Runtime Library Improvements

Chapel Team, Cray Inc.

Chapel version 1.11

April 2, 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

- Qthreads Improvements
- Cray Runtime Changes
- Launcher Improvements
- Runtime Priorities and Next Steps

Qthreads Improvements

Qthreads Tasking Changes

- **As default tasking layer, Qthreads got more exposure**
  - More testing, both over time and across configurations
- **Mixing public+Cray runtime layers added configurations**
- **Result: needed to fix several small problems**

QT Tasking: Tasks Can Monopolize Workers

- **Background:** Bug: tasks could monopolize worker threads

```
while (syncVar.readXX() != N) {
  …
  if … then begin syncVar += 1;
  …
}
```

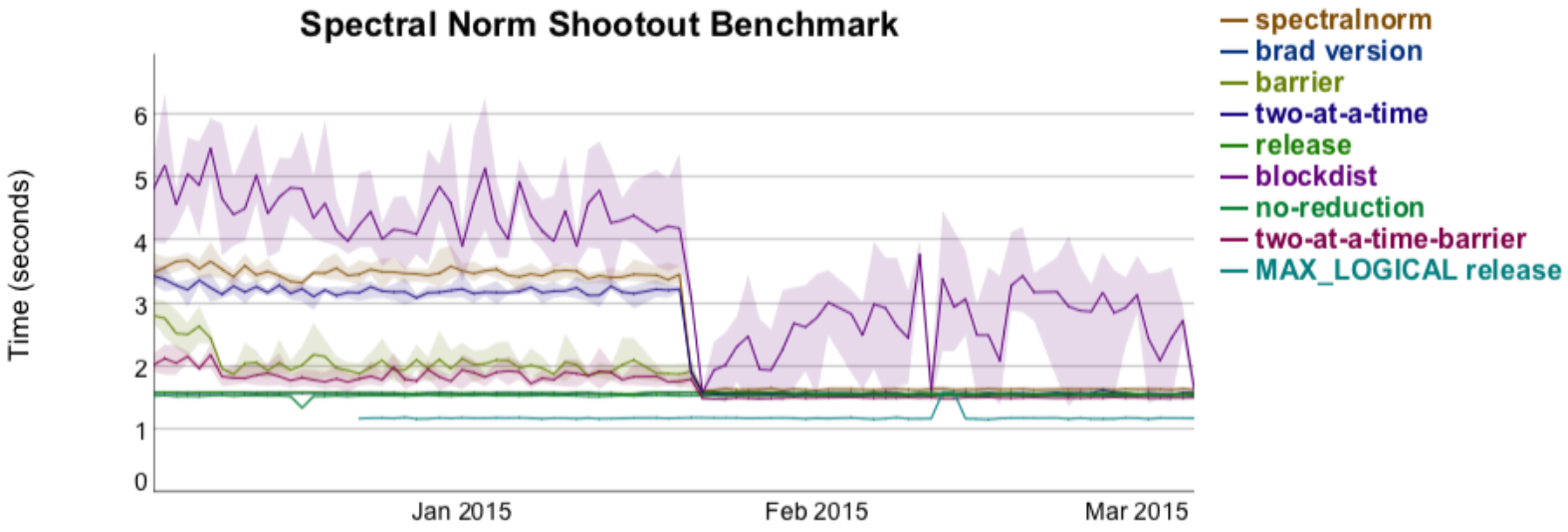
task T_1 is spinning on `syncVar`

task T_2 would release T_1 by writing to `syncVar`

  - If the same thread hosted T_1 and T_2, T_1 could starve T_2
    - *syncVar* F/E always as desired $\Rightarrow$ T_1 never blocked $\Rightarrow$ T_2 never ran
  - "Solved" in 1.10, but could execute slowly and cause testing timeouts
- **This Effort:** Occasionally yield worker even when not blocked
  - About 1% of the accesses for each sync var
    - But only when no other task seems to be trying to access it

QT Tasking: Impact of Worker Yielding

- **Impact:** Qthreads more stable with default scheduler
  - Nightly testing quieter (fewer sporadic timeouts)
  - Performance improvements for Spectral Norm



QT Tasking: Host Tasks in Top-level Processes

Background: Top-level locale processes could not host tasks

- Symptom: intermittent qthreads internal check failure with gasnet
  - 1 test out of ~5000
  - Not a problem in 1.10; appeared following a post-1.10 runtime improvement
- Cause: processes lacked certain capabilities needed to host tasks
  - Example: task-private memory to hold serial state, etc.

✓ Qthreads worker threads: host main program and tasks

✓ Comm polling threads (Active Message handlers): host “fast” on-stmts

✗ Top-level locale processes: host “fast” on-stmts when in comm barriers

This Effort: Added needed capabilities for top-level processes

Impact: Stability

- Qthreads more stable with multiple locales
- Nightly testing quieter

QT Tasking: Execute Serial Subtasks Directly

Background: Qthreads queued subtasks even when serial

- Followed Qthreads design point: many small tasks, well load-balanced
- Encouraged by Qthreads' original default 4 kb stacks
- Costly for Chapel subtasks of serial tasks

This Effort: Execute serial subtasks directly, by calling them

- Matches behavior of other tasking layers
- Enabled by current default 8 mb stacks

Impact: Improved performance

- `quicksort --n=16777216 --thresh=3` 10X faster on 8 CPUs

QT Tasking: Worker Threads Sleep on Darwin

Background: Bug: Chapel sleep() suspends Qthreads worker

- Qthreads intercepts syscalls so workers don't suspend
- So tasking shim sleep(3) should spin, yielding worker, until sleep done
- But intercept mechanism is broken for darwin (OS X)
- Thus tasking shim sleep(3) causes Qthreads worker to suspend

This Effort: work around broken syscall intercept

- Spin-yield until sleep time expires in the tasking layer shim

Impact: Fixes the broken syscall intercept with the most impact

Next Steps: Work around other instances?

- Better if Qthreads syscall intercept could be fixed for darwin, though

Cray Runtime Changes

Cray Runtime Changes: Outline

- Non-blocking comm interface mismatch, public vs. Cray
- Allow mixing public and Cray runtime layers
- Other Cray runtime layer changes
- Testing and performance status

Non-blocking Communication: Background

Two flavors of non-blocking communication support:

- **In Cray ugni comm layer implementation:**
  - Styled after ugni interface
  - Operations:
    Initiation: get
    Completion: try specific (NB), wait specific (B)
  - NB token owned by caller, updated by comm layer
- **In public gasnet comm layer implementation:**
  - Styled after GASNet interface, but incomplete
  - Operations:
    Initiation: get, put
    Completion: check specific (NB without try), wait many (B)
  - NB token owned and updated by comm layer

Non-blocking Comm Unification

This Effort:

- **One flavor of non-blocking communication support:**
  - Styled after GASNet interface, and complete
  - Operations:
    Initiation: get, put
    Completion: check specific (NB without try), try many (NB), **wait many (B)**
  - NB token owned and updated by comm layer
- **Refactored code to adapt to interface changes**
  - ugni comm: alloc/free descriptor, replace operations
  - muxed tasking: rework fine-grained compute/communicate overlap
  - gasnet comm: add try many (NB)
  - all comm layers: changes to comm diags counters

Mixing Public+Cray Layers: Background

- **Public and Cray runtime layers did not interoperate**
  - Non-blocking communication mismatches (just discussed)
  - ugni comm and muxed tasking assumed each other, implicitly
  - Example:

muxed tasking:
- gets all memory from runtime
- comm handler is a task

⇒ ugni comm layer sees only a few (special case) references to memory not registered with the NIC

but:

qthreads tasking:
- gets memory from libc
- comm handler is a pthread

⇒ comm layer sees lots of references to memory not registered with the NIC

Mixing Public+Cray Layers: This Effort

ugni comm

- Handle unregistered memory as source or target, local or remote

muxed tasking

- Make the comm handler a pthread
  - Improves latency, reduces starvation, follows Qthreads tasking lead
- Allow for running Chapel code in main process and comm thread
  - Adapt to comm handler change above, interoperate with gasnet comm
  - Mainly: emulate certain capabilities normally only present in soft-thread threads

Mixing Public+Cray Layers: This Effort (continued)

qthreads tasking

- Disable guard pages when impractical
  - Background: with comm=ugni we use libhugetlbfs, for NIC memory registration
    - But huge guard pages aren't practical
  - This effort: auto-disable guard pages when heap page != system page
  - Impact: no stack overflow detection with ugni+qthreads
- Limit memory pool size
  - Background: specialized memory pools improve allocator performance
    - Chapel limits pool sizes, but a bug resulted in stack pools > 1 GiB with comm=ugni
  - This work: fix the bug

Other Cray Runtime Changes: ugni Comm

ugni comm

- Change default heap size to 2/3 available node memory
  - Done after diagnosing over-allocation with ugni+qthreads
  - Diagnosis false, but kept change because startup faster, esp. on 128g nodes
  - With slurm, default still just 16g due to resource limiting for node sharing (*note*: post-release, may have a solution for this)
- Fix bugs in **`real(*)`** network AMOs that happened to hit the local node

Other Cray Runtime Changes: muxed Tasking

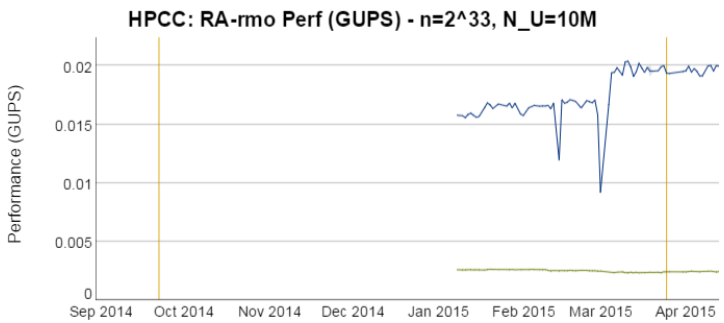
muxed tasking

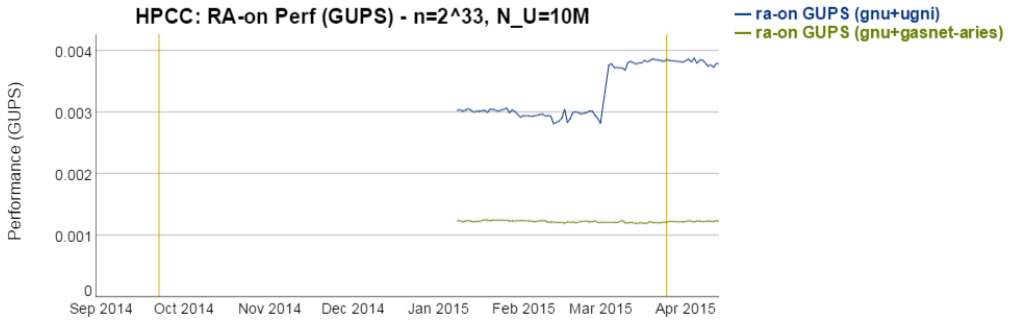
- Change default #threads per locale to #cores (matches other task layers)
  - Improved performance in most cases
  - More threads are still preferable for latency-limited code like SSCA2, RA, etc.
- Add support for oversubscription (more than 1 Chapel locale per node)
  - Supports testing with limited nodes (gasnet+muxed, for example)
- Add task-private data implementation
  - Was in tasking interface, but muxed had no implementation yet
  - Necessary but not sufficient for remote caching to work with muxed

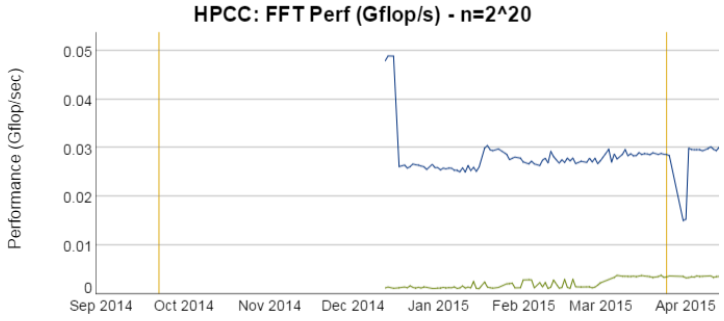
Cray RT Changes Impact #1: ugni Is Default

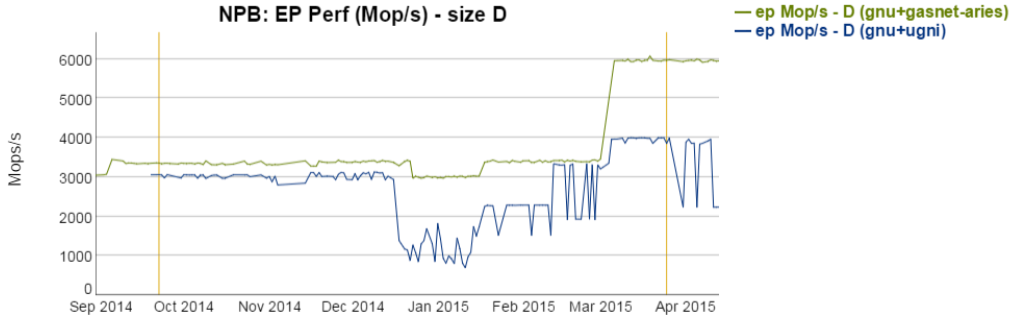
- **ugni+muxed is 1.11 default in pre-built Cray X* module!**
  - Have wanted this for a long time due to better performance
  - Didn't have the necessary testing confidence in previous releases
- **No big correctness testing disparities vs. gasnet+qthreads**
  - Comm diagnostics differences
  - Memory diagnostics differences
  - Message differences due to network atomics vs. processor atomics
  - Inter-locale races when writing output
- **Much improved performance**

Cray RT Changes Impact #1: ugni Is Default



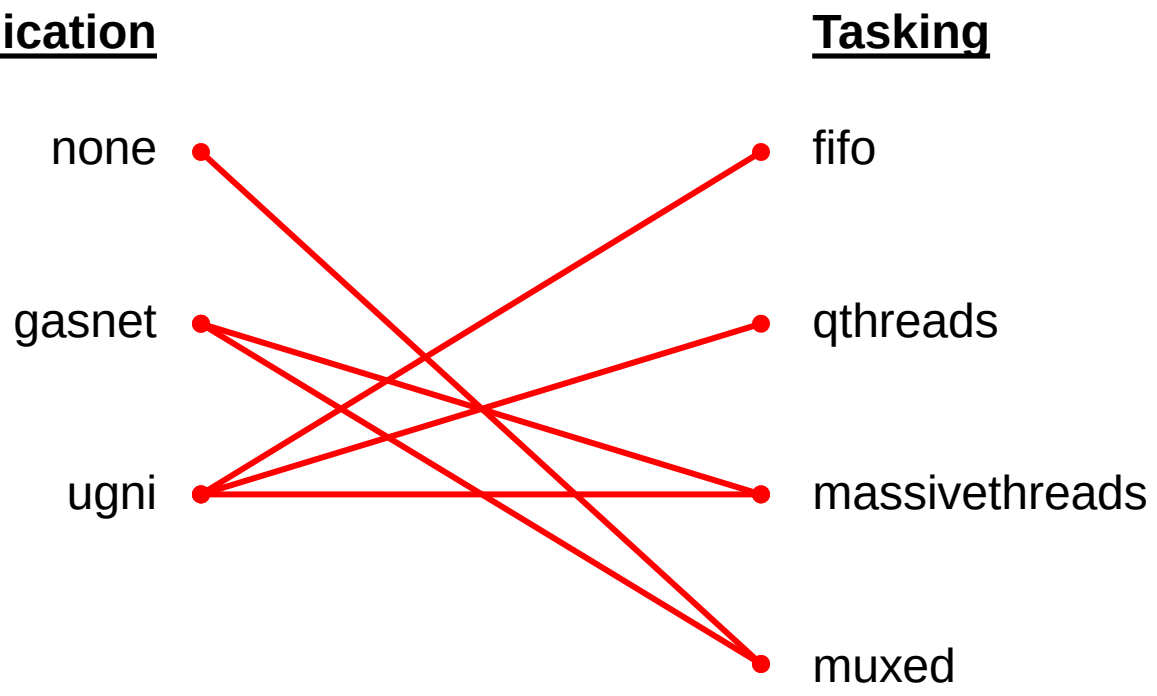






Cray RT Changes Impact #2: Interoperability

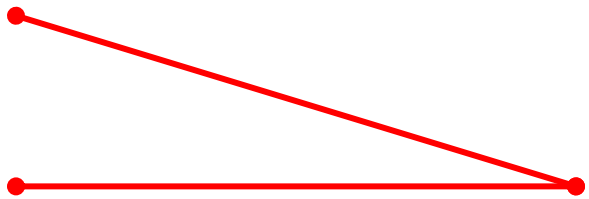
- **Combinations that did not work in Chapel 1.10:**



Cray RT Changes Impact #2: Interop

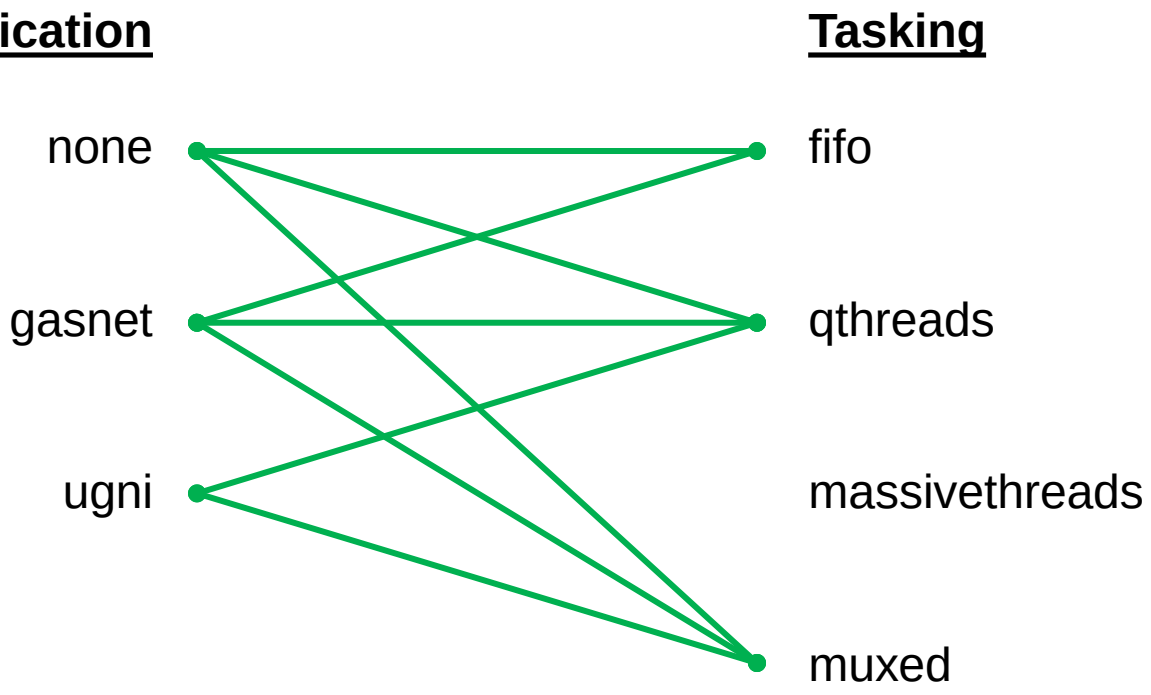
- **Combinations that do not work in Chapel 1.11:**

Communication	Tasking
none	fifo
gasnet	qthreads
ugni	massivethreads
	muxed



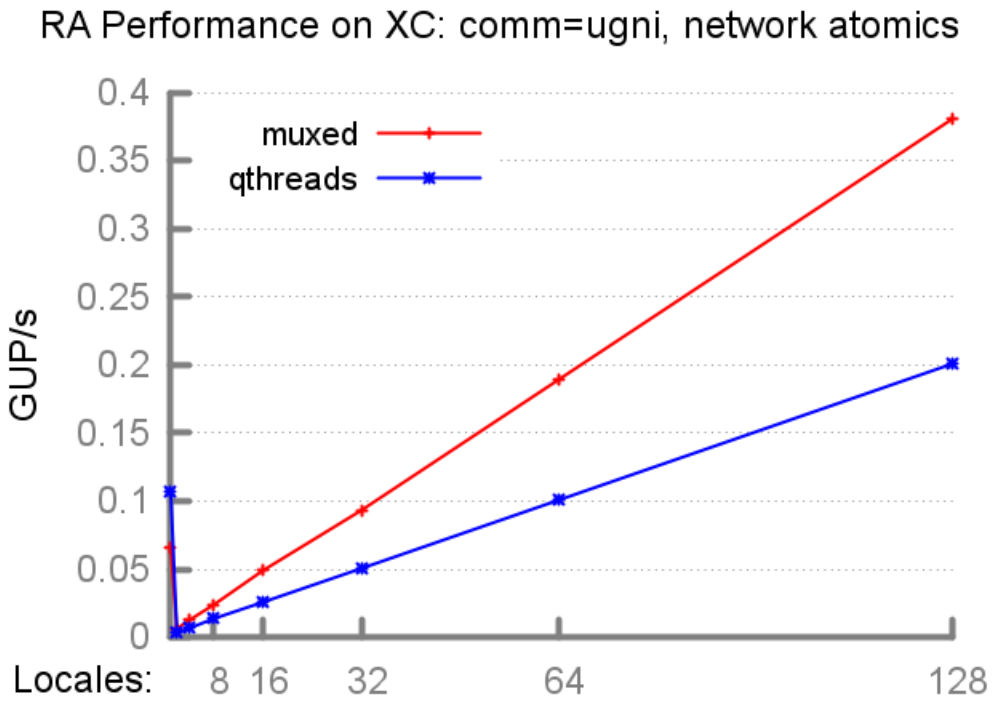
Cray RT Changes Impact #2: Interop

- **Combinations that are tested regularly:**



Cray RT Changes: Next Steps

- **ugni+qthreads allows us to consider retiring muxed**
  - Qthreads has NUMA support, wider use, etc.
  - Would save maintenance
  - (Still need comm=ugni, due to performance)
- **But need up-front work**
  - ugni+muxed fine-grained compute/comm overlap very effective
  - Would need to replicate in qthreads without revealing Cray IP
  - Exploring possible solutions now



Launcher Improvements

Launcher Improvements

slurm-srun:

- Added a CHPL_LAUNCHER_NODELIST / --nodelist option
  - maps down to slurm's --nodelist option
- Removed use of expect scripts for interactive mode
  - no longer needed; added maintenance and testing complexity

pbs-aprun:

- updated to use place/select syntax for PBS Professional (PBSPro)

Runtime Priorities and Next Steps

Runtime Priorities and Next Steps

- **Continue investigating retiring muxed tasking**
- **Address syscall intercept issue on Mac OS X**
- **Investigate Qthreads vs. OpenMP tasking differences**
  - experiments in CSU ICS paper saw more noise in Chapel runs
  - may be due to task-to-core mapping or races in spinning up tasks
- **Determine best practices for NUMA/KNL nodes**
- **Support optimization of reductions**

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/