PARALLEL PROGRAMMING WITH CHAPEL

Brad Chamberlain
PNW PLSE, May 9, 2023
OR:
PERFORMANCE AT ANY COST?
HPC* AND 24H OF LE MANS

Brad Chamberlain
PNW PLSE, May 9, 2023

HPC = High Performance Computing
PARALLEL COMPUTING HAS BECOME UBIQUITOUS

Traditional parallel computing:
- supercomputers
- commodity clusters

Today:
- multicore processors
- GPUs
- cloud computing
OAK RIDGE NATIONAL LABORATORY’S FRONTIER SUPERCOMPUTER

• 74 HPE Cray EX cabinets
• 9,408 AMD CPUs, 37,632 AMD GPUs
• 700 petabytes of storage capacity, peak write speeds of 5 terabytes per second using Cray ClusterStor storage system
• HPE Slingshot networking cables providing 100 GB/s network bandwidth.

1.1 exaflops of performance.

Built by HPE, ORNL’s Frontier supercomputer is #1 on the TOP500.

62.68 gigaflops/watt power efficiency for ORNL’s TDS system, 52.23 gigaflops/watt power efficiency for full system.

Built by HPE, ORNL’s TDS and full system are ranked #2 & #6 on the Green500.

7.9 exaflops on the HPL-MxP benchmark (formerly HPL-AI).

Built by HPE, ORNL’s Frontier supercomputer is #1 on the HPL-MxP list.

A STRAINED(?) ANALOGY

Gosh, I bet those supercomputer users have some swanky programming languages...

Gosh, those Le Mans racers must have an enviable driving experience...
HPC BENCHMARKS USING CONVENTIONAL PROGRAMMING APPROACHES

STREAM TRIAD: C + MPI + OPENMP

```c
#include <hpcc.h>
#include <hpcc.h>

static int VectorSz;  
static double *a, *b, *c;

int HPCC_StarStream(HPCC_Params *params) {  
  int myRank, commSize, j;  
  MPI_Comm comm = MPI_COMM_WORLD;  
  MPI_Comm_size(comm, &commSize);  
  MPI_Comm_rank(comm, &myRank);  

  rv = HPCC_StarStream( params, 0 == myRank);  
  return rv;}

int HPCC_LocalVectorSize( params, int doIO) { 
  double scalar;  
  VectorSize = HPCC_LocalVectorSize( params, 3, sizeof(double), 0);
  a = HPCC_MALLOC( double, VectorSize );
  b = HPCC_MALLOC( double, VectorSize );
  c = HPCC_MALLOC( double, VectorSize );

  return 0;
}

if (a || b || c) {  
  if (c) HPCC_free(c);
  if (b) HPCC_free(b);
  if (a) HPCC_free(a);
  if (doIO) {  
    printf( outfile, "Failed to allocate memory (%d). VectorSize = ");
  }
  return 1;
}

if (params->omp) {  
  #ifdef _OPENMP
  #endif

  HPCC_free( a);
  HPCC_free( b);
  HPCC_free( c);

  return 1;
}
```

HPCC RA: MPI KERNEL

```c
/* Perform updates to main table. */
for (i=0; i<NUPDATE; i++) {  
  for (j=0; j < VectorSize; j++) {  
    c[j] = 1.0;
  }
}
```

```c
/* Finish everyone else up... */
for (i=0; i<NUPDATE; i++) {  
  for (j=0; j < VectorSize; j++) {  
    a[j] = b[j]*scalar;c[j];
  }
}
```
WHAT IS CHAPEL?

Chapel: A modern parallel programming language
- portable & scalable
- open-source & collaborative
- pioneered and developed in Seattle (Cray Inc. / HPE)

Goals:
- Support general parallel programming
- Make parallel programming at scale far more productive
HPC BENCHMARKS: CONVENTIONAL APPROACHES VS. CHAPEL

**STREAM TRIAD: C + MPI + OPENMP**

```cpp
use BlockDist;
config const m = 1000, alpha = 3.0;
const Dom = Block.createDomain([1..m]);
var A, B, C: [Dom] real;
B = 2.0;
C = 1.0;
A = B + alpha * C;
```

**HPCC RA: MPI KERNEL**

```cpp
forall (_, r) in zip(Updates, RASTream()) do
    T[r & indexMask].xor(r);
```
HIGHLIGHT #1: CHAPEL SUPPORT FOR GPUs

Typical 2018-era Chapel Talk:

- **Me:** “Chapel’s generality goal is to support any parallel algorithm on any parallel architecture.”
- **Audience Q:** “So… does Chapel support GPUs?”
- **Me (with head bowed in shame):** “Only through interoperability with CUDA/OpenCL/OpenACC/OpenMP/…”

More recently:

- We’re targeting GPUs using Chapel’s traditional features for parallelism and locality

Let’s build up to a simple, “low-level” example using Stream Triad…
stream-ep.chpl

```chpl
config var n = 1_000_000,
    alpha = 0.01;

var A, B, C: [1..n] real;
A = B + alpha * C;
```

Declare three arrays of size ‘n’

Whole-array operations compute Stream Triad in parallel

So far, this is simply a multi-core program

Nothing refers to remote locales (nodes), explicitly or implicitly
STREAM TRIAD: DISTRIBUTED MEMORY

```chpl
stream-ep.chpl

config var n = 1_000_000,
    alpha = 0.01;

coforall loc in Locales {
    on loc {
        var A, B, C: [1..n] real;
        A = B + alpha * C;
    }
}
```

- `'coforall' loops execute each iteration as an independent task
- the array of locales (nodes) on which this program is running
- have each task run 'on' its locale
- then run multi-core Stream, as before

This is a CPU-only program

Nothing refers to GPUs, explicitly or implicitly
STREAM TRIAD: DISTRIBUTED MEMORY, GPUS ONLY

This is a GPU-only program

Nothing other than coordination code runs on the CPUs

Use a similar ‘coforall’ + ‘on’ idiom to run a Triad concurrently on each of this locale’s GPUs

stream-ep.chpl

```chapel
config var n = 1_000_000,
    alpha = 0.01;

coforall loc in Locales {
    on loc {
        coforall gpu in here.gpus do on gpu {
            var A, B, C: [1..n] real;
            A = B + alpha * C;
        }
    }
}
```
STREAM TRIAD: DISTRIBUTED MEMORY, GPUS AND CPUS

stream-ep.chpl

config var n = 1_000_000,
    alpha = 0.01;

coforall loc in Locales {
    on loc {
        cobegin {
            coforall gpu in here.gpus do on gpu {
                var A, B, C: [1..n] real;
                A = B + alpha * C;
            }
            {  
                var A, B, C: [1..n] real;
                A = B + alpha * C;
            }  
        }
    }
}

‘cobegin { ... }’ creates a task per child statement

one task runs our multi-GPU triad

the other runs the multi-CPU triad

This program uses all CPUs and GPUs across all of your compute nodes
Performance vs. reference versions has become increasingly competitive over the past 4 months.
HIGHLIGHT #2: APPLICATIONS OF CHAPEL

CHAMPS: 3D Unstructured CFD
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal

Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD

ChOp: Chapel-based Optimization
INRIA, IMEC, et al.

ChplUltra: Simulating Ultralight Dark Matter
Nikhil Padmanabhan, J. Luna Zagorac, et al.
Yale University et al.

Lattice-Symmetries: a Quantum Many-Body Toolbox
Tom Westerhout
Radboud University

Nelson Luis Dias
The Federal University of Paraná, Brazil

RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance

ChapQG: Layered Quasigeostrophic CFD
Ian Grooms and Scott Bachman
University of Colorado, Boulder et al.

Nelson Luis Dias
The Federal University of Paraná, Brazil

RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance

ChapQG: Layered Quasigeostrophic CFD
Ian Grooms and Scott Bachman
University of Colorado, Boulder et al.

(images provided by their respective teams and used with permission)
What is it?
• 3D unstructured CFD framework for airplane simulation
• ~85k lines of Chapel written from scratch in ~3 years

Who wrote it?
• Professor Éric Laurendeau’s students + postdocs at Polytechnique Montreal

Why Chapel?
• students found it far more productive to use
• enabled them to compete with more established CFD centers
APPLICATIONS OF CHAPEL

CHAMPS: 3D Unstructured CFD
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal

Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD

ChOp: Chapel-based Optimization
INRIA, IMEC, et al.

ChplUltra: Simulating Ultralight Dark Matter
Nikhil Padmanabhan, J. Luna Zagorac, et al.
Yale University et al.

Lattice-Symmetries: a Quantum Many-Body Toolbox
Tom Westerhout
Radboud University

Nelson Luis Dias
The Federal University of Paraná, Brazil

RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance

ChapQG: Layered Quasigeostrophic CFD
Ian Grooms and Scott Bachman
University of Colorado, Boulder et al.

Chapel-based Hydrological Model Calibration
Marjan Asgari et al.
University of Guelph

CrayAI HyperParameter Optimization (HPO)
Ben Albrecht et al.
Cray Inc. / HPE

CHGL: Chapel Hypergraph Library
Louis Jenkins, Cliff Joslyn, Jesun Firoz, et al.
PNNL

(images provided by their respective teams and used with permission)
ARKOUDA: A PYTHON LIBRARY AND FRAMEWORK FOR INTERACTIVE HPC

Arkouda Client
(written in Python)

Arkouda Server
(written in Chapel)

User writes Python code in Jupyter, making familiar NumPy/Pandas calls
ARKOUDA SUMMARY

What is it?
- A client-server framework for interactive supercomputing in Python
- ~30k lines of Chapel and ~25k lines of Python, written since 2019
- Open-source: https://github.com/Bears-R-Us/arkouda

Who wrote it?
- Mike Merrill, Bill Reus, et al., US DoD

Why Chapel?
- ability to develop on laptop, deploy on supercomputer
- close to Pythonic
SCALABILITY OF ARKOUDA’S ARGSORT ROUTINE

HPE Cray EX (spring 2023)
- 114,688 cores of AMD Rome
- Slingshot-11 network (200 Gb/s)
- 28 TiB of 8-byte values
- 1200 GiB/s (24 seconds elapsed time)

HPE Apollo (summer 2021)
- 73,728 cores of AMD Rome
- HDR Infiniband network (100 Gb/s)
- 72 TiB of 8-byte values
- 480 GiB/s (2.5 minutes elapsed time)

A notable performance achievement in ~100 lines of Chapel
HIGHLIGHT #3: THE CHAPEL TEAM AT HPE HAS GROWN
SUMMARY

Chapel is unique among programming languages
• built-in features for scalable parallel computing make it HPC-ready
• supports clean, concise code relative to conventional approaches
• ports and scales from laptops to supercomputers

Vendor-neutral GPU support is maturing rapidly
• fleshes out an overdue aspect of “any parallel hardware”

Chapel is being used for productive parallel computing at scale
• users are reaping its benefits in practical, cutting-edge applications
• in diverse application domains: from physical simulation to data science

We’re interested in helping new users and fostering new collaborations
CHAPEL RESOURCES

Chapel homepage: https://chapel-lang.org
  • (points to all other resources)

Social Media:
  • Twitter: @ChapelLanguage
  • Facebook: @ChapelLanguage
  • YouTube: http://www.youtube.com/c/ChapelParallelProgrammingLanguage

Community Discussion / Support:
  • Discourse: https://chapel.discourse.group/
  • Gitter: https://gitter.im/chapel-lang/chapel
  • Stack Overflow: https://stackoverflow.com/questions/tagged/chapel
  • GitHub Issues: https://github.com/chapel-lang/chapel/issues
THANK YOU

https://chapel-lang.org
@ChapelLanguage