A Comparison of a 1D Stencil Code in C+MPI, Co-Array Fortran, Unified Parallel C, X10, and Chapel

Steve Deitz
Cray Inc.
Iterative Averaging

Repeat until the sum of the changes is less than epsilon.
Example Computation

<table>
<thead>
<tr>
<th></th>
<th>0.0</th>
<th>0.00000</th>
<th>0.00000</th>
<th>0.00000</th>
<th>0.00000</th>
<th>0.00000</th>
<th>6.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
<td>3.00000</td>
<td>6.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.75000</td>
<td>1.50000</td>
<td>3.00000</td>
<td>3.75000</td>
<td>6.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.00000</td>
<td>0.37500</td>
<td>0.75000</td>
<td>2.25000</td>
<td>3.75000</td>
<td>6.0</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>1.00000</td>
<td>2.00000</td>
<td>3.00000</td>
<td>4.00000</td>
<td>5.00000</td>
<td>6.0</td>
<td></td>
</tr>
</tbody>
</table>
Parallel Languages

- C + MPI
  - Serial base language and two-sided message passing library
- Co-Array Fortran (CAF)
  - Modest extension to Fortran 95 (co-dimensions)
- Unified Parallel C (UPC)
  - Small set of extensions to C (shared data)
- X10
  - New HPCS language designed for high productivity
- Chapel
  - New HPCS language designed for high productivity
Array Allocation

### C + MPI

```c
my_n = n * (me+1)/p - n*me/p;
A = (double *)malloc((my_n+2)*sizeof(double));
```

### Co-Array Fortran

```fortran
MY_N = N*ME/P - N*(ME-1)/P
MAX_MY_N = (N+P-1)/P
ALLOCATE(A(0:MAX_MY_N+1)*)
```

### Unified Parallel C

```c
A = (shared double*)upc_all_alloc((n+2), sizeof(double));
```

### Chapel

```chapel
var D : domain(1) dmapped Block([1..n]) = [0..n+1];
var InnerD : subdomain(D) = [1..n];
var A : [D] real;
```
Array Initialization

**C + MPI**

```c
for (i = 0; i <= my_n; i++)
    A[i] = 0.0;
if (me == p - 1)
    A[my_n+1] = n + 1.0;
```

**Co-Array Fortran**

```fortran
A(:) = 0.0
IF (ME .EQ. P) THEN
    A(MY_N+1) = N + 1.0
ENDIF
```

**Unified Parallel C**

```upc
upc forall (i = 0; i <= n; i++; &A[i])
    A[i] = 0.0;
if (MYTHREAD == THREADS-1)
    A[n+1] = n + 1.0;
```

**Chapel**

```chapel
A = 0.0;
A(n+1) = n + 1.0;
```
C + MPI

```c
if (me < p-1)
    MPI_Send(&A[my_n], 1, MPI_DOUBLE, me+1, 1, MPI_COMM_WORLD);
if (me > 0)
    MPI_Recv(&A[0], 1, MPI_DOUBLE, me-1, 1, MPI_COMM_WORLD, &status);
if (me > 0)
    MPI_Send(&A[1], 1, MPI_DOUBLE, me-1, 1, MPI_COMM_WORLD);
if (me < p-1)
    MPI_Recv(&A[my_n+1], 1, MPI_DOUBLE, me+1, 1, MPI_COMM_WORLD, &status);
for (i = 1; i <= my_n; i++)
    Tmp[i] = (A[i-1]+A[i+1])/2.0;
```

Co-Array Fortran

```fortran
CALL SYNC_ALL()
IF (ME .LT. P) THEN
    A(MY_N+1) = A(1)[ME+1]
    A(0)[ME+1] = A(MY_N)
ENDIF
CALL SYNC_ALL()
DO I = 1, MY_N
    TMP(I) = (A(I-1)+A(I+1))/2.0
ENDDO
```

Unified Parallel C

```c
upc_barrier;
upc forall (i = 1; i <= n; i++; &A[i])
    Tmp[i] = (A[i-1]+A[i+1])/2.0;
```

Chapel

```chapel
forall i in InnerD do
    Tmp(i) = (A(i-1)+A(i+1))/2.0;
```
Reduction

C + MPI

```c
my_delta = 0.0;
for (i = 1; i <= my_n; i++)
    my_delta += fabs(A[i]-Tmp[i]);
MPI_Allreduce(&my_delta, &delta, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
```

Co-Array Fortran

```fortran
MY_DELTA = 0.0
DO I = 1,MY_N
    MY_DELTA = MY_DELTA + ABS(A(I)-TMP(I))
ENDDO
CALL SYNC_ALL()
DELTA = 0.0
DO I = 1,P
    DELTA = DELTA + MY_DELTA(I)
ENDDO
```

Unified Parallel C

```c
my_delta = 0.0;
upc_forall (i = 1; i <= n; i++; &A[i])
    my_delta += fabs(A[i]-Tmp[i]);
upc_lock(lock);
delta = delta + my_delta;
upc_unlock(lock);
```

Chapel

```chapel
delta = + reduce abs(A(InnerD)-Tmp(InnerD));
```
Reduction

C + MPI

```c
my_delta = 0.0;
for (i = 1; i <= my_n; i++)
    my_delta += fabs(A[i] - Tmp[i]);
MPI_Allreduce(&my_delta, &delta, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
```

Co-Array Fortran

```fortran
MY_DELTA = 0.0
DO I = 1,MY_N
    MY_DELTA = MY_DELTA + ABS(A(I) - TMP(I))
ENDDO
CALL SYNC_ALL()
DELTA = 0.0
DO I = 1,P
    DELTA = DELTA + MY_DELTA[I]
ENDDO
```

Unified Parallel C

```c
my_delta = 0.0;
upc_forall (i = 1; i <= n; i++; &A[i])
    my_delta += fabs(A[i] - Tmp[i]);
upc_lock(lock);
delta = delta + my_delta;
upc_unlock(lock);
```

Chapel

```chapel
delta = 0.0;
forall i in InnerD do
    delta += abs(A(i) - Tmp(i));
```
C + MPI

```c
if (me == 0)
    printf("Iterations: %d\n", iters);
```

Co-Array Fortran

```fortran
IF (ME .EQ. 1) THEN
    WRITE(*,*) 'Iterations: ', ITERS
ENDIF
```

Unified Parallel C

```c
if (MYTHREAD == 0)
    printf("Iterations: %d\n", iters);
```

Chapel

```chapel
writeln("Iterations: ", iters);
```
### Summary Comparison

#### Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>C+MPI</th>
<th>CAF</th>
<th>UPC</th>
<th>Chapel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global view of computation</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Global view of data</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Data distributions</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Arbitrary data distributions</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Multi-dimensional arrays</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Syntactic execution model</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>One-sided communication</td>
<td>½</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Support for reductions</td>
<td>✔</td>
<td>➙</td>
<td>➙</td>
<td>✔</td>
</tr>
<tr>
<td>Simple mechanisms</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>