Hewlett Packard
Enterprise

HPSF Project Update: The Chapel Programming Language

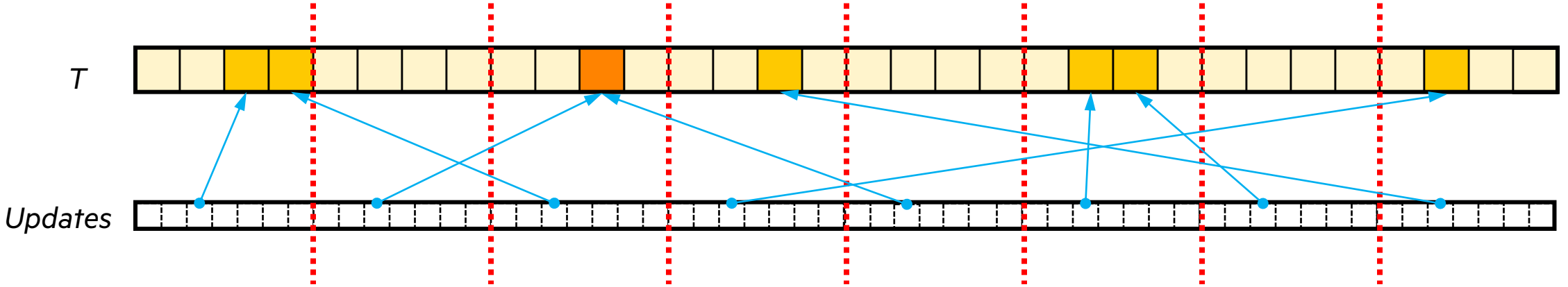
Michael Ferguson and Shreyas Khandekar, HPE

HPSFCon
May 5th, 2025


We want HPC to be more open, scalable, and productive
Let’s talk about an example

HPCC Random Access (RA)

Data Structure: distributed table



Computation: in parallel, update random table elements with random values

HPCC RA: MPI kernel

```
/* Perform updates to main table.  The scalar equivalent is:
 *
 *   for (i=0; i<NUPDATE; i++) {
 *     Ran = (Ran << 1) ^ (((s64Int) Ran < 0) ? POLY : 0);
 *     Table[Ran & (TABSIZE-1)] ^= Ran;
 *   }
 */

MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
          MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
 while (i < SendCnt) {
   /* receive messages */
   do {
     MPI_Test(&inreq, &have_done, &status);
     if (have_done) {
       if (status.MPI_TAG == UPDATE_TAG) {
         MPI_Get_count(&status, tparams.dtype64, &recvUpdates);
         bufferBase = 0;
         for (j=0; j < recvUpdates; j ++) {
           inmsg = LocalRecvBuffer[bufferBase+j];
           LocalOffset = (inmsg & (tparams.TableSize - 1)) -
                         tparams.GlobalStartMyProc;
           HPCC_Table[LocalOffset] ^= inmsg;
         }
       } else if (status.MPI_TAG == FINISHED_TAG) {
         NumberReceiving--;
       } else
         MPI_Abort( MPI_COMM_WORLD, -1 );
       MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
                 MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
     }
   } while (have_done && NumberReceiving > 0);
   if (pendingUpdates < maxPendingUpdates) {
     Ran = (Ran << 1) ^ ((s64Int) Ran < ZERO64B ? POLY : ZERO64B);
     GlobalOffset = Ran & (tparams.TableSize-1);
     if ( GlobalOffset < tparams.Top)
       WhichPe = ( GlobalOffset / (tparams.MinLocalTableSize + 1) );
     else
       WhichPe = ( (GlobalOffset - tparams.Remainder) /
                 tparams.MinLocalTableSize );
     if (WhichPe == tparams.MyProc) {
       LocalOffset = (Ran & (tparams.TableSize - 1)) -
                     tparams.GlobalStartMyProc;
       HPCC_Table[LocalOffset] ^= Ran;
     } else {
       HPCC_InsertUpdate(Ran, WhichPe, Buckets);
       pendingUpdates++;
     }
     i++;
   }
   else {
     MPI_Test(&outreq, &have_done, MPI_STATUS_IGNORE);
     if (have_done) {
       outreq = MPI_REQUEST_NULL;
       pe = HPCC_GetUpdates(Buckets, LocalSendBuffer, localBufferSize,
                            &peUpdates);
       MPI_Isend(&LocalSendBuffer, peUpdates, tparams.dtype64, (int)pe,
                 UPDATE_TAG, MPI_COMM_WORLD, &outreq);
       pendingUpdates -= peUpdates;
     }
   }
}
/* send remaining updates in buckets */
while (pendingUpdates > 0) {
   /* receive messages */
   do {
     MPI_Test(&inreq, &have_done, &status);
     if (have_done) {
       if (status.MPI_TAG == UPDATE_TAG) {
         MPI_Get_count(&status, tparams.dtype64, &recvUpdates);
         bufferBase = 0;
         for (j=0; j < recvUpdates; j ++) {
           inmsg = LocalRecvBuffer[bufferBase+j];
           LocalOffset = (inmsg & (tparams.TableSize - 1)) -
                         tparams.GlobalStartMyProc;
           HPCC_Table[LocalOffset] ^= inmsg;
         }
       } else if (status.MPI_TAG == FINISHED_TAG) {
         /* we got a done message.  Thanks for playing... */
         NumberReceiving--;
       } else {
         MPI_Abort( MPI_COMM_WORLD, -1 );
       }
       MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
                 MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
     }
   } while (have_done && NumberReceiving > 0);
     MPI_Test(&outreq, &have_done, MPI_STATUS_IGNORE);
     if (have_done) {
       outreq = MPI_REQUEST_NULL;
       pe = HPCC_GetUpdates(Buckets, LocalSendBuffer, localBufferSize,
                            &peUpdates);
       MPI_Isend(&LocalSendBuffer, peUpdates, tparams.dtype64, (int)pe,
                 UPDATE_TAG, MPI_COMM_WORLD, &outreq);
       pendingUpdates -= peUpdates;
     }
   }
}
/* send our done messages */
for (proc_count = 0 ; proc_count < tparams.NumProcs ; ++proc_count) {
   if (proc_count == tparams.MyProc) { tparams.finish_req[tparams.MyProc] =
                                       MPI_REQUEST_NULL; continue; }
   /* send garbage - who cares, no one will look at it */
   MPI_Isend(&Ran, 0, tparams.dtype64, proc_count, FINISHED_TAG,
             MPI_COMM_WORLD, tparams.finish_req + proc_count);
}
/* Finish everyone else up... */
while (NumberReceiving > 0) {
   MPI_Wait(&inreq, &status);
   if (status.MPI_TAG == UPDATE_TAG) {
     MPI_Get_count(&status, tparams.dtype64, &recvUpdates);
     bufferBase = 0;
     for (j=0; j < recvUpdates; j ++) {
       inmsg = LocalRecvBuffer[bufferBase+j];
       LocalOffset = (inmsg & (tparams.TableSize - 1)) -
                     tparams.GlobalStartMyProc;
       HPCC_Table[LocalOffset] ^= inmsg;
     }
   } else if (status.MPI_TAG == FINISHED_TAG) {
     /* we got a done message.  Thanks for playing... */
     NumberReceiving--;
   } else {
     MPI_Abort( MPI_COMM_WORLD, -1 );
   }
   MPI_Irecv(&LocalRecvBuffer, localBufferSize, tparams.dtype64,
             MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &inreq);
}

MPI_Waitall( tparams.NumProcs, tparams.finish_req, tparams.finish_statuses);
```

HPCC RA: Chapel vs. MPI kernel comment

Chapel Kernel

```
forall (_, r) in zip(Updates, RAStream()) do
  T[r & indexMask].xor(r);
```

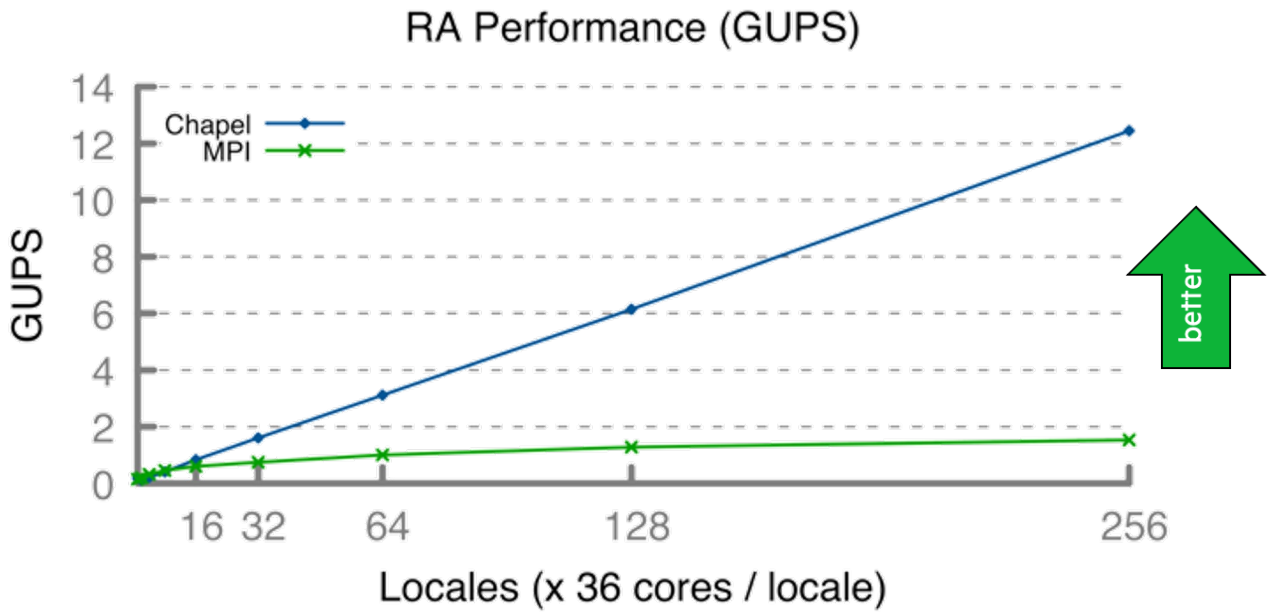
MPI Comment

```
/* Perform updates to main table. The scalar equivalent is:
 *
 *      for (i=0; i<NUPDATE; i++) {
 *        Ran = (Ran << 1) ^ (((s64Int) Ran < 0) ? POLY : 0);
 *        Table[Ran & (TABSIZE-1)] ^= Ran;
 *      }
 */
```

RA in C + MPI + OpenMP vs. Chapel

HPCC RA: MPI KERNEL

```
…
forall (_, r) in zip(Updates, RAStream()) do
  T[r & indexMask].xor(r);
…
```



So, what is this language, and why does it let us write compact, fast, and scalable programs?

The Chapel Parallel Programming Language

Chapel is a language designed for productive parallel programming, particularly on large-scale systems. Chapel is …

Easy to Use

"*We ask students at the master's degree to do stuff that would take 2 years and they do it in 3 months.*" Eric Laurendeau, Professor of Mechanical Engineering

Portable

HPE Cray EX, HPE Apollo, Cray XC, *nix systems, Mac, NVIDIA and AMD GPUs

Fast & Scalable

Achieved 8,500 GiB/s when sorting 256 TiB in 31 seconds on 8192 HPE Cray EX Nodes
… in just ~100 lines of Chapel

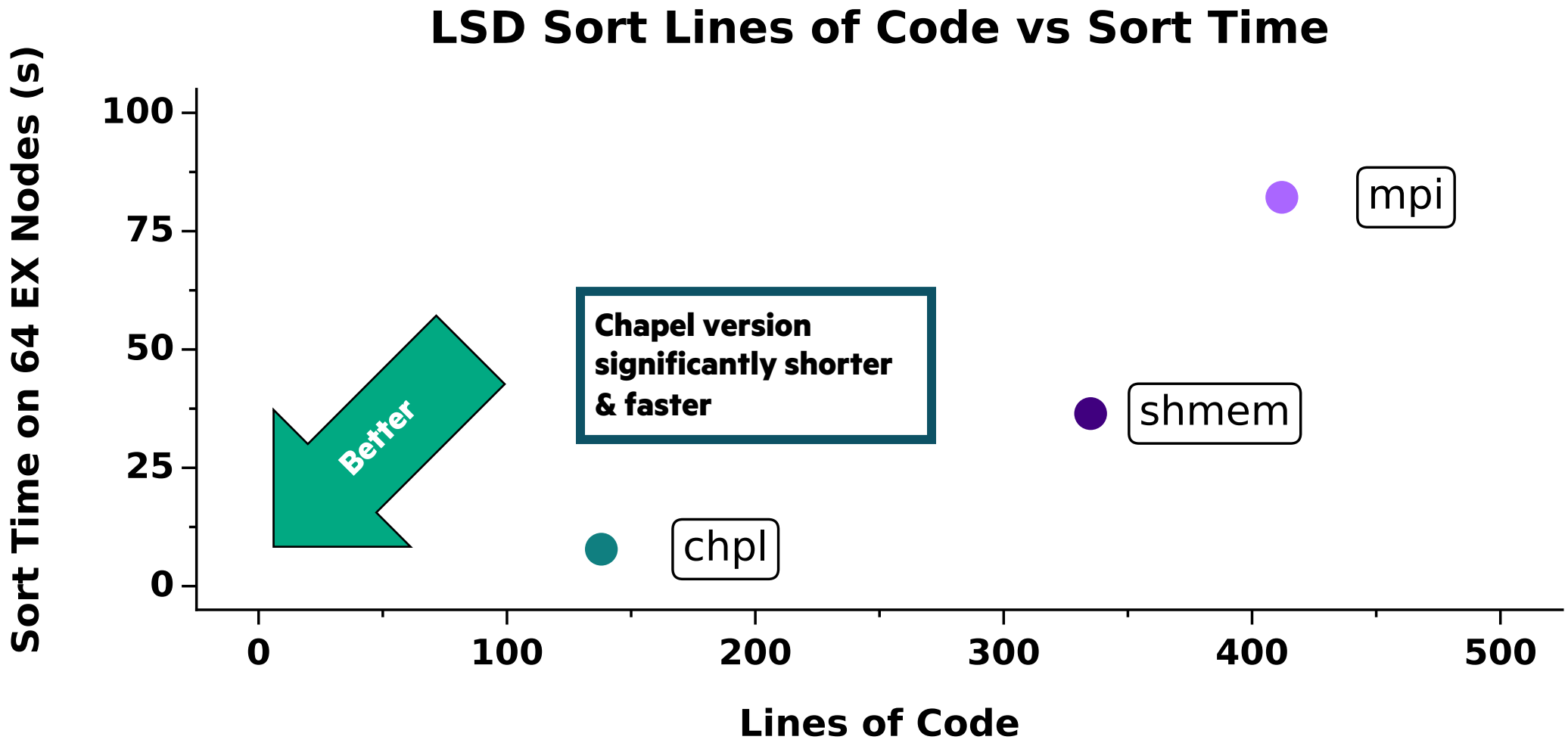
GPU-Ready

Real-world applications were ported on GPUs with few changes, and run on leadership-class systems such as Frontier and Perlmutter

Open source

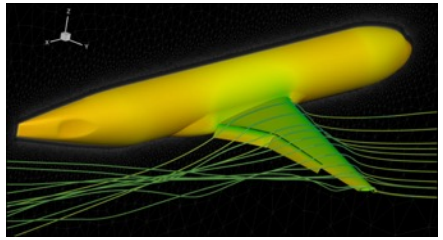
Apache 2.0 Licensed, hosted on GitHub & accepted into HPSF
Learn more and engage with the Chapel community at chapel-lang.org

Scaling Sort Performance of 64 Billion Elements on 64 EX nodes

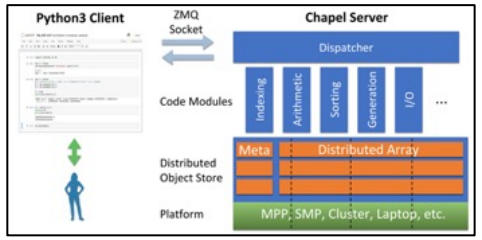


See https://github.com/mppf/distributed-lsb

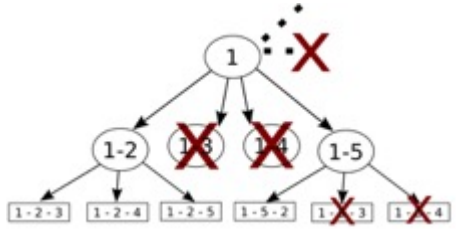
Applications of Chapel



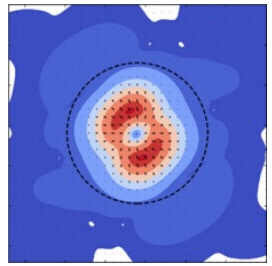
CHAMPS: 3D Unstructured CFD
Laurendeau, Bourgault-Côté, Parenteau, Plante, et al.
École Polytechnique Montréal



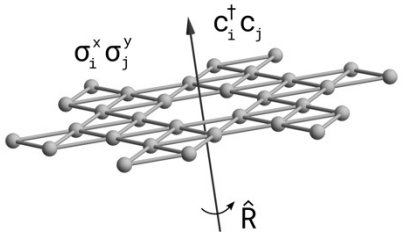
Arkouda: Interactive Data Science at Massive Scale
Mike Merrill, Bill Reus, et al.
U.S. DoD



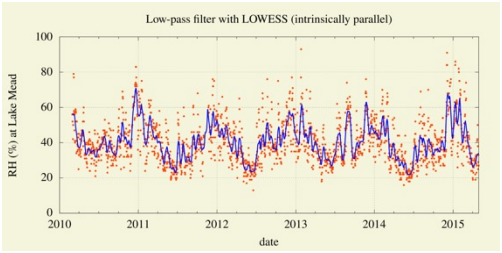
ChOp: Chapel-based Optimization
T. Carneiro, G. Helbecque, N. Melab, et al.
INRIA, IMEC, et al.



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

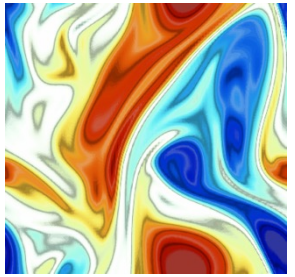


Lattice-Symmetries: a Quantum Many-Body Toolbox
Tom Westerhout
Radboud University

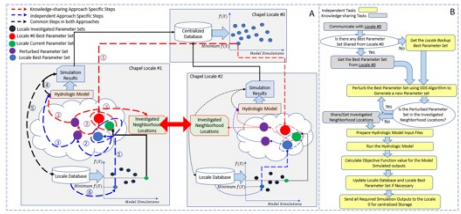


Desk dot chpl: Utilities for Environmental Eng.
Nelson Luis Dias
The Federal University of Paraná, Brazil

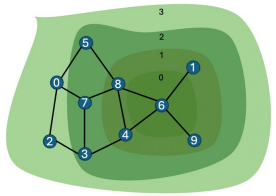
RapidQ: Mapping Coral Biodiversity
Rebecca Green, Helen Fox, Scott Bachman, et al.
The Coral Reef Alliance



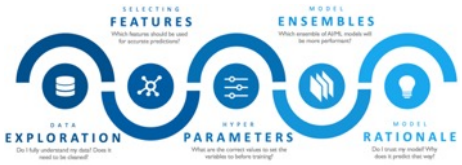
ChapQG: Layered Quasigeostrophic CFD
Ian Grooms and Scott Bachman
University of Colorado, Boulder et al.



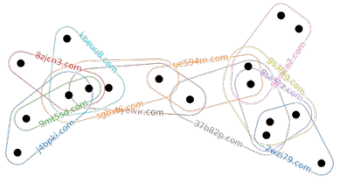
Chapel-based Hydrological Model Calibration
Marjan Asgari et al.
University of Guelph



Arachne Graph Analytics
Bader, Du, Rodriguez, et al.
New Jersey Institute of Technology



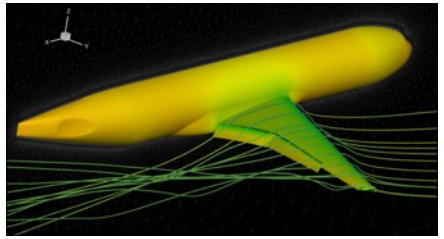
CrayAI HyperParameter Optimization (HPO)
Ben Albrecht et al.
Cray Inc. / HPE



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

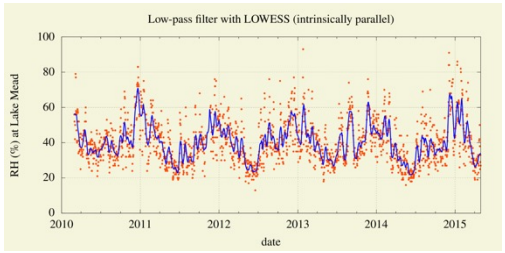
[images provided by their respective teams and used with permission]

Productivity Across Diverse Application Scales (code and system size)

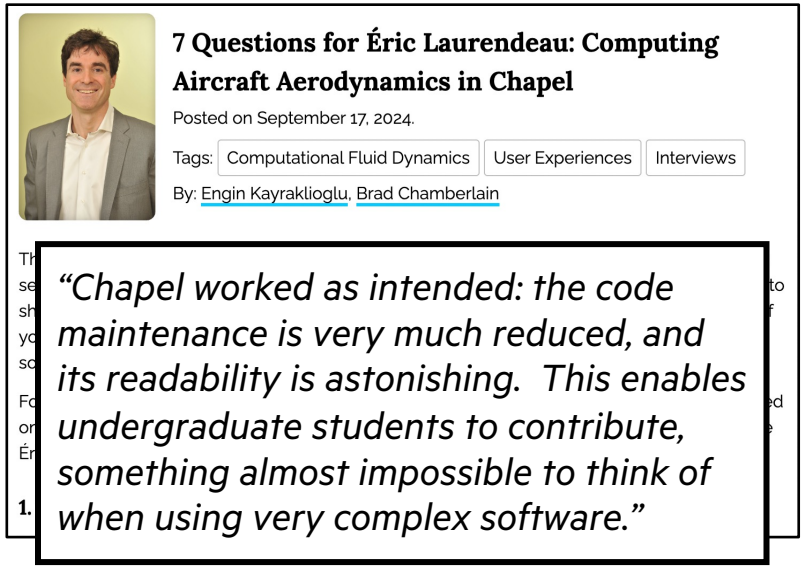


Computation: Aircraft simulation / CFD
Code size: 100,000+ lines
Systems: Desktops, HPC systems

Computation: Coral reef image analysis
Code size: ~300 lines
Systems: Desktops, HPC systems w/ GPUs



Computation: Atmospheric data analysis
Code size: 5000+ lines
Systems: Desktops w/ GPUs



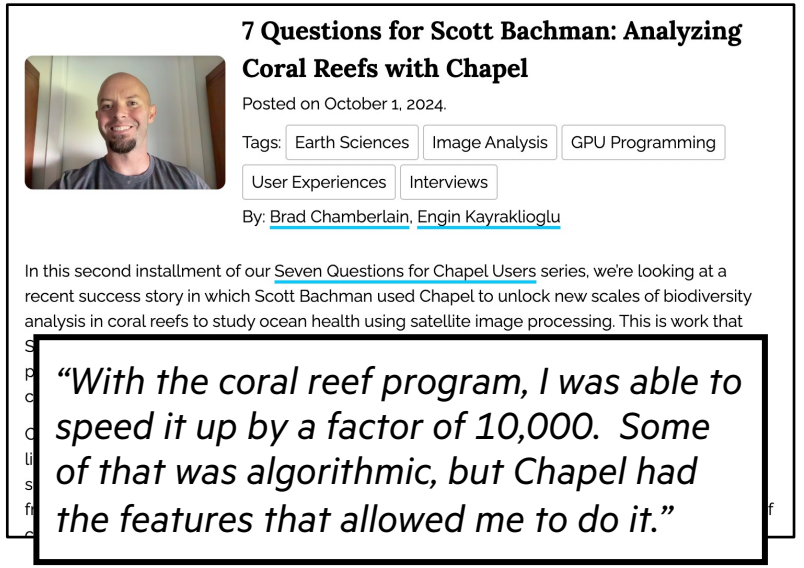
7 Questions for Éric Laurendeau: Computing Aircraft Aerodynamics in Chapel

Posted on September 17, 2024.

Tags: Computational Fluid Dynamics | User Experiences | Interviews

By: Engin Kayraklioglu, Brad Chamberlain

"Chapel worked as intended: the code maintenance is very much reduced, and its readability is astonishing. This enables undergraduate students to contribute, something almost impossible to think of when using very complex software."



7 Questions for Scott Bachman: Analyzing Coral Reefs with Chapel

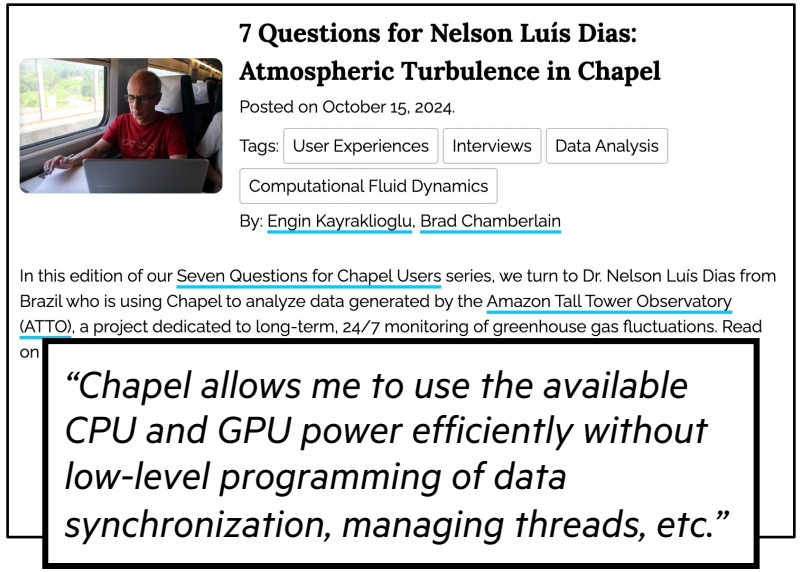
Posted on October 1, 2024.

Tags: Earth Sciences | Image Analysis | GPU Programming | User Experiences | Interviews

By: Brad Chamberlain, Engin Kayraklioglu

In this second installment of our Seven Questions for Chapel Users series, we're looking at a recent success story in which Scott Bachman used Chapel to unlock new scales of biodiversity analysis in coral reefs to study ocean health using satellite image processing. This is work that

"With the coral reef program, I was able to speed it up by a factor of 10,000. Some of that was algorithmic, but Chapel had the features that allowed me to do it."



7 Questions for Nelson Luís Dias: Atmospheric Turbulence in Chapel

Posted on October 15, 2024.

Tags: User Experiences | Interviews | Data Analysis | Computational Fluid Dynamics

By: Engin Kayraklioglu, Brad Chamberlain

In this edition of our Seven Questions for Chapel Users series, we turn to Dr. Nelson Luís Dias from Brazil who is using Chapel to analyze data generated by the Amazon Tall Tower Observatory (ATTO), a project dedicated to long-term, 24/7 monitoring of greenhouse gas fluctuations. Read on

"Chapel allows me to use the available CPU and GPU power efficiently without low-level programming of data synchronization, managing threads, etc."

[read this interview series at: https://chapel-lang.org/blog/series/7-questions-for-chapel-users/]

Chapel & HPSF: Shared Goals, Aligned Values

🧩 Lowering Barriers to HPC

- Easy to write a distributed-memory program
- Get started on a laptop, run on an HPC system

🌱 Aiding HPC Community Growth

- Open-source, contributor-friendly
- Used in new fields for HPC, including data science

🔧 Enabling HPC Development

- Projects like Arkouda, RapidQ, CHAMPS built in Chapel
- Chapel users are solving real world problems today!

🌍 Portable for Diverse Hardware

- HPC, cloud, desktop — CPU and GPU
- Language + runtime designed for extensibility

⚡ Performance + Productivity

- Competitive with MPI
- Shorter code for real applications -> easier to maintain

Where Does Chapel Run?

In the Browser:

- GitHub Codespaces
- Attempt This Online (ATO)

Laptops/Desktops:

- Linux/UNIX
- Mac OS X
- Windows with WSL

HPC Systems:

- Commodity clusters
- HPE/Cray supercomputers, such as:
  - El Capitan
  - Frontier
  - Perlmutter
  - Piz Daint
  - Polaris
  - …
- Other vendors' supercomputers

Cloud:

- AWS
- Other cloud providers (?)

CPUs:

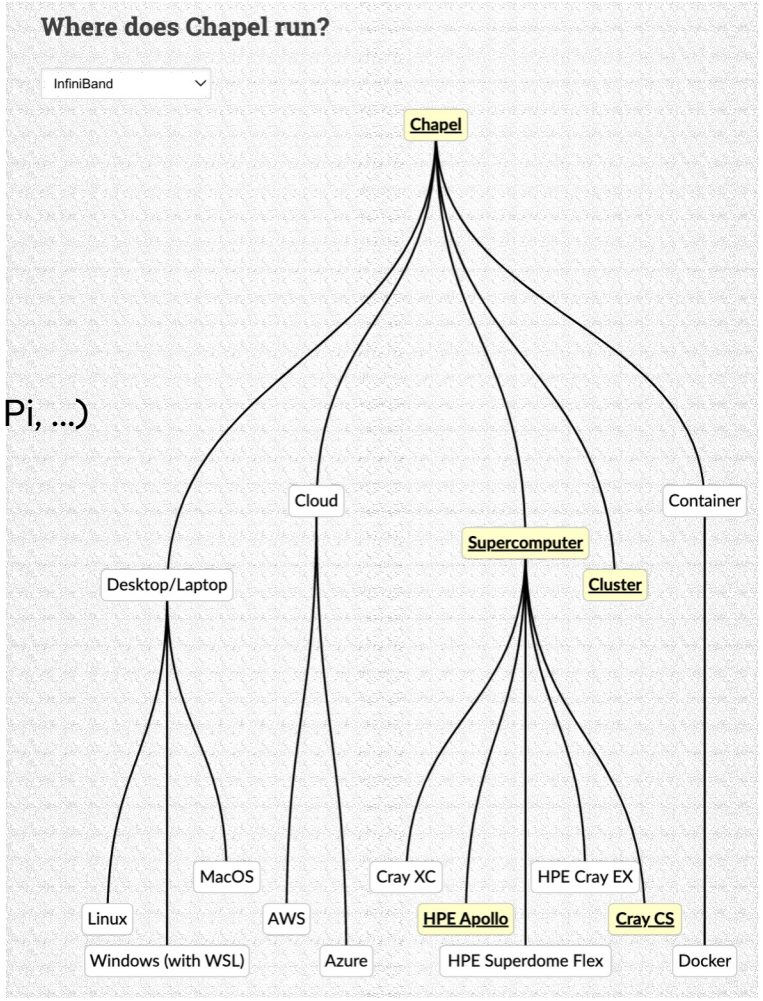
- Intel
- AMD
- ARM (M1/M2, Graviton, A64FX, Raspberry Pi, …)

GPUs:

- NVIDIA
- AMD

Networks:

- Slingshot
- InfiniBand
- AWS EFA
- Aries/Gemini
- Ethernet



https://chapel-lang.org/docs/usingchapel/portability.html

Chapel & HPSF: Status

- Accepted by the HPSF Technical Advisory Council (TAC) as an Established Project on Jan 27

- Taking steps to fully join HPSF:
  - Documented project governance
  - Drafted a technical charter
  - Revamped the project website & made the website repo public
  - Held a public design discussion on multidimensional array literals
  - The main weekly project meeting is now a public meeting
  - Still working through some of the legal details

- We are looking forward to meeting many of you this week!

Check out Chapel!

chapel-lang.org

github.com/chapel-lang/chapel