Priorities for version 1.14

Chapel Team, Cray Inc.

Chapel version 1.13

April 7, 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.

Proposed High-Effort Priorities for 1.14

1. **Close all remaining memory leaks**
   - arrays/domains/distributions, syncs/singles, … ; lock into testing
2. **Finalize and implement initializers and copy semantics**
   - improve existing types based on this work
3. **Multi-locale performance improvements**
   - application-driven scalability and performance improvements
4. **NUMA/KNL locale models and performance**
   - support for NUMA-aware memory allocation, HBM; Qthreads improvements
5. **Irregular array improvements**
   - distributed sparse / associative; performance vs. safety concerns
6. **Data analytics case studies**
7. **Single-locale performance improvements**
   - LCALS-driven improvements, eliminate inner multiplies, vectorization
8. **Implement error-handling**
9. **Expose comm/comp overlap for 'qthreads' (and retire 'muxed'?)**
10. **Improved support for first-class functions (+ closures?)**

Proposed Design Priorities for 1.14

1. **Continue fleshing out package manager design**
2. **Strategy for IPE / Jupyter / Chapel v2**
3. **Incremental recompilation / Separate compilation**
4. **Additional task-oriented hooks for forall-loops**
   - e.g., per-task declarations (?), setup/teardown code (?), task IDs (?)
5. **Expression and implementation of partial reductions**
6. **Non-transitive module uses**
7. **Task teams**
8. **Access to routines for explicit communication in Chapel**
   - pure SPMD; or treat locales as PEs
9. **Strategy for targeting GPUs**
10. **Clarify tuple semantics**

Proposed Modest-Effort Priorities for 1.14

1. Update string interfaces to be UTF-8-ready
2. Support for passing function pointers to external C routines
3. Reduce intents: support any reduction; support cobegins
4. Fix bug in which array "setter" is incorrectly invoked
5. Enable strided bulk communication optimization
6. Add support for .chplrc file
7. Expose library of utility routines for blocking ranges
8. Support for 'void' types
9. Develop good 'Sort' routine(s)
10. Support for 'use' requiring fully qualified names
    - e.g., '`use M except *;`' '`use M only ;`'
11. Make 'chpldoc' testing test .rst (or .rst → .txt) output
12. Fix performance issue in shrinking array-as-vector

Proposed BTR + Docs Priorities for 1.14

1. Complete Chapel Debian package
2. Slurm management of internal test machines
3. Multi-locale cluster-based performance testing
4. User-based issue tracking
5. Expand Users Guide
6. Have nightly testing results report as diff against “parent” run
7. Testing of chpldoc-based examples
8. Make test system Python 2 / 3 agnostic
9. Jenkins interface for testing specific git branch / SHA

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

THE SUPERCOMPUTER COMPANY