Chapel: Locality and Affinity

Steve Deitz

SC08: Tutorial S07 – 11/16/08

Outline

- Basics of Multi-Locale Chapel
  - The `locale` type and `Locales` array
  - The `on` statement, `here` locale, and communication
  - The `local` block and `ubiq` variables
- MyDistributedArray Example
- Array and Domain Distributions

**Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.**

The `locale` Type

- Definition
  - An architectural unit of locality
  - Has capacity for processing and storage
- Properties
  - Threads within a locale have ~uniform access to local memory
  - Memory within other locales is accessible, but at a price
- Example
  - A multicore processor or SMP node could be a locale

Chapel Programming Model

- Execution context

```
config const numLocales: int;
const LocaleSpace: domain(1) = [0..numLocales-1];
const Locales: [LocaleSpace] locale;
```

- Explicit Parallelism and Locality Model
  - Execution on remote locales is introduced by **`on`**
  - Parallelism is introduced by **`begin`**/**`cobegin`**/**`coforall`**
  - Note: Distributions may employ the above constructs
- Starting a program
  - Execution begins with one task running on Locale 0
  - The number of locales is specified on the command line

```
> a.out -nl 2
```

**Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.**

Executing on Remote Locales

- Syntax

```
on-stmt:
  on expr { stmt }
```

- Semantics
  - Executes the statement on the locale specified by the expression
  - Does not introduce concurrency
- Example

```
var A: [LocaleSpace] int;
coforall loc in Locales do on loc {
  A(loc.id) = computation(loc.id);
}
```

  - Note: `locale.id` returns a locale's index in the Locales array

Here

- Built-in locale

```
const here: locale;
```

- Semantics
  - Refers to the locale on which the task is executing
- Example

```
writeln(here.id);
on Locales(1) do
  writeln(here.id);
```

- Output

```
0
1
```

Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.

Querying a Locale

- Syntax

```
locale-query-expr:
  expr . locale
```

- Semantics
  - Evaluates the locale on which the expression is located
- Example

```
var i: int;
on Locales(1) {
  write(i.locale.id);
  on i do write(here.id);
}
```

- Output

```
00
```


Remote Reads and Writes

- Example

```
var i = 0;
on Locales(1) {
  writeln((here.id, i.locale.id, i));
  i = 1;
  writeln((here.id, i.locale.id, i));
}
writeln((here.id, i.locale.id, i));
```

- Output

```
(1, 0, 0)
(1, 0, 1)
(0, 0, 1)
```


Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.

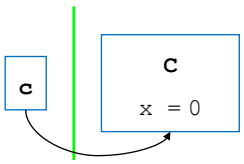
Remote Classes

- Example

```
class C {
  var x: int;
}

var c: C;
on Locales(1) do c = new C();

writeln((here.id, c.locale.id, c));
```



- Output

```
(0, 1, {x = 0})
```


Local Blocks

- Syntax

```
local-stmt:
  local stmt
```

- Semantics
  - Asserts there is no communication in the local statement
  - Runtime checks can be disabled
- Example

```
c = Root.child(1);
on c do local {
  traverseTree(c);
}
```

```
local {
  A(D) = B(D);
}
```


Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.

Ubiquitous Variables

- Syntax

```
ubiquitous-variable-declaration:
  ubiq variable-declaration
```

- Semantics
  - Each locale has its own copy of this variable
  - Can be used to replicate data
- Example

```
ubiq var i: int;
for loc in Locales do on loc {
  i = loc.id;
}
```

- Note: **here** is a ubiquitous constant of locale type

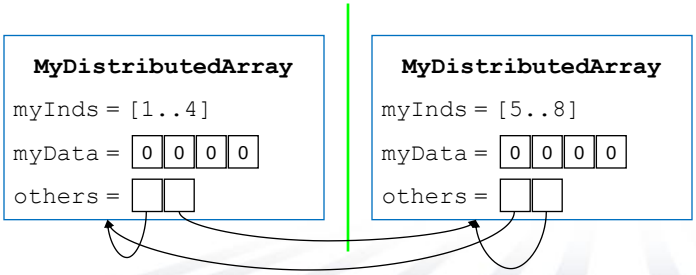
Outline

- Basics of Multi-Locale Chapel
- MyDistributedArray Example
  - Or how to build a class that acts like a simple distributed array
- Array and Domain Distributions

Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.

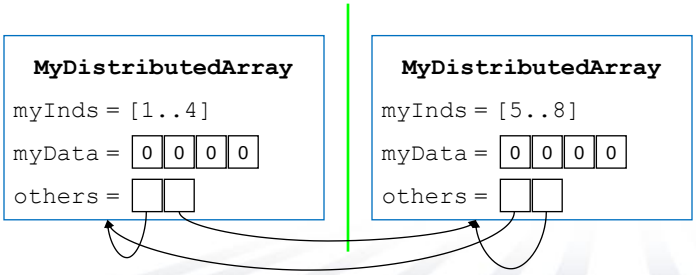
MyDistributedArray Class Declaration

```
class MyDistributedArray {
  var myInds: domain(1);
  var myData: [myInds] int;
  var others: [LocaleSpace] MyDistributedArray;
}
```



MyDistributedArray Access Function

```
def MyDistributedArray.this(i: int) var {
  if myInds.member(i) then
    return myData(i);
  else
    return others((i-1)*numLocales/n).myData(i);
}
```



Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.

MyDistributedArray Construction

```
var A: MyDistributedArray;

var AS: [LocaleSpace] MyDistributedArray;

for loc in Locales do on loc do
  AS(loc.id) =
    new MyDistributedArray([loc.id*n/numLocales+1..
                            (loc.id+1)*n/numLocales]);

A = AS(0);

for loc in Locales do
  AS(loc.id).others = AS;

for i in 1..n do // A is a hotspot
  A(i) = i;
```

Applying Ubiq to MyDistributedArray

```
ubiq var A: MyDistributedArray;

var AS: [LocaleSpace] MyDistributedArray;

for loc in Locales do on loc {
  AS(loc.id) =
    new MyDistributedArray([loc.id*n/numLocales+1..
                            (loc.id+1)*n/numLocales]);
  A = AS(loc.id);
}

for loc in Locales do
  AS(loc.id).others = AS;

for i in 1..n do // No more hotspot
  A(i) = i;
```

Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.

Outline

- Basics of Multi-Locale Chapel
- MyDistributedArray Example
- Array and Domain Distributions
  - The distribution—domain—array hierarchy
  - Using distributions

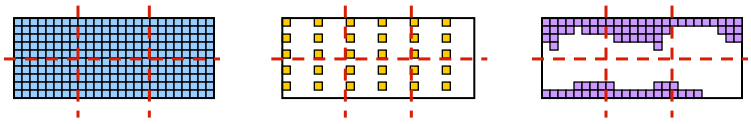
What is a Distribution?

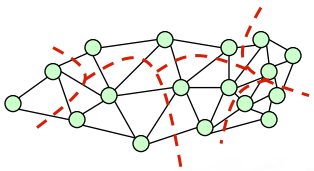
A distribution is a structure that implements…

…the mapping from indices to locales

…the per-locale representation of domain indices and array elements

…the compiler's target interface for lowering global-view operations



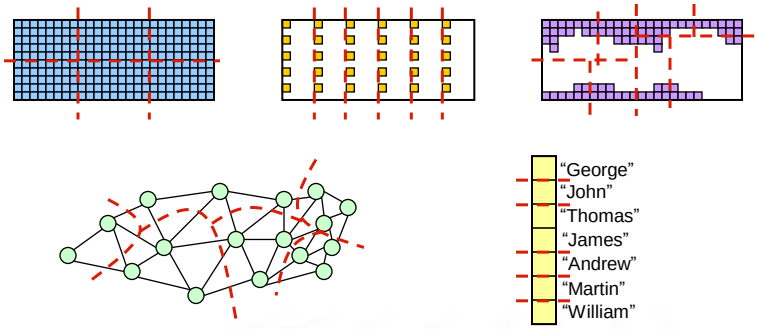


Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.

What is a Distribution?

A distribution is a structure that implements…

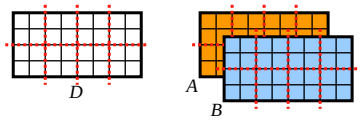
- …the mapping from indices to locales
- …the per-locale representation of domain indices and array elements
- …the compiler's target interface for lowering global-view operations



Distributing Domains

Domains may be distributed across locales

```
var Dist = new Block();
var D: domain(2) dist Dist;
var A,B: [D] real;
```



A distribution implies…

- …ownership of the domain's indices (and its arrays' elements)
- …the default work ownership for operations on the domains/arrays

**Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.**

Distributions (Work in Progress)

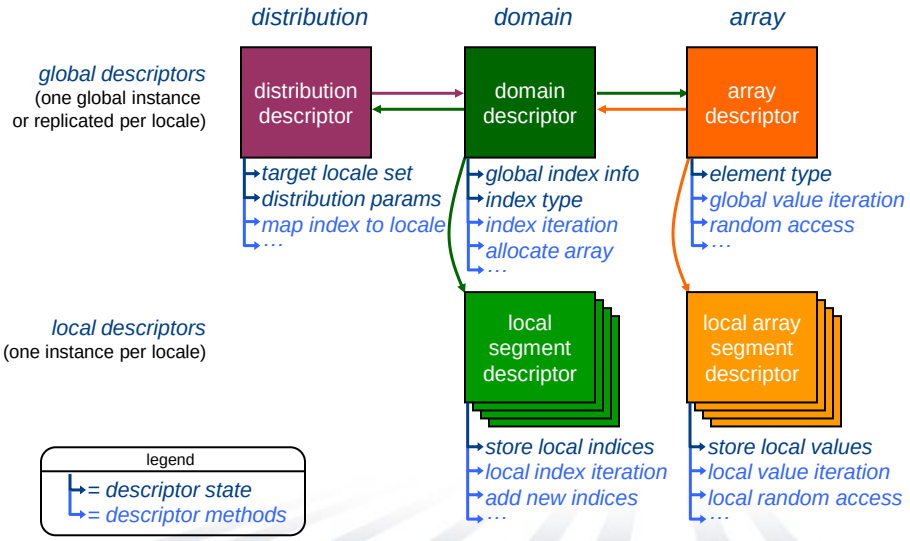
- Distributions support lowering
  - From the user's global view operations on a distributed array
  - To the fragmented implementation for a distributed memory machine
- Users can implement custom distributions
  - Using task parallel features, on clauses, domains/arrays
  - Must implement standard interface
    - For allocation/reallocation of domain indices and array elements
    - For mapping indices to locales and values
    - For iterating in parallel and serial
    - …
  - Optional interface for performance
- Chapel provides a standard library of distributions…
  - Uses the same mechanism as user-defined distributions
  - Tuned for different platforms to maximize performance

Chapel Distributions

	distribution	*domain*	*array*
global descriptors (one global instance or replicated per locale)	**Responsibility:** Mapping of indices to locales	**Responsibility:** How to store, iterate over domain indices	**Responsibility:** How to store, access, iterate over array elements
local descriptors (one instance per locale)		**Responsibility:** How to store, iterate over *local* domain indices	**Responsibility:** How to store, access, iterate over *local* array elements

**Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.**

Chapel Distributions



Questions?

Brad Chamberlain, Steve Deitz,
Samuel Figueroa, David Iten; Cray Inc.