Chapel: Why yes, we're still here

Salishan Conference on High-Speed Computing
April 24, 2013

Sung-Eun Choi
Chapel Team, Cray Inc.

What is Chapel?

- **An emerging (parallel) programming language**
  - Design and development led by Cray, Inc.
    - In collaboration with academia, labs, industry
  - Initiated under the DARPA HPCS program

Overall goal: Improve programmer productivity
- includes portability and performance

HPCS milestone: Come up with a way to improve the productivity of programming large-scale graph problems
- Traditional HPC problems have a “solution” (for now)
- We have absolutely no idea what to do about graph problems

Chapel's HPCS goal: SSCA#2

- **Demonstration of SSCA#2 (kernel 4) on a Cascade system**
  - Complete Chapel implementation of SSCA#2 benchmark
  - Run largest problem size that completes in a fixed time

- **Scalable Synthetic Compact Application #2**
  - Unstructured graph analysis benchmark
    - Kernel 4 computes betweenness centrality
  - Representative large data analytics problems
  - http://www.graphanalysis.org/benchmark/

Chapel's internal goal: No one trick pony

- **At the end of the HPCS program, we wanted a near complete programming language**
  - Modern base language implementation
    - OO support, generics, iterators, etc.
  - Abstractions for data- and task-parallelism
    - Arrays, domains (index sets) and distributions (aka domain maps)
    - Task creation and synchronization mechanisms
  - Abstractions to reason about locality
    - Data has a location (locale)
    - Migrate tasks to data or locales

- **Design with performance optimization in mind (even if we don't have enough time to implement the optimizations)**

Chapel's HPCS Scorecard

- ✓ **Graph-representation independent implementation of SSCA#2 (four working representations)**
  - generics, iterators, base language
- ✓ **R-MAT (recursive matrix) graph representation**
  - Distributed dense arrays (node list)
  - Associative domains (edge lists)
- ✓ **Latency hiding for fine grained communication**
  - Cray XE/XK/XC custom tasking/threading layers
  - Cray Gemini/Aries custom communication layers
- ✓ **Optimized remote memory operations**
  - Use of network AMOs
- ✓ **Other performance optimizations**
  - Hand implementation of task-private variables and associated support
  - Manual optimization to make up for lack of optimization and/or conservative analysis

A few things that (sort of) got left behind

- Performance impact of NUMA nodes
- Performance of bulk-synchronous style codes
- Scalar performance
- Heterogeneous architectures
- Other benchmarks

A few things our collaborators worked on

- **Performance impact of NUMA nodes**
  - Qthreads (SNL), MassiveThreads (University of Tokyo)
- **Performance of bulk-synchronous style codes**
  - Bulk and bulk-strided transfers (University of Malaga)
- **Scalar performance**
  - Native processor atomics, externs, Quick I/O, etc. (LTS)
- **Heterogeneous architectures**
  - Chapel on GPUs (UIUC)
- **Other benchmarks**
  - Language shootout (LTS and our interns), LULESH (LLNL), MADNESS (ORNL)

So, what next?

Chapel: The language everyone loves but no one uses yet

Salishan Conference on High-Speed Computing
April 24, 2013

Sung-Eun Choi
Chapel Team, Cray Inc.

CHUG: The Chapel User group

- **Today most users are developers**
  - Large parts of the implementation are written in Chapel
  - Notable exception: educators
- **The general sentiment among current non-users is that Chapel is very close to what they want**
  - We've achieved acceptance without adoption
- **To gain adoption we need to provide**
  - Better performance
  - Hardened implementation
  - Assurance of longevity

Chapel: The next five (or so) years

- **Ramp up staffing**
- **Fill in the gaps**
  - RAII and other OO stuff, exception handling, eurekas, task teams, etc.
- **Address heterogeneity**
  - Hierarchical locales support
- **Benchmarking**
  - More Proxy Apps, Chapel on HDFS
- **Improve overall performance**
  - Too much to mention here
- **Prepare to hand over governance to an external entity**
  - e.g., “The Chapel Foundation”

Chapel on HDFS

- **UW professional masters student project**
  - Day job is to run a Hadoop cluster
  - Hadoop has some serious shortcomings
  - Can Chapel be a more general alternative?

- **Project: Port a simple Hadoop MapReduce program to Chapel**
  - MR part written in Chapel
  - Interface with HDFS

Chapel on HDFS: Results

- **MR part pretty easy to write**
  - Strings are leaked (almost fixed)
  - Associative domain performance not good enough
- **HDFS interface was accessible via the extern facilities**
  - Extern capability is still cumbersome
- **Summer intern will take over this project to address some of the issues**

Summary

- **The end of an era**
  - DARPA HPCS program did a nice job of setting up the Chapel project for success beyond the program itself
    - Well positioned for Big Data (whatever that means)
    - Strong fan base
- **The start of a new era**
  - Prepare for productization
    - Performance and general hardening
  - Set it free
    - Hand over governance