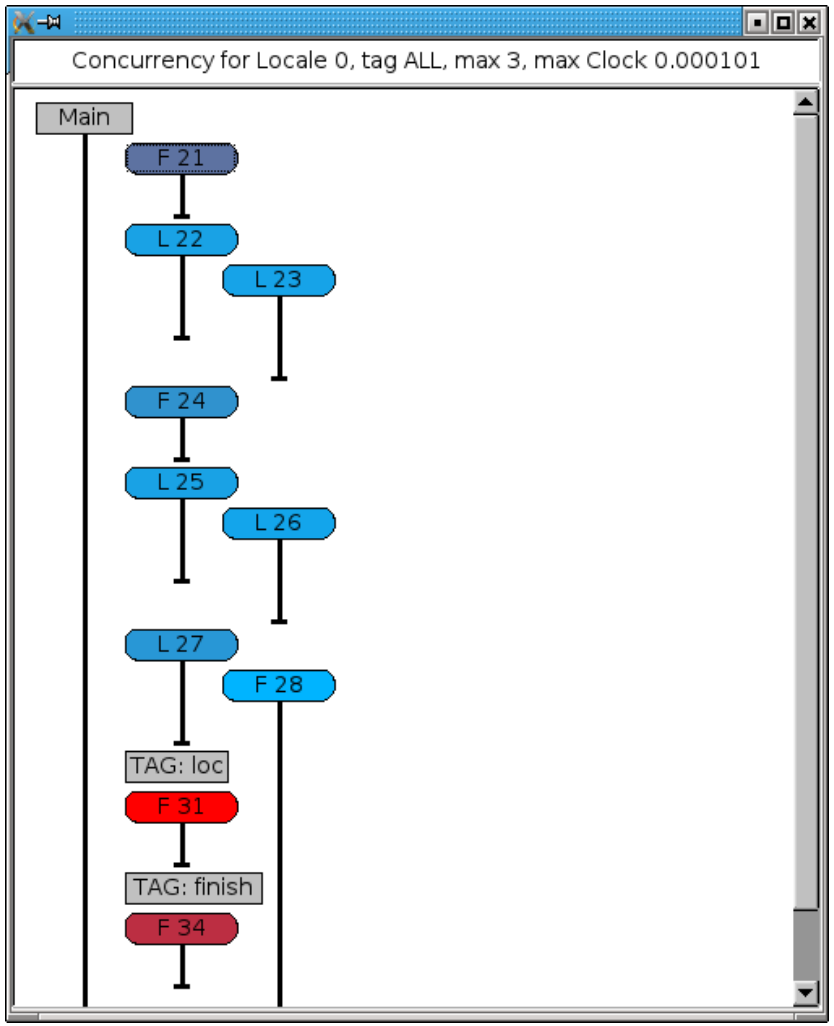
chplvis: Jacobi Example Display

- **8 locale run with tags**
  - 'All' ⇒ program start to first tag
  - Tag names taken from calls
  - Gray segments indicate no communication into that locale



chplvis: Intra-locale task example

- **Task concurrency display**
  - Ovals are tasks
  - Colors show task clock time
  - Lines show (logical) lifetime
  - Tags may be shown in task timeline
- **Display interaction**
  - Click task for list of the task's communications
  - Hover over task for summary display of times and communication counts



chplvis: Status and Next Steps

Status:

- No known bugs
- Works well up to 32 locales
- All planned features working

Next Steps:

- Get user feedback
- Improve visualization for more than 32 locales
- Longer-term: introduce more data-centric views of computation

‘chpldoc’ Improvements

Chpldoc Improvements: Background

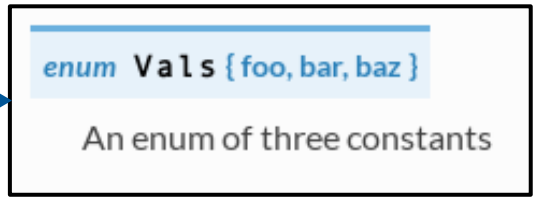
- **chpldoc was much improved in 1.11**
  - Including distinct flags from chpl binary

Chapel 1.11:

```
chpldoc -h
Usage: chpldoc [flags] [source files]

Documentation Options:
  -o, --output-dir <dirname>

      --save-sphinx <directory>

      --comment-style <indicator>

      --text-only
  -h, --help
```

- **Still some bugs to iron out**
  - Initialization expressions of certain types were handled poorly
  - Numeric enum values were dropped on the floor

```
/* An enum of three constants */
enum Vals {foo = 1, bar, baz};
```



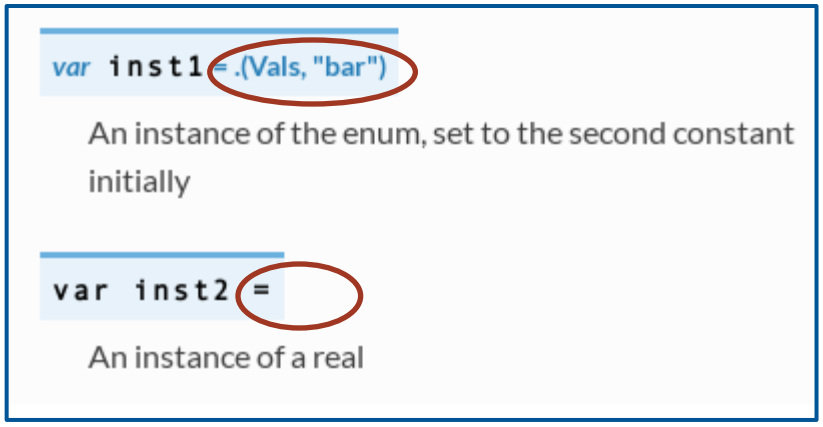
  - Needed pragma to hide symbols from output

Chpldoc Improvements: This Effort

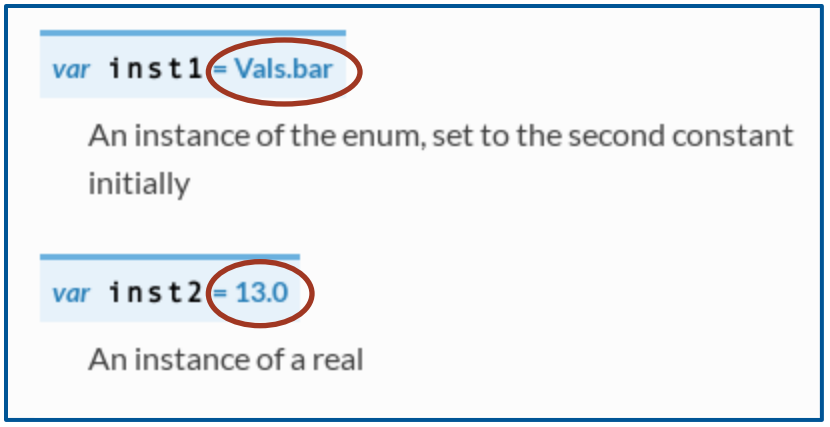
- chpldoc now shows an enum's numeric values
- fixed enum, real, imag, complex initialization expressions

```
/* An instance of the enum, set to the
second constant initially */
var inst1 = Vals.bar;
/* An instance of a real */
var inst2 = 13.0;
```

Chapel 1.11:



Chapel 1.12:



Chpldoc Improvements: This Effort

- **Previously: only ignored symbols with pragma "no doc"**

```
pragma "no doc"
var foo: int;
```

  - Pragmas aren't currently intended for end-users
  - Already wanted privacy control in namespaces
    - This was a quick fix, not a long-term solution

- **Now: private symbols are also ignored by chpldoc**
  - Still supporting "no doc" pragma
    - Not all symbols support 'private' yet
    - Even then, may not want everything 'public' to be documented

Chpldoc Improvements: This Effort

- **More flags**
  - New: --author for setting author output
  - Other flags shared with chpl binary
    - License, copyright, etc.
    - Better developer debugging

Chapel 1.11:

```
chpldoc -h
Usage: chpldoc [flags] [source files]

Documentation Options:
  -o, --output-dir <dirname>

      --save-sphinx <directory>

      --comment-style <indicator>

      --text-only
  -h, --help
```

Chapel 1.12:

```
chpldoc -h
Usage: chpldoc [flags] [source files]

Documentation Options:
  -o, --output-dir <dirname>

      --save-sphinx <directory>

      --author <author>
      --comment-style <indicator>

      --text-only
      --[no-]print-commands

Information Options:
  -h, --help
      --help-env
      --help-settings
      --version
      --copyright
      --license

Developer Flags:
      --[no-]devel
      --gdb
      --lldb
      --print-chpl-home
```

Chpldoc Improvements: Next Steps

- **More bug fixes**
- **Support testing example codes within chpldoc comments**
  - Similar to python doctests
- **Compatibility with newly converted .rst files**
- **Support class/record view, including inheritance info**
  - And add class and record index
- **Link to source code from docs**

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
THE SUPERCOMPUTER COMPANY

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