

HPCC STREAM and RA in Chapel Performance and Potential

Steve Deitz Cray Inc.

What is Chapel?



- A new parallel language
 - Under development at Cray Inc.
 - Supported through the DARPA HPCS program
- Goals
 - Improve programmer productivity
 - Improve the programmability of parallel computers
 - Match or improve performance of MPI/UPC/CAF
 - Provide better portability than MPI/UPC/CAF
 - Improve robustness of parallel codes
 - Support multi-core and multi-node systems

Outline



- What is Chapel?
- Chapel's Parallel Programming Model
- HPCC STREAM Triad in Chapel
- HPCC RA in Chapel
- Summary and Future Work



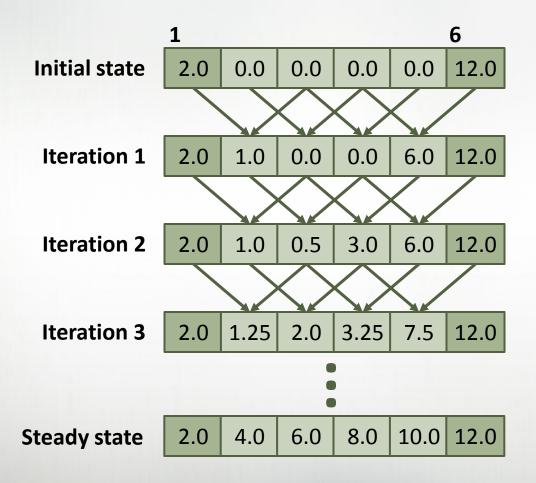
Fragmented vs. Global-View: Definitions

- Programming model
 The mental model of a programmer
- Fragmented models
 Programmers take point-of-view of a single processor/thread
- SPMD models (Single Program, Multiple Data)

 Fragmented models with multiple copies of one program
- Global-view models
 Programmers write code to describe computation as a whole



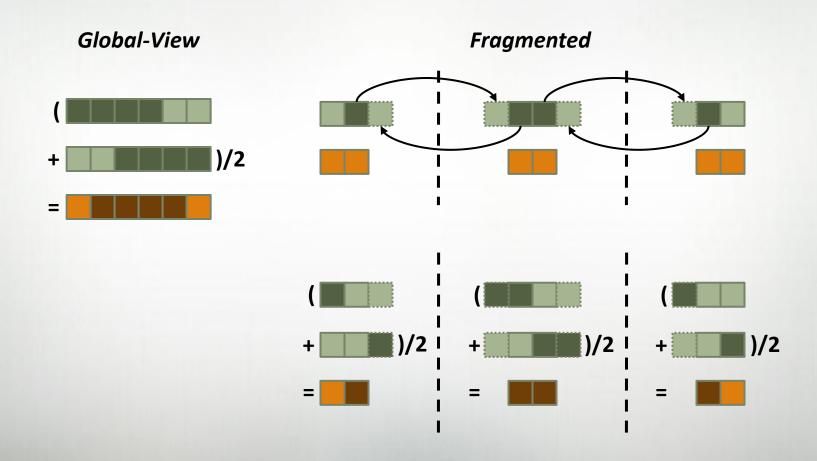






3-Point Stencil Example

Global-View vs. Fragmented Computation





3-Point Stencil Example: Code

Global-View vs. Fragmented Code

Global-View

```
def main() {
    var n = 1000;
    var A, B: [1..n] real;

    forall i in 2..n-1 do
        B(i) = (A(i-1)+A(i+1))/2;
}
```

Assumes p divides n

Fragmented

```
def main() {
  var n = 1000;
  var me = commID(), p = commProcs(),
    mvN = n/p, myLo = 1, myHi = myN;
  var A, B: [0..myN+1] real;
  if me < p {
    send (me+1, A(myN));
    recv (me+1, A (myN+1));
  } else myHi = myN-1;
  if me > 1 {
    send (me-1, A(1));
    recv (me-1, A(0));
  } else myLo = 2;
  for i in myLo..myHi do
    B(i) = (A(i-1)+A(i+1))/2;
```



NAS MG Stencil in Fortran + MPI, in Chapel

```
if(axis .eq. 1)then
do i3=2,n3-1
do i2=2,n2-1
         implicit none
                                                                                                                                                                                                                                                                                                                                                 46 m = -1
                                                                                                                                                                             implicit none
                                                                                               buff(1:buff_len,buff_id+1)[nbr(axis,dir
,k)] =
buff(1:buff_len,buff_id)
                                                                                                                                                                                                                                                                                                                                                 buff_id = 2 + dir
buff_len = 0
                                                                                           if(axis.eq. 2)then
if(dir.eq. -1)then
do in=2,n3-1
do in=1,n1
bufflen = bufflen + 1
buff(bufflen, bufflen + 1
2,13)
                                                                                                                                                                                                                                                                                                                                                            buff_len = buff_len + 1
buff(buff_len,buff_id) = u(2,
i2.i3)
                                                                                                                                                                                                                                                                                                                                                                                                                                           do il=1,nl
                                                                                                                                                                             integer i3, i2, i1
                                                                                                                                                                                                                                                                                                                                                                                                                                               indx = indx + 1
u(i1,1,i3) = buff(indx, buff_id)
                   call give3( axis, -1, u, n1, n2, n3, kk)
                                                                                                                                                                                                                                                                                                                                                 if( axis .eq. 2 )then
do i3=2,n3-1
                  call sync all()
call take3(axis, -1, u, n1, n2, n3)
call take3(axis, +1, u, n1, n2, n3)
else
call commlp(axis, u, n1, n2, n3, kk
                                                                                                                                                                                                                                                                integer axis, dir, n1, n2, n3
double precision u( n1, n2, n3 )
                                                                                                                                                                                                                                                                                                                                                                                                                                   if(axis.eq. 3)then
do i2=1,n2
do i1=1,n1
                                                                                    integer i3, i2, i1, buff_len,buff_id
integer i, kk, indx
                                                                                                                                                                                                                                                                                                                                                                                                                                              indx = indx + 1
                                                                                                   do i3=2,n3-1
do i1=1,n1
buff len = buff_len + 1
                                                                                                                                                                                              u(n1.i2.i3) = buff(indx.buff id)
                                                                                                                                                                                                                                                                                                                                                                                                                                            u(i1,i2,1) = buff(indx, buff_id)
enddo
                                                                                                                                                                                                                                                                                                                                                 if(axis .eq. 3)then
do i2=1,n2
do i1=1,n1
buff_len = buff_len + 1
             do axis = 1, 3
call sync_all()
call sync_all()
                                                                                                      buff(buff_len, buff_id)= u(
i1,n2-1,i3)
                                                                                                                                                                                  else if (dir .eg. +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
                                                                                                                                                                                                                                                                                                                                                           buff(buff_len, buff_id) = u(
i1,i2,2)
              enddo
call zero3(u,n1,n2,n3)
                                                                                                                                                                                                                                                               buff(i,buff_id) = 0.000
enddo
                                                                                                      buff(1:buff_len,buff_id+1)[nbr(axis,dir,k)] =
                                                                                                                                                                                                                                                                                                                                                                                                                                  subroutine
    rprj3(r,mlk,m2k,m3k,s,mlj,m2j,m3j,k)
implicit none
include 'cafnpb.h'
include 'globals.h'
                                                                                         > buff(1:buff_len,buff_id)
          subroutine give3 (axis. dir. u. nl. n2. n3. k
          use caf_intrinsics
                                                                                                                                                                                                                                                               do i=1,nm2
buff(i,buff_id) = 0.000
                                                                                                                                                                                                                                                                                                                                                 dir = -1
                                                                                                  do i2=1,n2
do i1=1,n1
buff_len = buff_len + 1
                                                                                                                                                                                                                                                                                                                                                                                                                                   double precision r(mlk,m2k,m3k),
s(mlj,m2j,m3j)
                                                                                                                                                                                     do i3=2.n3-1
                                                                                                                                                                                                                                                                                                                                                buff_id = 3 + dir
indx = 0
                                                                                                                                                                                     do i1=1,n1
indx = indx + 1
u(i1,n2,i3) = buff(indx, buff_id)
enddo
enddo
                                                                                                                                                                                                                                                                                                                                                                                                                                    integer j3, j2, j1, i3, i2, i1, d1, d2, d3, j
double precision x1(m), y1(m), x2,y2
                                                                                                     buff(buff_len, buff_id) = u(
i1.i2.2)
                                                                                                                                                                                                                                                                                                                                                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)
                                                                                                                                                                                                                                                                                                                                                                                                                                   else
dl = 1
                                                                                                                                                                                  else if (dir .eq. +1 ) then
                                                                                                     buff(1:buff_len,buff_id+1)[nbr(axis,dir
                                                                                                                                                                                     do (2=2 n2=1
                                                                                                                                                                                          i3=2,n3-1

do i1=1,n1

indx = indx + 1

u(i1,1,i3) = buff(indx, buff_id)

enddo
                                                                                               buff(1:buff len,buff id)
                                                                                                                                                                                                                                                                         buff len = buff len + 1
buff(buff_len, buff_id) = u(n1-1,
i2.i3)
                                                                                                                                                                                                                                                                                                                                                                                                                                   if (m2k.eq.3) then
         if(axis .eq. 1)then
if(dir .eq. -1)then
                                                                                                else if( dir .eq. +1 ) then
                                                                                                                                                                                                                                                                                                                                                                                                                                   else
d2 = 1
                                                                                                                                                                                                                                                                                                                                                          indx = indx + 1
u(i1,n2,i3) = buff(indx, buff_id)
enddo
                                                                                                                                                                                                                                                                                                                                                                                                                                    endif
                                                                                                     buff(buff_len, buff_id) = u(
i1,i2,n3-1)
                                                                                                                                                                             if(axis .eq. 3)then
if(dir .eq. -1)then
                                                                                                                                                                                                                                                                        buff_len = buff_len + 1
buff(buff_len, buff_id) = u(i1,n2-
1,i3)
                                                                                                                                                                                                                                                                                                                                                if( axis .eq. 3 ) then
    do i2=1,n2
    do i1=1,n1
    indx = indx + 1
        u(i1,i2,n3) = buff(indx, buff_id )
        rendete
                buff(1:buff_len,buff_id+1)[nbr(axis,dir
buff(1:buff_2).
                                                                                              buff(1:buff_len,buff_id+1)[nbr(axis,dir
,k)] =
buff(1:buff_len,buff_id)
                                                                                                                                                                                                                                                                                                                                                                                                                                   do j3=2,m3j-1
                                                                                                                                                                                              u(i1.i2.n3) = buff(indx, buff id)
                                                                                                                                                                                                                                                      if( axis .eq. 3 ) then
    do i2=1,n2
    do i1=1,n1
    buff_len = buff_len + 1
    buff_len = buff_len + buff_id ) = u(
    i1,i2,n3=1)
    enddo
              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( nl-
1, i2,i3)
                                                                                                                                                                                  else if (dir .eq. +1 ) then
                                                                                                                                                                                                                                                                                                                                                                                                                                         do j1=2,mlj
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-
                                                                                                                                                                                                                                                                                                                                                  buff_id = 3 + dir
indx = 0
                                                                                           subroutine take3 (axis, dir. u. nl. n2, n3)
def rprj3(S, R) {
```

```
\begin{array}{ll} 1,12,&13+1)\\ y_1(11-2)&=z(11-2,12-1,13-2))&=z(11-2,12-1,13-2))&=z(11-2,12-1,13-2))&=z(11-2,12-1,13-2))&=z(11-2,12-1,13-2))&=z(11-2,13-2))&=z(11-2,13-2))&=z(11-2,13-2))&=z(11,13-2))&=z(11,13-2,13-2))&=z(11,13-2,13-2))&=z(11,13-2,13-2))&=z(11,13-2,13-2)&=z(11,13-2,13-2))&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&=z(11,13-2,13-2)&
```

Outline



- What is Chapel?
- Chapel's Parallel Programming Model
- HPCC STREAM Triad in Chapel
- HPCC RA in Chapel
- Summary and Future Work

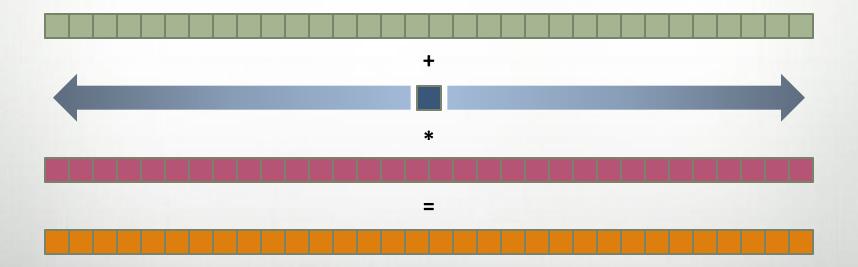


Introduction to STREAM Triad

Given: m-element vectors A, B, C

Compute: forall i in 1..m do

$$\mathbf{A}(i) = \mathbf{B}(i) + \boldsymbol{\alpha} * \mathbf{C}(i);$$



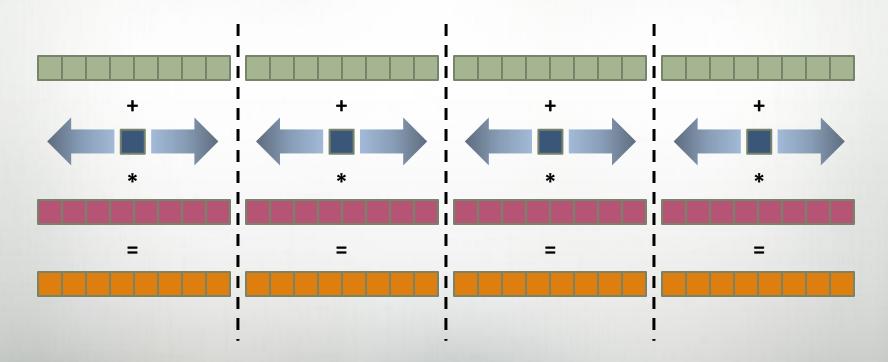


Distributed Parallelization of STREAM Triad

Given: *m*-element vectors *A*, *B*, *C*

Compute: forall i in 1..m do

$$\mathbf{A}(i) = \mathbf{B}(i) + \boldsymbol{\alpha} * \mathbf{C}(i);$$



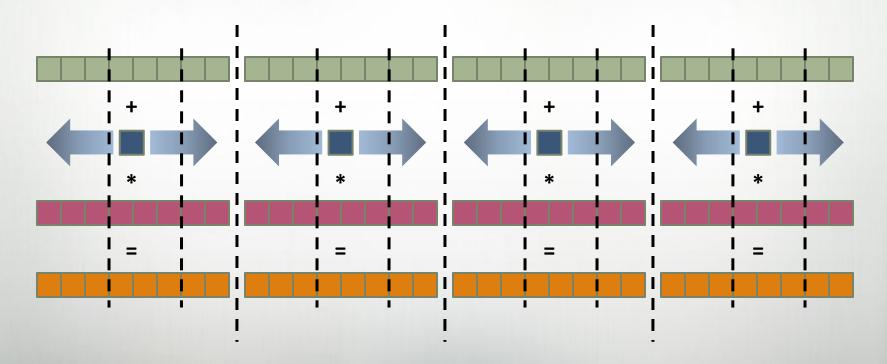


Further Parallelization of STREAM Triad

Given: *m*-element vectors *A*, *B*, *C*

Compute: forall i in 1..m do

$$\mathbf{A}(i) = \mathbf{B}(i) + \boldsymbol{\alpha} * \mathbf{C}(i);$$





STREAM Triad in Chapel: Single Locale

```
Given: m-element vectors A, B, C
Compute: forall i in 1..m do
              \mathbf{A}(i) = \mathbf{B}(i) + \boldsymbol{\alpha} * \mathbf{C}(i);
config const m: int(64) = ...;
const alpha: real = 3.0;
const ProblemSpace: domain(1,int(64)) = [1..m];
var A, B, C: [ProblemSpace] real;
forall i in ProblemSpace do
  A(i) = B(i) + alpha * C(i);
```



STREAM Triad in Chapel: Single Locale

Given: m-element vectors A, B, C

```
Compute: forall i in 1..m do
A(i) = B(i) + \alpha * C(i);
```

```
config const m: int(64) = ...;
const alpha: real = 3.0;
const ProblemSpace: domain(1,int(64)) = [1..m];
var A, B, C: [ProblemSpace] real;
```

```
A = B + alpha * C;
```

More concise variation using whole array operations



STREAM Triad in Chapel: Single Locale

a = b + alpha * c;

Variation that iterates directly over the arrays



STREAM Triad in Chapel: Multi-Locale

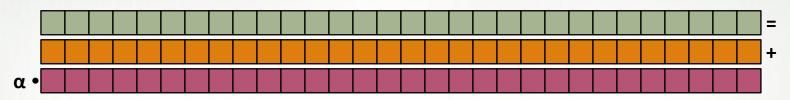
```
Given: m-element vectors A, B, C
Compute: forall i in 1..m do
               \mathbf{A}(\mathbf{i}) = \mathbf{B}(\mathbf{i}) + \boldsymbol{\alpha} * \mathbf{C}(\mathbf{i});
config const m: int(64) = ..., tpl = ...;
const alpha: real = 3.0;
const BlockDist = new Block(1,int(64),[1..m],tpl);
const ProblemSpace: domain(1, int(64))
                         distributed BlockDist = [1..m];
var A, B, C: [ProblemSpace] real;
forall (a,b,c) in (A,B,C) do
  a = b + alpha * c;
```



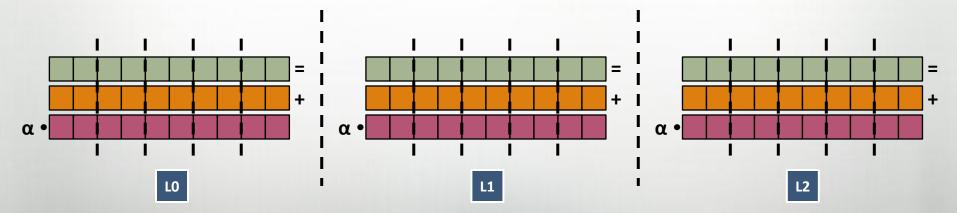


A "recipe" for distributed arrays that...

Instructs the compiler how to Map the global view...



...to a fragmented, per-processor implementation





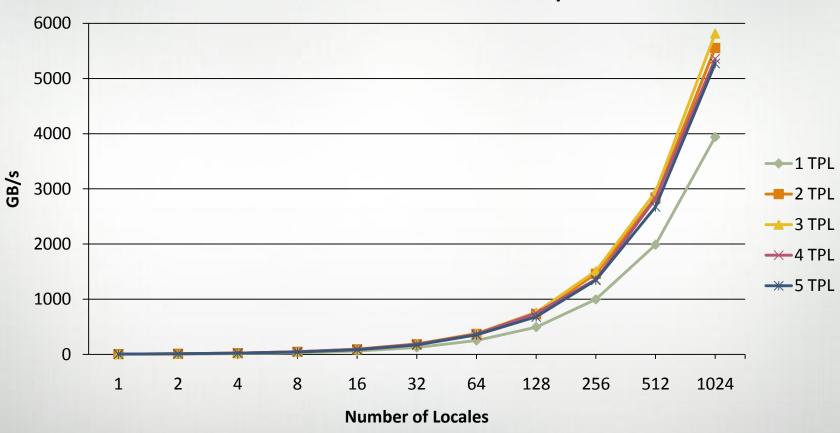
STREAM Triad in Chapel: Multi-Locale

```
Given: m-element vectors A, B, C
Compute: forall i in 1..m do
               \mathbf{A}(\mathbf{i}) = \mathbf{B}(\mathbf{i}) + \boldsymbol{\alpha} * \mathbf{C}(\mathbf{i});
config const m: int(64) = ..., tpl = ...;
const alpha: real = 3.0;
const BlockDist = new Block(1,int(64),[1..m],tpl);
const ProblemSpace: domain(1, int(64))
                         distributed BlockDist = [1..m];
var A, B, C: [ProblemSpace] real;
forall (a,b,c) in (A,B,C) do
  a = b + alpha * c;
```



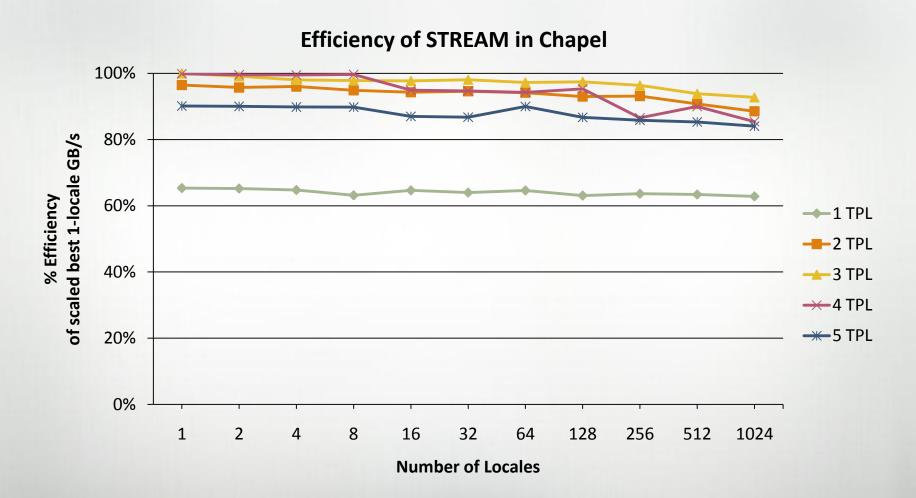








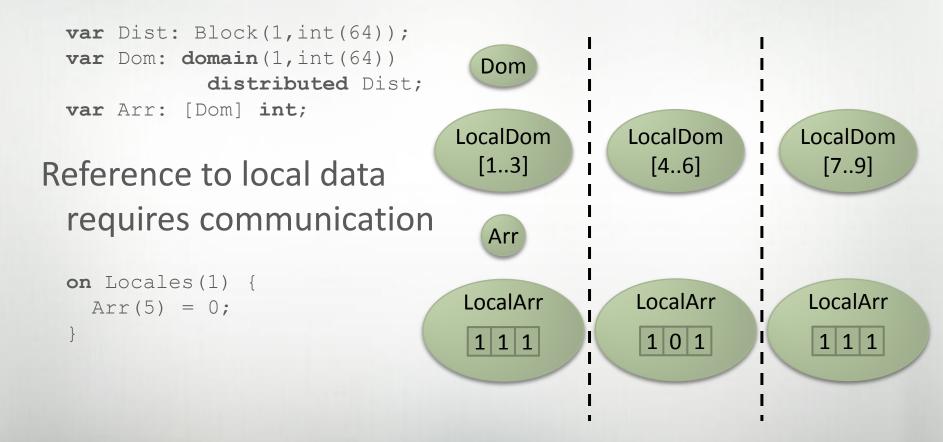








Simple example



Outline



- What is Chapel?
- Chapel's Parallel Programming Model
- HPCC STREAM Triad in Chapel
- HPCC RA in Chapel
- Summary and Future Work

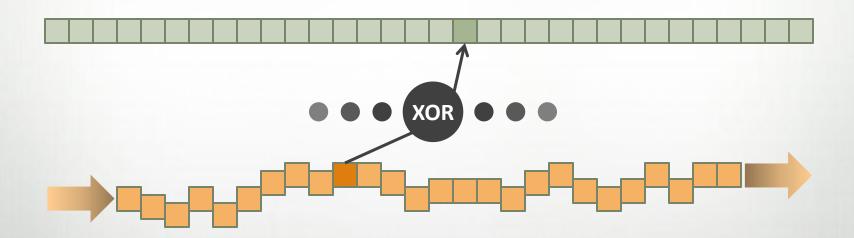




Given: m-element table T (where $m = 2^n$)

Compute: forall r in RandomUpdates do

$$T(r \& (m-1)) ^= r;$$



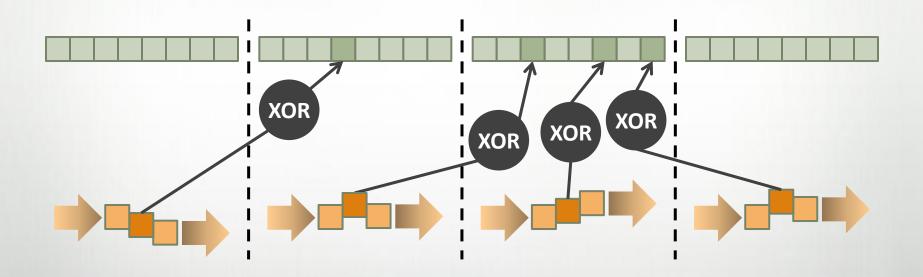




Given: m-element table T (where $m = 2^n$)

Compute: forall r in RandomUpdates do

$$T(r \& (m-1)) ^= r;$$





RA in Chapel: Single Locale



RA in Chapel: Multi-Locale

```
Given: m-element table T (where m = 2^n)
Compute: forall r in RandomUpdates do
              T(r \& (m-1)) ^= r;
config const m = \ldots, N \cup = \ldots, tpl = \ldots;
const TableDist = new Block(1, uint(64), [0..m-1], tpl),
      UpdateDist = new Block(1, uint(64), [0..N U-1], tpl),
      TableSpace: domain(1, uint(64))
                    distributed TableDist = [0..m-1],
      Updates: domain(1, uint(64))
                 distributed UpdateDist = [0..N U-1];
var T: [TableSpace] uint(64);
forall (i,r) in (Updates, RAStream()) do
  on T(r \& (m-1)) do
    T(r \& (m-1)) ^= r;
```



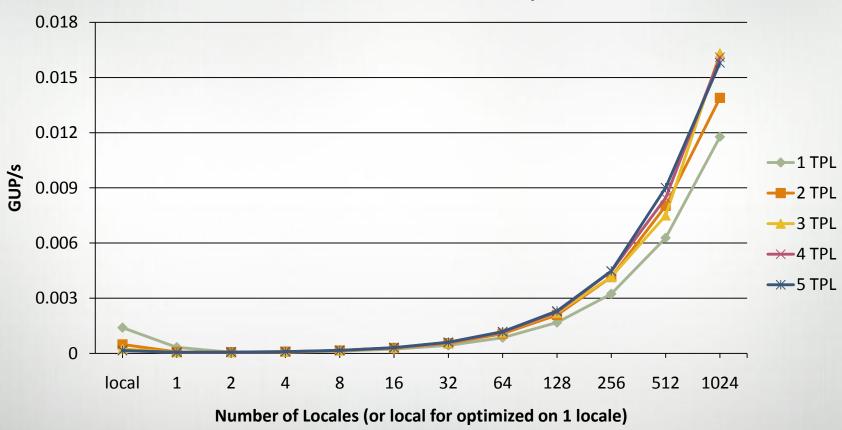
RA in Chapel: Multi-Locale

```
m-element table T (where m = 2^n)
Given:
Compute: forall r in RandomUpdates do
               T(r \& (m-1)) ^= r;
config const m = \ldots, N U = \ldots, tpl = \ldots;
const TableDist = new Block(1, uint(64), [0..m-1], tpl),
      UpdateDist = new Block(1, uint(64), [0..N U-1], tpl),
      TableSpace: domain(1, uint(64))
                    distributed TableDist = [0..m-1],
      Updates: domain(1, uint(64))
                 distributed UpdateDist = [0..N U-1];
var T: [TableSpace] uint(64);
                                        Call ind2loc method directly
forall (i,r) in (Updates, RAStream()) do
  on T.domain.dist.ind2loc(f & (m-1)) do
    T(r \& (m-1)) ^= r;
```



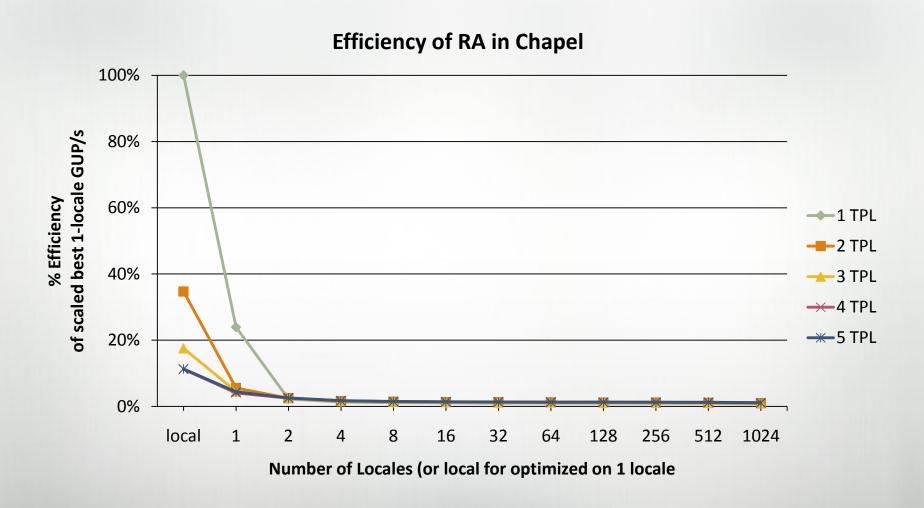






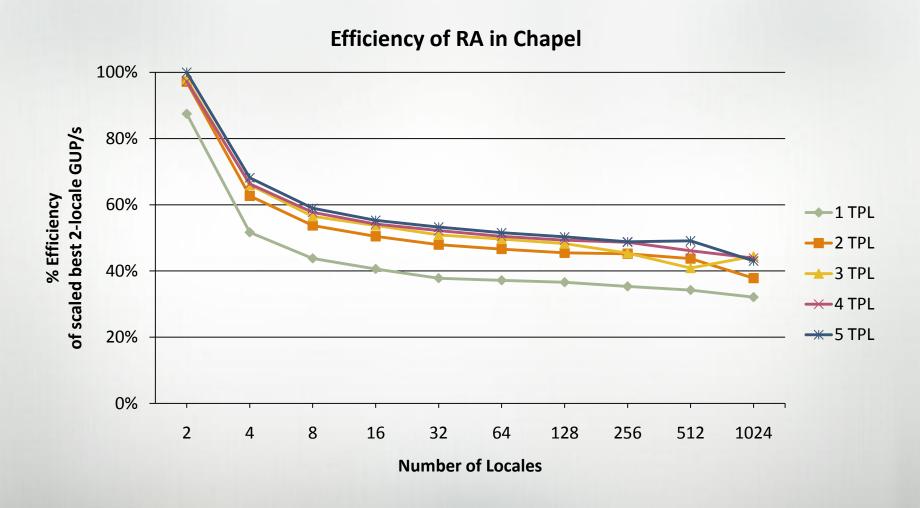










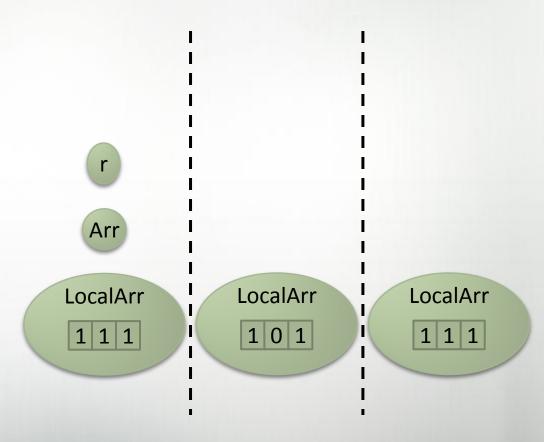




Optimization: Remote Value Forwarding

Simple example

```
var Arr: [Dom] int;
var r: int;
on Locales(1) {
   Arr(r) ^= r;
}
```



Outline



- What is Chapel?
- Chapel's Parallel Programming Model
- HPCC STREAM Triad in Chapel
- HPCC RA in Chapel
- Summary and Future Work

Summary



The global-view programming model is easy to use.

- Shorter, more concise code
- Separation of concerns (partitioning)
- Easy to change data distributions

Distributions implement the global-view model.

- Flexible mechanism for experimentation
- Implementation of distributions is in Chapel

Future Work



- Optimizations
 - Within the compiler
 - Within the runtime
 - Within the distributions
- Complete implementation of Block distribution
- Implement new distributions
 - Cyclic, BlockCyclic, RecursiveBisection
- Experiment with variations of STREAM and RA



Questions?