Chapel

Cray Cascade's High Productivity Language

Mary Beth Hribar
Steven Deitz
Brad Chamberlain

Cray Inc.

CUG 2006

Chapel Contributors

- **Cray Inc.**
  - Brad Chamberlain
  - Steven Deitz
  - Shannon Hoffswell
  - John Plevyak
  - Wayne Wong
  - David Callahan
  - Mackale Joyner
- **Caltech/JPL:**
  - Hans Zima
  - Roxana Diaconescu
  - Mark James

Chapel's Context

HPCS = High *Productivity* Computing Systems
(a DARPA program)

Overall Goal: Increase productivity by 10× by 2010

Productivity = Programmability
+ Performance
+ Portability
+ Robustness

Result must be…

…revolutionary, not evolutionary

…marketable product

Phase II Competitors (7/03-7/06): Cray (Cascade), IBM, Sun

Chapel Design Objectives

- a *global view* of computation
- support for general parallelism
  - data- and task-parallel; nested parallelism
- clean separation of algorithm and implementation
- broad-market language features
  - OOP, GC, latent types, overloading, generic functions/types, …
- data abstractions
  - sparse arrays, hash tables, sets, graphs, …
- good performance
- portability
- interoperability with existing codes

Outline

- ➤ Chapel Motivation & Foundations
  - ✓ Context and objectives for Chapel
  - ➤ Programming models and productivity
- Chapel Overview
- Chapel Activities and Plans

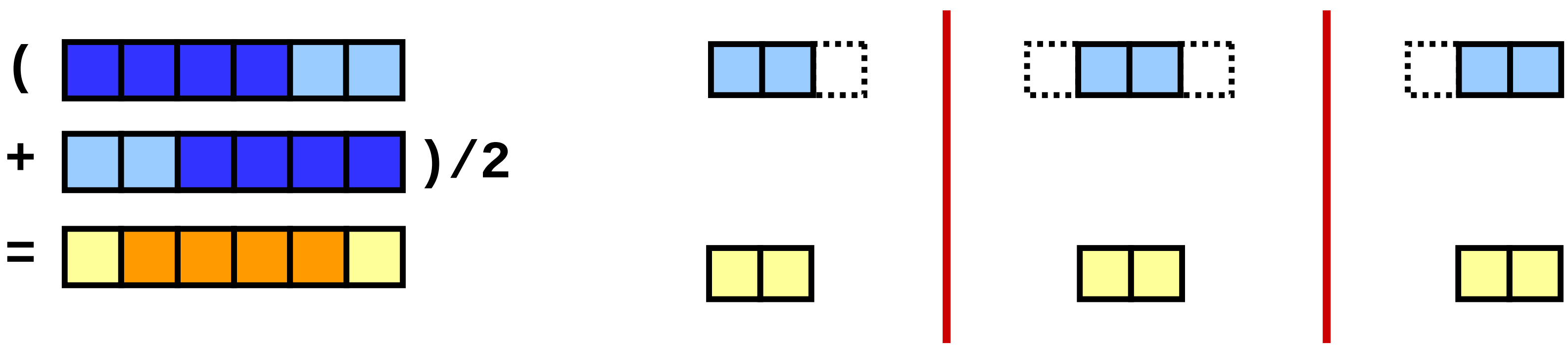
Parallel Programming Models

- *Fragmented Programming Models:*
  - Programmers *must* program on a task-by-task basis:
    - break distributed data structures into per-task chunks:
    - break work into per-task iterations/control flow
- *Global-view Programming Models:*
  - Programmers need not program task-by-task
    - access distributed data structures as though local
    - introduce parallelism using language keywords
    - burden of decomposition shifts to compiler/runtime
    - user may guide this process via language constructs

Global-view vs. Fragmented

- **Example:** “Apply 3-pt stencil to vector”

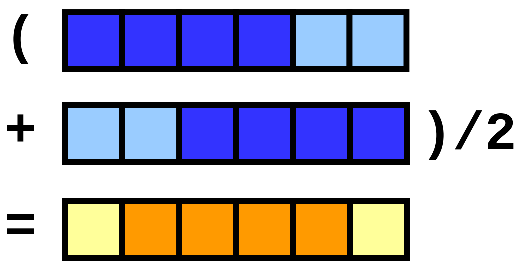
global-view | fragmented



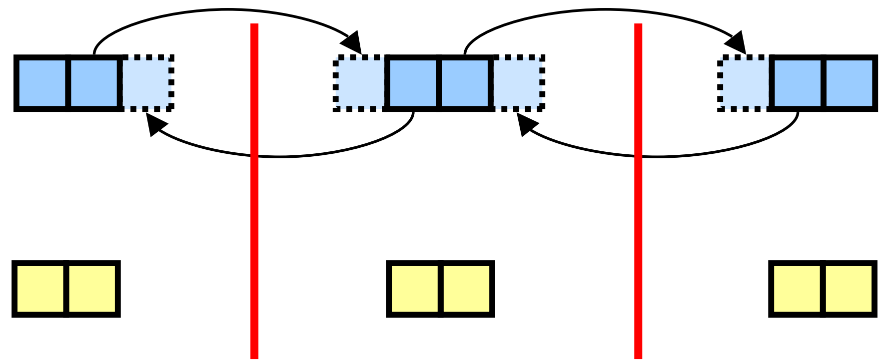
Global-view vs. Fragmented

- **Example:** "Apply 3-pt stencil to vector"

global-view

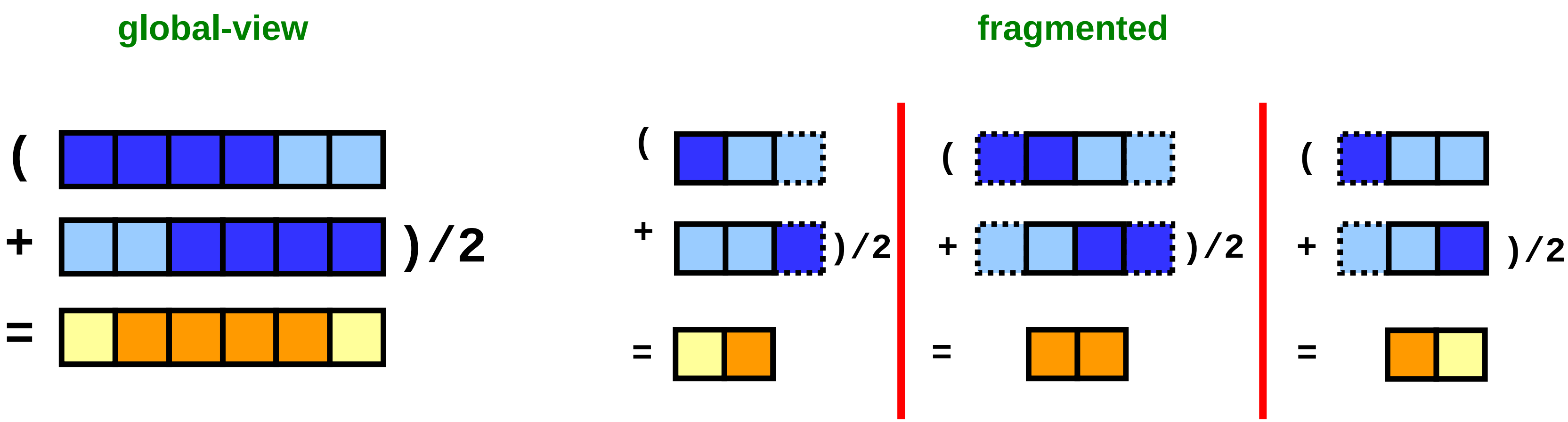


fragmented



Global-view vs. Fragmented

- **Example:** "Apply 3-pt stencil to vector"



Global-view vs. Fragmented

- **Example:** "Apply 3-pt stencil to vector"

global-view

```
var n: int = 1000;
var a, b: [1..n] float;

forall i in (2..n-1) {
  b(i) = (a(i-1) + a(i+1))/2;
}
```

fragmented

```
var n: int = 1000;
var locN: int = n/numProcs;
var a, b: [0..locN+1] float;
var innerLo: int = 1;
var innerHi: int = locN;

if (iHaveRightNeighbor) {
  send(right, a(locN));
  recv(right, a(locN+1));
} else {
  innerHi = locN-1;
}
if (iHaveLeftNeighbor) {
  send(left, a(1));
  recv(left, a(0));
} else {
  innerLo = 2;
}
forall i in (innerLo..innerHi) {
  b(i) = (a(i-1) + a(i+1))/2;
}
```

Assumes *numProcs* divides *n*; a more general version would require additional effort

Global-view vs. Fragmented

- **Example:** "Apply 3-pt stencil to vector"

fragmented (pseudocode + MPI)

```
var n: int = 1000, locN: int = n/numProcs;
var a, b: [0..locN+1] float;
var innerLo: int = 1, innerHi: int = locN;
var numProcs, myPE: int;
var retval: int;
var status: MPI_Status;

MPI_Comm_size(MPI_COMM_WORLD, &numProcs);
MPI_Comm_rank(MPI_COMM_WORLD, &myPE);
if (myPE < numProcs-1) {
  retval = MPI_Send(&(a(locN)), 1, MPI_FLOAT, myPE+1, 0, MPI_COMM_WORLD);
  if (retval != MPI_SUCCESS) { handleError(retval); }
  retval = MPI_Recv(&(a(locN+1)), 1, MPI_FLOAT, myPE+1, 1, MPI_COMM_WORLD, &status);
  if (retval != MPI_SUCCESS) { handleErrorWithStatus(retval, status); }
} else
  innerHi = locN-1;
if (myPE > 0) {
  retval = MPI_Send(&(a(1)), 1, MPI_FLOAT, myPE-1, 1, MPI_COMM_WORLD);
  if (retval != MPI_SUCCESS) { handleError(retval); }
  retval = MPI_Recv(&(a(0)), 1, MPI_FLOAT, myPE-1, 0, MPI_COMM_WORLD, &status);
  if (retval != MPI_SUCCESS) { handleErrorWithStatus(retval, status); }
} else
  innerLo = 2;
forall i in (innerLo..innerHi) {
  b(i) = (a(i-1) + a(i+1))/2;
}
```

Communication becomes geometrically more complex for higher-dimensional arrays

Fortran+MPI 3D NAS MG Stencil

```
      subroutine comm3(u,n1,n2,n3,kk)
      use caf_intrinsics

      implicit none

      include 'cafnpb.h'
      include 'globals.h'

      integer n1, n2, n3, kk
      double precision u(n1,n2,n3)
      integer axis

      if( .not. dead(kk) )then
         do  axis = 1, 3
            if( nprocs .ne. 1) then
               call sync_all()
               call give3( axis, +1, u, n1, n2, n3, kk )
               call give3( axis, -1, u, n1, n2, n3, kk )
               call sync_all()
               call take3( axis, -1, u, n1, n2, n3 )
               call take3( axis, +1, u, n1, n2, n3 )
            else
               call comm1p( axis, u, n1, n2, n3, kk )
            endif
         enddo
      else
         do  axis = 1, 3
            call sync_all()
            call sync_all()
         enddo
         call zero3(u,n1,n2,n3)
      endif
      return
      end


      subroutine give3( axis, dir, u, n1, n2, n3, k )
      use caf_intrinsics


      implicit none

      include 'cafnpb.h'
      include 'globals.h'

      integer axis, dir, n1, n2, n3, k, ierr
      double precision u( n1, n2, n3 )

      integer i3, i2, i1, buff_len,buff_id

      buff_id = 2 + dir
      buff_len = 0

      if( axis .eq.  1 )then
         if( dir .eq. -1 )then

            do  i3=2,n3-1
               do  i2=2,n2-1
                  buff_len = buff_len + 1
                  buff(buff_len,buff_id ) = u( 2,  i2,i3)
               enddo
            enddo

            buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
     >      buff(1:buff_len,buff_id)

         else if( dir .eq. +1 ) then

            do  i3=2,n3-1
               do  i2=2,n2-1
                  buff_len = buff_len + 1
                  buff(buff_len, buff_id ) = u( n1-1, i2,i3)
               enddo
            enddo

            buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
     >      buff(1:buff_len,buff_id)

         endif
      endif

      if( axis .eq.  2 )then
         if( dir .eq. -1 )then
            do  i3=2,n3-1
               do  i1=1,n1
                  buff_len = buff_len + 1
                  buff(buff_len, buff_id ) = u( i1,  2,i3)
               enddo
            enddo

       buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
     >      buff(1:buff_len,buff_id)

         else if( dir .eq. +1 ) then

            do  i3=2,n3-1
               do  i1=1,n1
                  buff_len = buff_len + 1
                  buff(buff_len,  buff_id )= u( i1,n2-1,i3)
               enddo
            enddo

            buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
     >      buff(1:buff_len,buff_id)

         endif
      endif

      if( axis .eq.  3 )then
         if( dir .eq. -1 )then

            do  i2=1,n2
               do  i1=1,n1
                  buff_len = buff_len + 1
                  buff(buff_len, buff_id ) = u( i1,i2,2)
               enddo
            enddo

            buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
     >      buff(1:buff_len,buff_id)

         else if( dir .eq. +1 ) then

            do  i2=1,n2
               do  i1=1,n1
                  buff_len = buff_len + 1
                  buff(buff_len, buff_id ) = u( i1,i2,n3-1)
               enddo
            enddo

            buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
     >      buff(1:buff_len,buff_id)

         endif
      endif

      return
      end


      subroutine take3( axis, dir, u, n1, n2, n3 )
      use caf_intrinsics


      implicit none

      include 'cafnpb.h'
      include 'globals.h'

      integer axis, dir, n1, n2, n3
      double precision u( n1, n2, n3 )

      integer buff_id, indx

      integer i3, i2, i1

      buff_id = 3 + dir
      indx = 0

      if( axis .eq.  1 )then
         if( dir .eq. -1 )then

            do  i3=2,n3-1
               do  i2=2,n2-1
                  indx = indx + 1
                  u(n1,i2,i3) = buff(indx, buff_id )
               enddo
            enddo

         else if( dir .eq. +1 ) then

            do  i3=2,n3-1
               do  i2=2,n2-1
                  indx = indx + 1
                  u(1,i2,i3) = buff(indx, buff_id )
               enddo
            enddo

         endif
      endif

      if( axis .eq.  2 )then
         if( dir .eq. -1 )then

            do  i3=2,n3-1
               do  i1=1,n1
                  indx = indx + 1
                  u(i1,n2,i3) = buff(indx, buff_id )
               enddo
            enddo

         else if( dir .eq. +1 ) then

            do  i3=2,n3-1
               do  i1=1,n1
                  indx = indx + 1
                  u(i1,1,i3) = buff(indx, buff_id )
               enddo
            enddo

         endif
      endif

      if( axis .eq.  3 )then
         if( dir .eq. -1 )then

            do  i2=1,n2
               do  i1=1,n1
                  indx = indx + 1
                  u(i1,i2,n3) = buff(indx, buff_id )
               enddo
            enddo

         else if( dir .eq. +1 ) then

            do  i2=1,n2
               do  i1=1,n1
                  indx = indx + 1
                  u(i1,i2,1) = buff(indx, buff_id )
               enddo
            enddo

         endif
      endif

      return
      end


      subroutine comm1p( axis, u, n1, n2, n3, kk )
      use caf_intrinsics


      implicit none

      include 'cafnpb.h'
      include 'globals.h'

      integer axis, dir, n1, n2, n3
      double precision u( n1, n2, n3 )

      integer i3, i2, i1, buff_len,buff_id
      integer i, kk, indx

      dir = -1

      buff_id = 3 + dir
      buff_len = nm2

      do  i=1,nm2
         buff(i,buff_id) = 0.0D0
      enddo


      dir = +1

      buff_id = 3 + dir
      buff_len = nm2

      do  i=1,nm2
         buff(i,buff_id) = 0.0D0
      enddo

      dir = +1

      buff_id = 2 + dir
      buff_len = 0

      if( axis .eq.  1 )then
         do  i3=2,n3-1
            do  i2=2,n2-1
               buff_len = buff_len + 1
               buff(buff_len, buff_id ) = u( n1-1,
     >           i2,i3)
            enddo
         enddo
      endif

      if( axis .eq.  2 )then
         do  i3=2,n3-1
            do  i1=1,n1
               buff_len = buff_len + 1
               buff(buff_len,  buff_id )= u( i1,n2-
     >           1,i3)
            enddo
         enddo
      endif

      if( axis .eq.  3 )then
         do  i2=1,n2
            do  i1=1,n1
               buff_len = buff_len + 1
               buff(buff_len, buff_id ) = u( i1,i2,n3-
     >           1)
            enddo
         enddo
      endif

      dir = -1

      buff_id = 2 + dir
      buff_len = 0

      if( axis .eq.  1 )then
         do  i3=2,n3-1
            do  i2=2,n2-1
               buff_len = buff_len + 1
               buff(buff_len,buff_id ) = u( 2,  i2,i3)
            enddo
         enddo
      endif

      if( axis .eq.  2 )then
         do  i3=2,n3-1
            do  i1=1,n1
               buff_len = buff_len + 1
               buff(buff_len, buff_id ) = u( i1,
     >           2,i3)
            enddo
         enddo
      endif

      if( axis .eq.  3 )then
         do  i2=1,n2
            do  i1=1,n1
               buff_len = buff_len + 1
               buff(buff_len, buff_id ) = u( i1,i2,2)
            enddo
         enddo
      endif

      do  i=1,nm2
         buff(i,4) = buff(i,3)
         buff(i,2) = buff(i,1)
      enddo

      dir = -1

      buff_id = 3 + dir
      indx = 0

      if( axis .eq.  1 )then
         do  i3=2,n3-1
            do  i2=2,n2-1
               indx = indx + 1
               u(n1,i2,i3) = buff(indx, buff_id )
            enddo
         enddo
      endif

      if( axis .eq.  2 )then
         do  i3=2,n3-1
            do  i1=1,n1
               indx = indx + 1
               u(i1,n2,i3) = buff(indx, buff_id )
            enddo
         enddo
      endif

      if( axis .eq.  3 )then
         do  i2=1,n2
            do  i1=1,n1
               indx = indx + 1
               u(i1,i2,n3) = buff(indx, buff_id )
            enddo
         enddo
      endif


      dir = +1

      buff_id = 3 + dir
      indx = 0

      if( axis .eq.  1 )then
         do  i3=2,n3-1
            do  i2=2,n2-1
               indx = indx + 1
               u(1,i2,i3) = buff(indx, buff_id )
            enddo
         enddo
      endif

      if( axis .eq.  2 )then
         do  i3=2,n3-1
            do  i1=1,n1
               indx = indx + 1
               u(i1,1,i3) = buff(indx, buff_id )
            enddo
         enddo
      endif

      if( axis .eq.  3 )then
         do  i2=1,n2
            do  i1=1,n1
               indx = indx + 1
               u(i1,i2,1) = buff(indx, buff_id )
            enddo
         enddo
      endif

      return
      end

      subroutine rprj3(r,m1k,m2k,m3k,s,m1j,m2j,m3j,k)
      implicit none
      include 'cafnpb.h'
      include 'globals.h'

      integer m1k, m2k, m3k, m1j, m2j, m3j,k

      double precision r(m1k,m2k,m3k), s(m1j,m2j,m3j)
      integer j3, j2, j1, i3, i2, i1, d1, d2, d3, j
      double precision x1(m), y1(m), x2,y2

      if(m1k.eq.3)then
        d1 = 2
      else
        d1 = 1
      endif

      if(m2k.eq.3)then
        d2 = 2
      else
        d2 = 1
      endif

      if(m3k.eq.3)then
        d3 = 2
      else
        d3 = 1
      endif

      do  j3=2,m3j-1
        i3 = 2*j3-d3
        do  j2=2,m2j-1
          i2 = 2*j2-d2
          do j1=2,m1j
            i1 = 2*j1-d1
            x1(i1-1) = r(i1-1,i2-1,i3  ) + r(i1-1,i2+1,i3  )
     >               + r(i1-1,i2,  i3-1) + r(i1-1,i2,  i3+1)
            y1(i1-1) = r(i1-1,i2-1,i3-1) + r(i1-1,i2-1,i3+1)
     >               + r(i1-1,i2+1,i3-1) + r(i1-1,i2+1,i3+1)
          enddo
          do  j1=2,m1j-1
            i1 = 2*j1-d1
            y2 = r(i1,  i2-1,i3-1) + r(i1,  i2-1,i3+1)
     >         + r(i1,  i2+1,i3-1) + r(i1,  i2+1,i3+1)
            x2 = r(i1,  i2-1,i3  ) + r(i1,  i2+1,i3  )
     >         + r(i1,  i2,  i3-1) + r(i1,  i2,  i3+1)
            s(j1,j2,j3) =
     >           0.5D0 * r(i1,i2,i3)
     >         + 0.25D0 * (r(i1-1,i2,i3) + r(i1+1,i2,i3) + x2)
     >         + 0.125D0 * ( x1(i1-1) + x1(i1+1) + y2)
     >         + 0.0625D0 * ( y1(i1-1) + y1(i1+1) )
          enddo
        enddo
      enddo
      j = k-1
      call comm3(s,m1j,m2j,m3j,j)
      return
      end
```

Chapel 3D NAS MG Stencil

```
param coeff: domain(1) = [0..3]; // for 4 unique weight values
param Stencil: domain(3) = [-1..1, -1..1, -1..1]; // 27-points

function rprj3(S, R) {
  param w: [coeff] float = (/0.5, 0.25, 0.125, 0.0625/);
  param w3d: [(i,j,k) in Stencil] float
           = w((i!=0) + (j!=0) + (k!=0));
  const SD = S.domain,
        Rstr = R.stride;

  S = [ijk in SD] sum reduce [off in Stencil]
                               (w3d(off) * R(ijk + Rstr*off));
}
```

Fragmented Language Summary

- Fragmented programming models…

  …manage per-task details in-line with the computation

  - per-task local bounds, data structures
  - communication, synchronization

  …are our main parallel programmability limiter today

Fragmented Language Summary

- Fragmented programming models…
  …tend to be easier to compile than global-view languages
  - at minimum, only need a good node compiler

  …deserve credit for the majority of the community's parallel application successes to date

Global-View Language Summary

- Single-processor languages are trivially global-view
  - Matlab, Java, Python, Perl, C, C++, Fortran, …
- Parallel global-view languages have been developed…
  - HPF (High Performance Fortran), ZPL, Sisal, NESL, Cilk, Cray MTA extensions to C/Fortran, …
- …yet most have not achieved widespread adoption
  - reasons why are as varied as the languages themselves
- Chapel has been designed…

  …to support global-view programming

  …with experience from preceding global-view languages

Outline

- ✓ Chapel Motivation & Foundations
- ➢ Chapel Overview
- Chapel Activities and Plans

What is Chapel?

- *Chapel:* Cascade High-Productivity Language
- Overall goal: "Solve the parallel programming problem"
  - simplify the creation of parallel programs
  - support their evolution to extreme-performance, production-grade codes
  - emphasize generality
- Motivating Language Technologies:
  - global-view multithreaded parallel programming
  - locality-aware programming

Multithreaded Parallel Programming

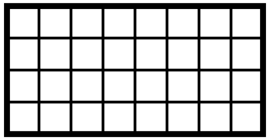
- Virtualization of threads
  - *i.e.*, no fork/join, naming of threads
- Abstractions for data and task parallelism
  - *data:* domains, arrays, iterators, …
  - *task:* cobegins, atomic transactions, sync variables, …
- Composition of parallelism
- Global view of computation, data structures

Data Parallelism: Domains

- *domain:* an index set
  - specifies size and shape of arrays
  - supports sequential and parallel iteration
  - potentially decomposed across locales
- Three main classes:
  - *arithmetic:* indices are Cartesian tuples
    - rectilinear, multidimensional, optionally strided and/or sparse
  - *indefinite:* indices serve as hash keys
    - supports hash tables, associative arrays, dictionaries
  - *opaque:* indices are anonymous
    - supports sets, graph-based computations
- Chapel's fundamental concept for data parallelism

Simple Domain Declarations

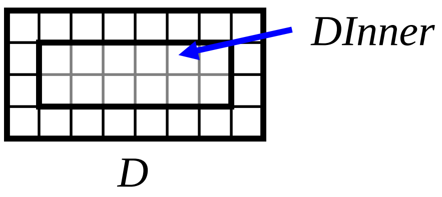
```
var m: int = 4;
var n: int = 8;

var D: domain(2) = [1..m, 1..n];
```



D

Simple Domain Declarations

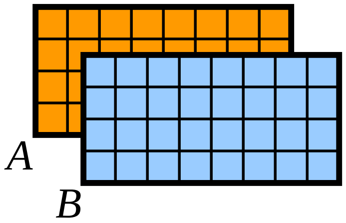
```
var m: int = 4;
var n: int = 8;

var D: domain(2) = [1..m, 1..n];
var DInner: subdomain(D) = [2..m-1, 2..n-1];
```



Domain Uses

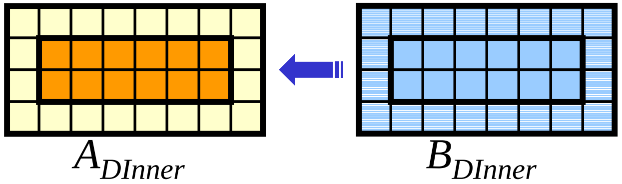
- Declaring arrays:

```
var A, B: [D] float;
```



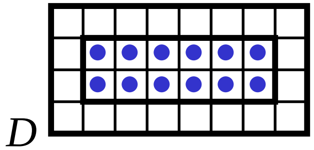
- Sub-array references:

```
A(DInner) = B(DInner);
```



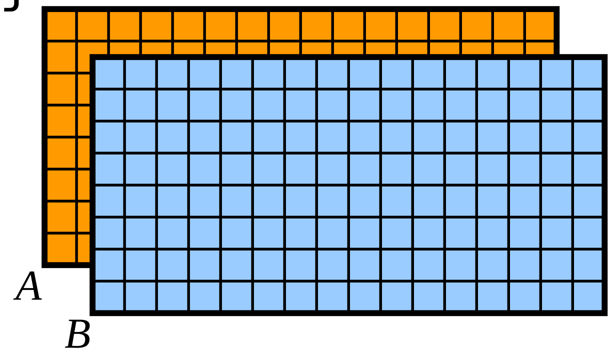
- Iteration:

```
    forall (i,j) in DInner { …A(i,j)… }
or: forall ind in DInner { …A(ind)… }
or: [ind in DInner] …A(ind)…
```



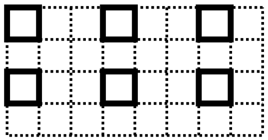
- Array reallocation:

```
D = [1..2*m, 1..2*n];
```



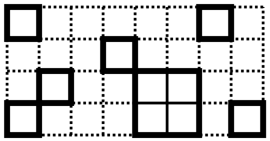
Other Arithmetic Domains

```
var StridedD: subdomain(D) = D by (2,3);
```



StridedD

```
var indexList: seq(index(D)) = …;
var SparseD: sparse subdomain(D) = indexList;
```



SparseD

Task Parallelism

- *co-begins:* indicate statements that may run in parallel:

```
computePivot(lo, hi, data);
cobegin {
  Quicksort(lo, pivot, data);
  Quicksort(pivot, hi, data);
}
```

```
cobegin {
  ComputeTaskA(…);
  ComputeTaskB(…);
}
```

- *atomic sections:* support atomic transactions

```
atomic {
  newnode.next = insertpt;
  newnode.prev = insertpt.prev;
  insertpt.prev.next = newnode;
  insertpt.prev = newnode;
}
```

- *sync and single-assignment variables:* synchronize tasks
  - similar to Cray MTA C/Fortran

Locality-aware Programming

- *locale:* architectural unit of storage and processing
- programmer specifies number of locales on executable command-line

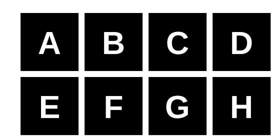
```
prompt> myChapelProg –nl=8
```

- Chapel programs are provided with built-in locale array:

```
const Locales: [1..numLocales] locale;
```

- Users may use it to create their own locale arrays:

```
var CompGrid: [1..GridRows, 1..GridCols] locale = …;
```



CompGrid

```
var TaskALocs: [1..numTaskALocs] locale = Locales(1..2);
var TaskBLocs: [1..numTaskBLocs] locale = Locales(3..numLocales);
```

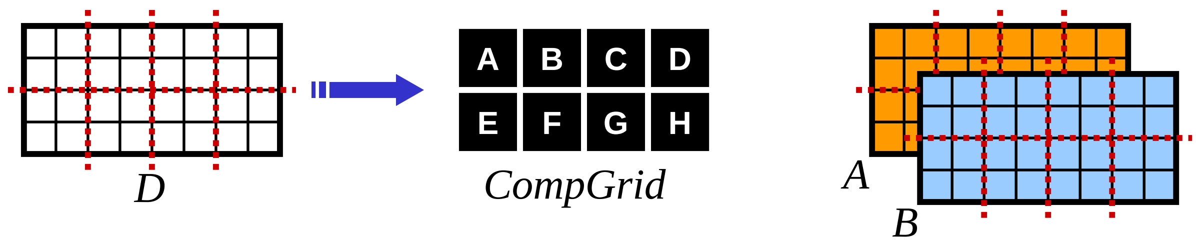
TaskALocs

TaskBLocs

Data Distribution

- domains may be distributed across locales

```
var D: domain(2) distributed(Block(2) on CompGrid) = …;
```



- Distributions specify…

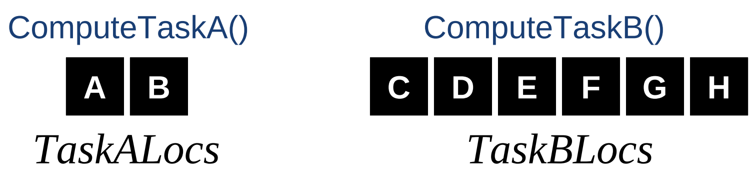
  …mapping of indices to locales

  …per-locale storage layout of domain indices and array elements
- Distributions implemented as a class hierarchy
  - Chapel provides a number of standard distributions
  - Users may also write their own ← **one of our biggest challenges**

Computation Distribution

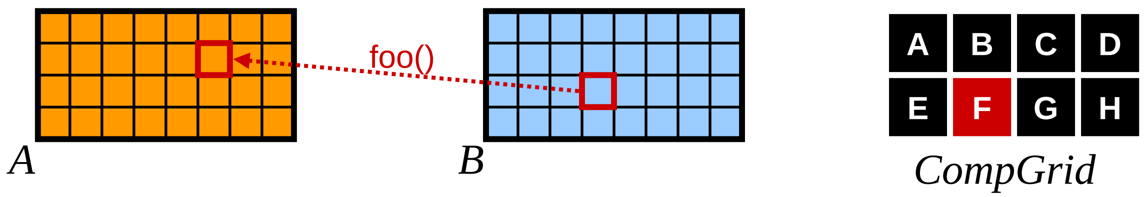
- “on” keyword binds computation to locale(s):

```
cobegin {
  on TaskALocs do ComputeTaskA(…);
  on TaskBLocs do ComputeTaskB(…);
}
```



- “on” can also be used in a data-driven manner:

```
forall (i,j) in D {
  on B(j/2,i*2) do A(i,j) = foo(B(j/2,i*2));
}
```



Chapel Challenges

- User Acceptance
  - True of any new language
  - Skeptical audience
- Commodity Architecture Implementation
  - Chapel designed with idealized architecture in mind
  - Clusters are not ideal in many respects
  - Results in implementation and performance challenges
- And many others as well…

Outline

- ✓ Chapel Motivation & Foundations
- ✓ Chapel Overview
- ➢ Chapel Activities and Plans

Phase II Activities

- 2003-2006:
  - Application studies to drive language design
    - HPCC, NPB, SSCA benchmarks
    - kernels from Cray customer applications
    - other kernels of interest (connected components, FMM)
  - Design and specification of Chapel language
  - Implementation work on portable Chapel prototype
  - Outreach to inform users and get feedback
    - government: LANL, Sandia, LLNL, ORNL, JPL, NITRD
    - conferences: ICS, PPoPP, LCPC, PGAS, HIPS, HPL, LaR
    - mainstream industry: Microsoft (w/ AMD attendance)
    - HPCS: biannual reviews, SW productivity meetings

What's next?

- HPCS phase III
  - July 2006 – December 2010
  - 2 vendors expected to be funded
  - proposals submitted May 5th
- HPCS Language Effort forking off
  - all 3 phase II language teams eligible for phase III
  - High Productivity Language Systems (HPLS) team
    - language experts/enthusiasts from national labs, academia
    - to study, evaluate the vendor languages, report to DARPA
    - July 2006 – December 2007
  - DARPA hopes…

    …that a language consortium will emerge from this effort

    …to involve mainstream computing vendors as well

    …to avoid repeating mistakes of the past (Ada, HPF, …)

Proposed Phase III Activities

- Short-term (2006-2007):
  - support user evaluations of Chapel
    - HPCS mission partners
    - HPLS language evaluation team
    - software productivity team
    - other potential user communities
  - continue Chapel implementation
  - capture application studies as tutorials
  - revise language as suggested by these activities
- Longer-term (2008-2010):
  - participate in HPLS consortium language efforts
  - help build support for language in community
  - fold HPLS language into Cascade compiler, tools

Summary

- Chapel is being designed to…
  …enhance programmer productivity
  …address a wide range of HEC algorithms
- Via high-level, extensible abstractions for…
  …multithreaded parallel programming
  …locality-aware programming
- Status:
  - *draft* language specification available at:
    http://chapel.cs.washington.edu
  - Open source implementation proceeding apace
  - Your feedback desired!