Vendor-Neutral GPU Programming in Chapel

Jade Abraham
April 16, 2024
GPUs are easy to find...but difficult to program

• AI has driven huge demand for GPUs

• Many GPU solutions are C/C++ based
  • Can be a non-starter for scientists to access GPU parallelism
  • Using distributed GPUs requires additional support (e.g. MPI)

Chapel is an open-source alternative for productive
distributed/shared memory GPU programming in a vendor-neutral way.
Chapel: A modern parallel programming language

- portable & scalable
- open-source & collaborative

Goals:
- Support general parallel programming
- Make parallel programming at scale far more productive
What is Chapel?

**Chapel works everywhere**
- you can develop on your laptop and have the code scale on a supercomputer
- runs on Linux laptops/clusters, Cray systems, MacOS, WSL, AWS, Raspberry Pi
- shown to scale on Cray networks (Slingshot, Aries), InfiniBand, RDMA-Ethernet

**Chapel makes distributed/shared memory parallel programming easy**
- data-parallel, locality-aware loops,
- ability to move execution to remote nodes,
- distributed arrays and bulk array operations

**Chapel is GPU-ready**
- clear, concise kernels
- the same Chapel features that target CPU parallelism target GPUs
- vendor neutral
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.

Your Application Here?

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.

ChaplUltra: Simulating Ultralight Dark Matter
Nikhil Padmanabhan, J. Luna Zagorac, et al.
Yale University 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)
Coding in Chapel
config const nThreads = here.maxTaskPar, nPerThread = 4;
const n = nThreads * nPerThread;
var Arr: [0..<n] int;

coforall tid in 0..<nThreads {
    const startIdx = tid * nPerThread;
    Arr[startIdx..#nPerThread] = tid;
}
Multithread, Single GPU

```javascript
config const nThreads = here.maxTaskPar, nPerThread = 4;
const n = nThreads * nPerThread;
var Arr: [0..<n] int;

coforall tid in 0..<nThreads do on here.gpus[0] {
    const startIdx = tid * nPerThread;
    var GpuArr = Arr[startIdx..#nPerThread];
    GpuArr = tid;
    Arr[startIdx..#nPerThread] = GpuArr;
}
```

- **Enable GPU diagnostics to visualize what is happening**
- **Migrate execution to a GPU**
- **Copy the array chunk between the host and device**
Demo
use GpuDiagnostics;

config const nGpus = here.gpus.size, nPerGpu = 4;
const n = nGpus * nPerGpu;
var Arr: [0..<n] int;

startVerboseGpu();
coforall gid in 0..<nGpus do on here.gpus[gid] {
    const startIdx = gid * nPerGpu;
    var GpuArr = Arr[startIdx..#nPerGpu];
    GpuArr = gid;
    Arr[startIdx..#nPerGpu] = GpuArr;
}
stopVerboseGpu();
Demo
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.

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.

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

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

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)
Coral Reef Spectral Biodiversity

1. Read in a (M x N) raster image of habitat data
2. Create a (P x P) mask to find all points within a given radius.
3. Convolve this mask over the entire domain and perform a weighted reduce at each location.

Algorithmic complexity: $O(MNP^3)$

Typically:
- $M, N > 10,000$
- $P \sim 400$
Coral Reef Spectral Biodiversity

```haskell
proc convolve(InputArr, OutputArr) { // 3D Input, 2D Output
    for ... {
        tonOfMath();
    }
}

proc main() {
    var InputArr: ...;
    var OutputArr: ...;

    convolve(InputArr, OutputArr);
}
```
Coral Reef Spectral Biodiversity

```plaintext
proc convolve(InputArr, OutputArr) {
// 3D Input, 2D Output
  foreach ... {
    tonOfMath();
  }
}

proc main() {
  var InputArr: ...;
  var OutputArr: ...;
  coforall loc in Locales do on loc {
    coforall gpu in here.gpus do on gpu {
      coforall task in 0..#numWorkers {
        var MyInputArr = InputArr[...];
        var MyOutputArr: ...;
        convolve(MyInputArr, MyOutputArr);
        OutputArr[...] = MyOutputArr;
      }
    }
  }
}
```

Using a different loop flavor to enable GPU execution.

Multi-node, multi-GPU, multi-thread parallelism are expressed using the same language constructs.

// use all nodes in parallel...
// using GPUs on this node in parallel...
// using numWorkers on this GPU in parallel.

High-level, intuitive array operations work across nodes and/or devices.
Coral Reef Spectral Biodiversity

```chapel
proc convolve(InputArr, OutputArr) {    // 3D Input, 2D Output
    foreach ... {
        tonOfMath();
    }
}
proc main() {
    var InputArr: ...;
    var OutputArr: ...;

    coforall loc in Locales do on loc {
        // using all nodes in parallel...
        coforall gpu in here.gpus do on gpu {
            // using GPUs on this node in parallel...
            coforall task in 0..#numWorkers {
                // using parallel tasks on this GPU.
                var MyInputArr = InputArr[...];
                var MyOutputArr: ...;
                convolve(MyInputArr, OutputArr);
                OutputArr[...] = MyOutputArr;
            }
        }
    }
}
```

Ready to run on multiple nodes on Frontier!

- 5x improvement going from 2 to 64 nodes
  - (from 16 to 512 GPUs)

- Straightforward code changes:
  - from sequential Chapel code
  - to GPU-enabled one
  - to multi-node, multi-GPU, multi-thread

- Scalability improvements coming soon!
Get Connected with Chapel

• ChapelCon – free virtual event (https://chapel-lang.org/ChapelCon24.html)
  • June 5th – Tutorial Day
  • June 6th – Coding Day
  • June 7th – Conference Day

• Come code with us!
  • Github - https://github.com/chapel-lang/chapel
  • Gitter - https://gitter.im/chapel-lang/chapel
  • Discourse - https://chapel.discourse.group
  • StackOverflow - https://stackoverflow.com/questions/tagged/chapel

• Follow us on social media
  • X - https://x.com/ChapelLanguage
  • Facebook - https://www.facebook.com/ChapelLanguage
  • LinkedIn - https://www.linkedin.com/company/chapel-programming-language
  • YouTube - https://www.youtube.com/@ChapelLanguage

Registration for ChapelCon
https://shorturl.at/hvEW1
Thank you