Domain Maps in Chapel

Pieter Hijma

October 27, 2017

1 Introduction

This document describes how domain maps work in Chapel. All code in this document is runnable and we start with the following header that gives us the block distribution:

```
use BlockDist;
```

2 Ranges

Before we start our discussion of domain maps, we explain what a range is, since ranges are needed to create domains. The expression `1..5` is a range starting from 1 to 5 including 5:

```
var yRange : range = 1..5;
var xRange : range = 1..6;
```

3 Domain map types

Each domain has a domain map associated with it. If a domain does not have a domain map specified explicitly, the default domain map is used which typically maps the complete domain to the locale on which the domain value is created or declared in row-major order (similar to C).
A domain map has the type `dmap`. For example, variables of type `dmap(Block(rank=2))` can store domain maps of two-dimensional block distributions:

```
var MyDomainMap: dmap(Block(rank=2)) = new dmap(new Block({yRange,xRange}));
```

4 Domain Types

With this domain map with block distribution of type `dmap`, we can create domain mapped domains by creating a domain type:

```
type MyBlockedDomainType = domain(2) dmapped MyDomainMap;
```

5 Domains

With the type above we can create a domain that has the block distribution:

```
var MyBlockedDomain : MyBlockedDomainType = {yRange, xRange};
```

Note here that the expression `{yRange, xRange}` is of type `domain(2)` (with the default domain map) and we assign the value to a variable of type `domain(2) dmapped MyDomainMap` with our domain map with the block distribution. Assignment typically only transfers the index sets of the domains.

6 Arrays

With this domain we can declare an array:

```
var MyBlockedArray : [MyBlockedDomain] int;
```

7 Syntactic sugar

To make declaring arrays more convenient, Chapel has the `dmapped` operator with which domain maps can be created. Note again that the domain map of a domain is determined by its type and assignment of domains only transfers the index sets and not necessarily the type.
The following statements declare an array similar to `MyBlockedArray`:

```
const MyBlockedDomainSugared : domain(2) dmapped Block({yRange, xRange})
  = {yRange, xRange};
var MyBlockedArraySugared : [MyBlockedDomainSugared] int;
```

You can also use the `dmapped` operator on a domain directly:

```
const MyBlockedDomainDirect = {yRange, xRange} dmapped Block({yRange, xRange});
var MyBlockedArrayDirect : [MyBlockedDomainDirect] int;
```

Note that the type of `MyBlockedDomainDirect` is now `domain(2) dmapped Block({yRange, xRange})` as the constant automatically receives the type of the expression. However, as said before, assignment does not necessarily transfer types but in any case the index sets. This means that in the following statement `MyBlockedDomainStandard` has the standard distribution and not a domain with the block distribution:

```
const MyBlockedDomainStandard : domain(2) =
  {yRange, xRange} dmapped Block({yRange, xRange});
var MyBlockedArrayStandard : [MyBlockedDomainStandard] int;
```

8 Running the code

The following two functions print all the locales of each element of the arrays. The print function is listed below:

```
proc printLocales(a, s) {
  writeln(s, ":");
  forall i in a do i = i.locale.id;
  writeln(a);
  writeln();
}
```

The main function prints each array:

```
proc main() {
  printLocales(MyBlockedArray, "MyBlockedArray");
  printLocales(MyBlockedArraySugared, "MyBlockedArraySugared");
  printLocales(MyBlockedArrayDirect, "MyBlockedArrayDirect");
  printLocales(MyBlockedArrayStandard, "MyBlockedArrayStandard (uses standard distribution)");
}
```