Implementing Stencil Problems in Chapel: An Experience Report

Per Fuchs Pieter Hijma Clemens Grelck

Saturday 22 June 2019
CHIUW 2019

VU VRIJE UNIVERSITEIT AMSTERDAM
UNIVERSITY OF AMSTERDAM

Context

- Teach Chapel in course
  - Programming Concurrent Systems (UvA)
  - Parallel Programming Practical (VU)
- Paper based on student report by Per Fuchs
- 4 versions
  - sequential
  - global view single locale
  - global view multi locale
  - local view multi locale

Context

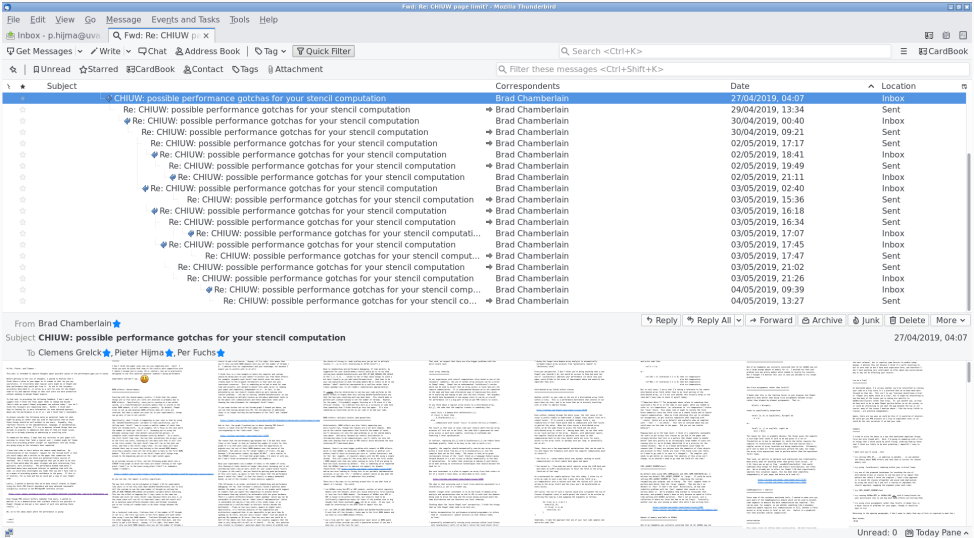
System

- DAS-5
  - Dual socket, 8 cores, 2 hyperthreads
  - 48 Gbps InfiniBand
  - CentOS 7.4, 3.10.0 GNU/Linux
- Software
  - Chapel 1.19 (latest)
  - GCC 6.4.0 (latest version on DAS-5)

Motivation

- Over the years disappointing multi-locale performance
- Our attempt to get to the bottom of this
  - from users' perspective
- CHIUW as target in mind
- Can serve as explanation to (former) students

Conversation Users/Implementers



Textbook implementation

```
1  config const N = 8192;
2  config const M = 8192;
3  config const I = 500;
4  config const E = 0.01;
5
6  const HaloDomain: domain(2) = {0..N+1, 0..M+1};
7  const CylinderDomain: subdomain(HaloDomain) = {1..N, 1..M};
8
9  const LeftHalo: subdomain(HaloDomain) = {1..N, 0..0};
10 const RightHalo: subdomain(HaloDomain) = {1..N, M+1..M+1};
11 const UpperHalo: subdomain(HaloDomain) = {0..0, 0..M+1};
12 const LowerHalo: subdomain(HaloDomain) = {N+1..N+1, 0..M+1};
13
14 const LeftColumn: subdomain(HaloDomain) = {1..N, 1..1};
15 const RightColumn: subdomain(HaloDomain) = {1..N, M..M};
16 const UpperRow: subdomain(HaloDomain) = {1..1, 0..M+1};
17 const LowerRow: subdomain(HaloDomain) = {N..N, 0..M+1};
18
19 class Cylinder {
20   var temperature : [HaloDomain] real;
21 }
```

Textbook implementation

```
1  config const N = 8192;
2  config const M = 8192;
3  config const I = 500;
4  config const E = 0.01;
5
6  const HaloDomain: domain(2) = {0..N+1, 0..M+1};
7  const CylinderDomain: subdomain(HaloDomain) = {1..N, 1..M};
8
9  const LeftHalo: subdomain(HaloDomain) = {1..N, 0..0};
10 const RightHalo: subdomain(HaloDomain) = {1..N, M+1..M+1};
11 const UpperHalo: subdomain(HaloDomain) = {0..0, 0..M+1};
12 const LowerHalo: subdomain(HaloDomain) = {N+1..N+1, 0..M+1};
13
14 const LeftColumn: subdomain(HaloDomain) = {1..N, 1..1};
15 const RightColumn: subdomain(HaloDomain) = {1..N, M..M};
16 const UpperRow: subdomain(HaloDomain) = {1..1, 0..M+1};
17 const LowerRow: subdomain(HaloDomain) = {N..N, 0..M+1};
18
19 class Cylinder {
20   var temperature : [HaloDomain] real;
21 }
```

Textbook implementation

```
1  config const N = 8192;
2  config const M = 8192;
3  config const I = 500;
4  config const E = 0.01;
5
6  const HaloDomain: domain(2) = {0..N+1, 0..M+1};
7  const CylinderDomain: subdomain(HaloDomain) = {1..N, 1..M};
8
9  const LeftHalo: subdomain(HaloDomain) = {1..N, 0..0};
10 const RightHalo: subdomain(HaloDomain) = {1..N, M+1..M+1};
11 const UpperHalo: subdomain(HaloDomain) = {0..0, 0..M+1};
12 const LowerHalo: subdomain(HaloDomain) = {N+1..N+1, 0..M+1};
13
14 const LeftColumn: subdomain(HaloDomain) = {1..N, 1..1};
15 const RightColumn: subdomain(HaloDomain) = {1..N, M..M};
16 const UpperRow: subdomain(HaloDomain) = {1..1, 0..M+1};
17 const LowerRow: subdomain(HaloDomain) = {N..N, 0..M+1};
18
19 class Cylinder {
20   var temperature : [HaloDomain] real;
21 }
```

Textbook implementation

```
for iteration in 1..I {

  forall (i,j) in zip(LeftHalo, RightColumn) {
    src.temperature[i] = src.temperature[j];
  }
  forall (i,j) in zip(RightHalo, LeftColumn) {
    src.temperature[i] = src.temperature[j];
  }

  forall (i, j) in CylinderDomain {
    var weight = conductivity[i, j];
    var remaining_weight = 1 - weight;

    dst.temperature[i, j] =
      weight * src.temperature[i, j] +
      // four direct neighbors
      remaining_weight * factor_direct_neighbors *
      (src.temperature[i-1, j] +
       src.temperature[i, j+1] +
       src.temperature[i+1, j] +
       src.temperature[i, j-1]) +
      // four diagonal neighbors
      remaining_weight * factor_diagonal_neighbors *
      (src.temperature[i-1, j-1] +
       src.temperature[i-1, j+1] +
       src.temperature[i+1, j+1] +
       src.temperature[i+1, j-1]);
  }

  max_difference = max reduce [ij in CylinderDomain]
    abs(dst.temperature[ij] - src.temperature[ij]);

  if max_difference < E break;

  src <=> dst;
}
```

Textbook implementation

```
for iteration in 1..I {

  forall (i,j) in zip(LeftHalo, RightColumn) {
    src.temperature[i] = src.temperature[j];
  }
  forall (i,j) in zip(RightHalo, LeftColumn) {
    src.temperature[i] = src.temperature[j];
  }

  forall (i, j) in CylinderDomain {
    var weight = conductivity[i, j];
    var remaining_weight = 1 - weight;

    dst.temperature[i, j] =
      weight * src.temperature[i, j] +
      // four direct neighbors
      remaining_weight * factor_direct_neighbors *
      (src.temperature[i-1, j] +
       src.temperature[i, j+1] +
       src.temperature[i+1, j] +
       src.temperature[i, j-1]) +
      // four diagonal neighbors
      remaining_weight * factor_diagonal_neighbors *
      (src.temperature[i-1, j-1] +
       src.temperature[i-1, j+1] +
       src.temperature[i+1, j+1] +
       src.temperature[i+1, j-1]);
  }

  max_difference = max reduce [ij in CylinderDomain]
    abs(dst.temperature[ij] - src.temperature[ij]);

  if max_difference < E break;

  src <=> dst;
}
```

Textbook implementation

```
for iteration in 1..I {

  forall (i,j) in zip(LeftHalo, RightColumn) {
    src.temperature[i] = src.temperature[j];
  }
  forall (i,j) in zip(RightHalo, LeftColumn) {
    src.temperature[i] = src.temperature[j];
  }

  forall (i, j) in CylinderDomain {
    var weight = conductivity[i, j];
    var remaining_weight = 1 - weight;

    dst.temperature[i, j] =
      weight * src.temperature[i, j] +
      // four direct neighbors
      remaining_weight * factor_direct_neighbors *
      (src.temperature[i-1, j] +
       src.temperature[i, j+1] +
       src.temperature[i+1, j] +
       src.temperature[i, j-1]) +
      // four diagonal neighbors
      remaining_weight * factor_diagonal_neighbors *
      (src.temperature[i-1, j-1] +
       src.temperature[i-1, j+1] +
       src.temperature[i+1, j+1] +
       src.temperature[i+1, j-1]);
  }

  max_difference = max reduce [ij in CylinderDomain]
    abs(dst.temperature[ij] - src.temperature[ij]);

  if max_difference < E break;

  src <=> dst;
}
```

Vectorization by the C compiler

- vectorization has a large impact

Feedback from Chapel

- LLVM backend will give more control over vectorization

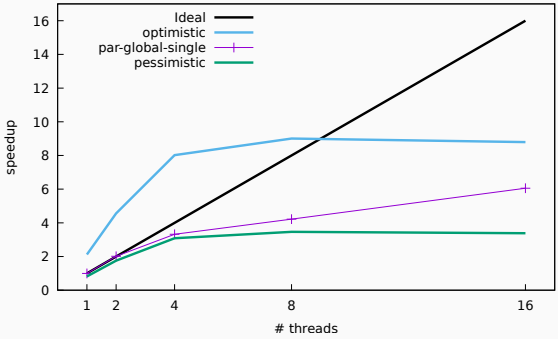
Global View Single Locale

```
for iteration in 1..I {

  forall (i,j) in zip(LeftHalo, RightColumn) {
    src.temperature[i] = src.temperature[j];
  }
  forall (i,j) in zip(RightHalo, LeftColumn) {
    src.temperature[i] = src.temperature[j];
  }

  forall (i, j) in CylinderDomain {
    var weight = conductivity[i, j];
    var remaining_weight = 1 - weight;

    dst.temperature[i, j] =
      weight * src.temperature[i, j] +
      // four direct neighbors
      remaining_weight * factor_direct_neighbors *
      (src.temperature[i-1, j] +
       src.temperature[i, j+1] +
       src.temperature[i+1, j] +
       src.temperature[i, j-1]) +
      // four diagonal neighbors
      remaining_weight * factor_diagonal_neighbors *
      (src.temperature[i-1, j-1] +
       src.temperature[i-1, j+1] +
       src.temperature[i+1, j+1] +
       src.temperature[i+1, j-1]);
  }

  max_difference = max reduce [ij in CylinderDomain]
    abs(dst.temperature[ij] - src.temperature[ij]);

  if max_difference < E break;

  src <=> dst;
}
```

Global View Single Locale

```
1   for iteration in 1..I {
2
3     forall (i,j) in zip(LeftHalo, RightColumn) {
4       src.temperature[i] = src.temperature[j];
5     }
6     forall (i,j) in zip(RightHalo, LeftColumn) {
7       src.temperature[i] = src.temperature[j];
8     }
9
10    forall (i, j) in CylinderDomain {
11      var weight = conductivity[i, j];
12      var remaining_weight = 1 - weight;
13
14      dst.temperature[i, j] =
15        weight * src.temperature[i, j] +
16        // four direct neighbors
17        remaining_weight * factor_direct_neighbors *
18        (src.temperature[i-1, j] +
19         src.temperature[i, j+1] +
20         src.temperature[i+1, j] +
21         src.temperature[i, j-1]) +
22        // four diagonal neighbors
23        remaining_weight * factor_diagonal_neighbors *
24        (src.temperature[i-1, j-1] +
25         src.temperature[i-1, j+1] +
26         src.temperature[i+1, j+1] +
27         src.temperature[i+1, j-1]);
28    }
29
30    max_difference = max reduce [ij in CylinderDomain]
31      abs(dst.temperature[ij] - src.temperature[ij]);
32
33    if max_difference < E break;
34
35    src <=> dst;
36  }
```

Global View Single Locale



Threads	1	2	4	8	16
time (s)	152.20	75.23	45.58	35.85	24.99
Speedup	0.99	2.01	3.32	4.22	6.06
BW STREAM (GB/s)	18.8	40.4	71.1	79.9	78.0
Required BW (GB/s)	22.9	46.4	76.6	97.3	139.6

Global View Multi Locale

```
1  use StencilDist;
2
3  var MyLocaleView = {0..numLocales-1, 0..0};
4  var MyLocales =
5    reshape(Locales[0..numLocales-1], MyLocaleView);
6
7  const HaloDomain: domain(2) dmapped Stencil(
8    boundingBox = {0..N+1, 0..M+1},
9    targetLocales = MyLocales,
10   fluff=(1,1)) = {0..N+1, 0..M+1};
```

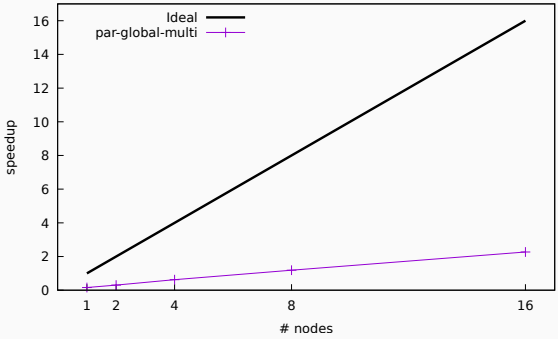
Global View Multi Locale

```
1  for iteration in 1..I {
2
3    ref s = src.temperature;
4    ref d = dst.temperature;
5
6    // exchange columns
7
8    forall (i, j) in CylinderDomain {
9      local {
10       var weight = conductivity[i, j];
11       var remaining_weight = 1 - weight;
12
13       d[i, j] =
14         weight * s[i, j] +
15         // neighbor computations
16       }
17     }
18     d.updateFluff();
19
20     max_difference = max reduce [ij in CylinderDomain]
21       abs(d[ij] - s[ij]);
22
23   if max_difference < E break;
24
25   src <=> dst;
26 }
```

Global View Multi Locale

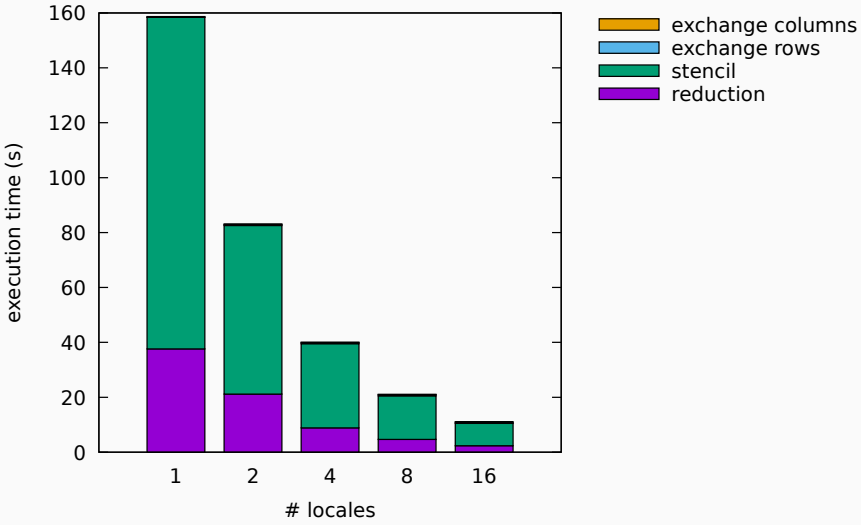
```
1  for iteration in 1..I {
2
3    ref s = src.temperature;
4    ref d = dst.temperature;
5
6    // exchange columns
7
8    forall (i, j) in CylinderDomain {
9      local {
10       var weight = conductivity[i, j];
11       var remaining_weight = 1 - weight;
12
13       d[i, j] =
14         weight * s[i, j] +
15         // neighbor computations
16     }
17   }
18   d.updateFluff();
19
20   max_difference = max reduce [ij in CylinderDomain]
21       abs(d[ij] - s[ij]);
22
23   if max_difference < E break;
24
25   src <=> dst;
26 }
```

Global View Multi Locale



Nodes	1	2	4	8	16
time (s)	158.68	83.13	40.08	21.10	11.03
Speedup	0.16	0.30	0.62	1.18	2.27
Efficiency (%)	0.16	0.15	0.16	0.15	0.14

Global View Multi Locale



Feedback from Chapel

Locality

- `local` blocks are deprecated
- `localAccess` expressions

NUMA awareness

- InfiniBand uses GASNet as communication layer
- GASNet is not NUMA aware
- Registering memory for the network is required
- Happens only on one NUMA-node
- Other NUMA-node suffers from this

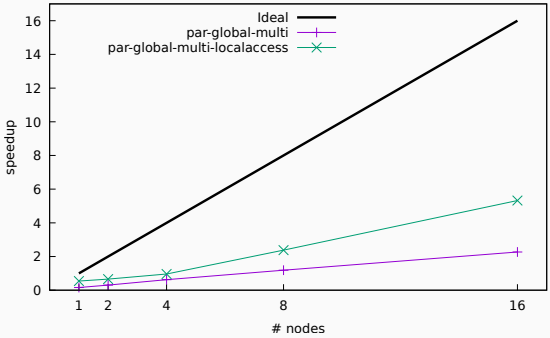
localAccess

```
1  for iteration in 1..I {
2
3    ref s = src.temperature;
4    ref d = dst.temperature;
5
6    // exchange columns
7
8    forall (i, j) in CylinderDomain {
9      local {
10       var weight = conductivity[i, j];
11       var remaining_weight = 1 - weight;
12
13       d[i, j] =
14         weight * s[i, j] +
15         // neighbor computations
16     }
17   }
18   d.updateFluff();
19
20   max_difference = max reduce [ij in CylinderDomain]
21     abs(d[ij] - s[ij]);
22
23   if max_difference < E break;
24
25   src <=> dst;
26 }
```

localAccess

```
1  for iteration in 1..I {
2
3    ref s = src.temperature;
4    ref d = dst.temperature;
5
6    // exchange columns
7
8    forall (i, j) in CylinderDomain {
9
10     var weight = conductivity.localAccess[i, j];
11     var remaining_weight = 1 - weight;
12
13     d.localAccess[i, j] =
14       weight * s.localAccess[i, j] +
15       // neighbor computations
16
17   }
18   d.updateFluff();
19
20   max_difference = max reduce [ij in CylinderDomain]
21     abs(d.localAccess[ij] - s.localAccess[ij]);
22
23   if max_difference < E break;
24
25   src <=> dst;
26 }
```

Performance difference



- factor 3.5 faster for 1 locale
- factor 2.4 faster for 16 locales

Local View Multi Locale

```
1 const Row : domain(1) = {0..M+1};
2
3 class Communicator {
4   var firstRow: [Row] real;
5   var lastRow: [Row] real;
6 }
7
8 var communicators: [LocaleSpace] Communicator;
```

Local View Multi Locale

```
1  var globalIterations = 0;
2  var globalMaxDiff: real = min(real);
3
4  coforall l in Locales with
5      (ref globalIterations, ref globalMaxDiff) {
6    on l {
7      const myCommunicator = new unmanaged Communicator();
8      communicators[here.id] = myCommunicator;
9
10     allLocalesBarrier.barrier();
11     const isLast = here.id == LocaleSpace.last;
12     const isFirst = here.id == LocaleSpace.first;
13
14     const beforeCommunicator = if !isFirst
15       then communicators[here.id - 1]
16       else new unmanaged Communicator();
17     const nextCommunicator = if !isLast
18       then communicators[here.id + 1]
19       else new unmanaged Communicator();
```

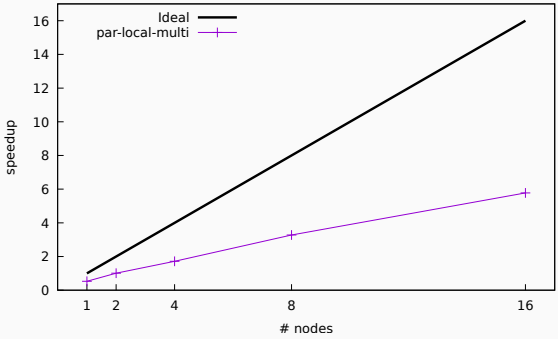
Multi-resolution Design

```
const LocalCylinderDomain = CylinderDomain.localSubdomain();
const LocalHaloDomain = LocalCylinderDomain.expand(1, 1);

const LocalUpperRow = LocalHaloDomain.dim(1).first;
const LocalLowerRow = LocalHaloDomain.dim(1).last;
```

Local View Multi Locale

```
do {
  ref s = src.temperature;
  ref d = dst.temperature;

  local {
    local_max_difference = min(real);

    forall (i, j) in LocalCylinderDomain with
        (max reduce local_max_difference) {
      const weight = localConductivity[i, j];
      const remaining_weight = 1 - weight;
      const oldTemp = s[i, j];

      const newTemp = weight * oldTemp +
        // neighbor computations

      d[i, j] = newTemp;
      local_max_difference = max(local_max_difference,
          abs(new_temp - old_temp));
    }
    // exhange columns
  }

  nextCommunicator.firstRow = d[LocalLowerRow - 1, ..];
  beforeCommunicator.lastRow = d[LocalUpperRow + 1, ..];

  src <=> dst;

  // located on locale 0
  max_diffs[here.id] = local_max_difference;

  allLocalesBarrier.barrier();

  if (!isLast) {
    forall col in Row {
      d[LocalLowerRow, col] = myCommunicator.lastRow[col];
    }
  }
  if (!isFirst) {
    forall col in Row {
      d[LocalUpperRow, col] =
    }
  }

  local_max_difference = max reduce max_diffs;

  local_iteration += 1;
} while (local_max_difference > E && local_iteration < I);
```

Local View Multi Locale



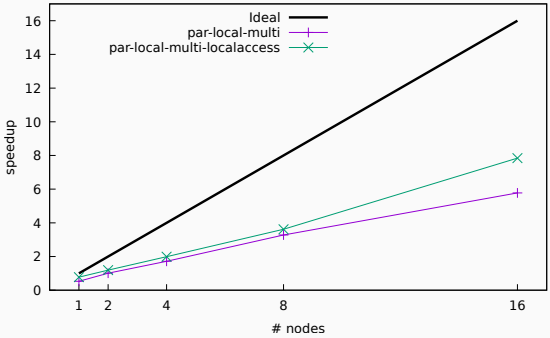
Nodes	1	2	4	8	16
time (s)	47.15	24.74	14.55	7.62	4.33
Speedup	0.53	1.01	1.72	3.28	5.78
Efficiency	0.53	0.51	0.43	0.41	0.36

Feedback Chapel

Same issues

- locality control
- NUMA awareness of the communication layer

Performance difference

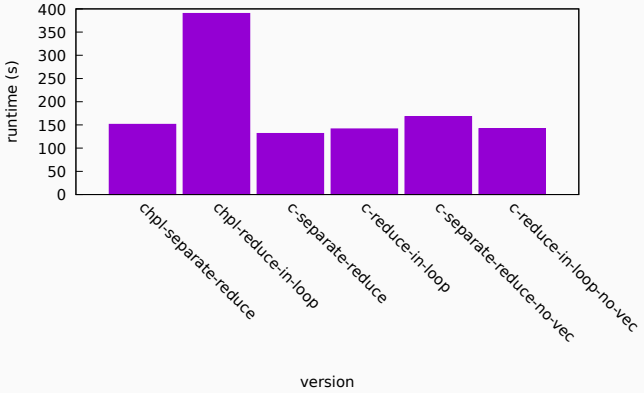


- factor 1.46 faster for 1 locale
- factor 1.36 faster for 16 locales

Conclusions

- We can explain the disappointing performance
- We cannot achieve high performance on our system
  - NUMA awareness of communication layer remains a problem
  - new OpenFabrics layer is promising
- Chapel is a very powerful and expressive language

Performance

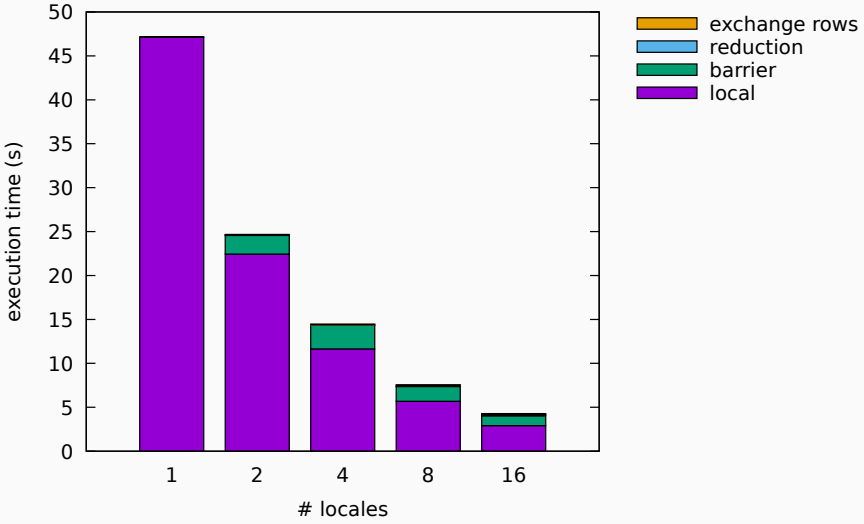


localAccess more fine-grained

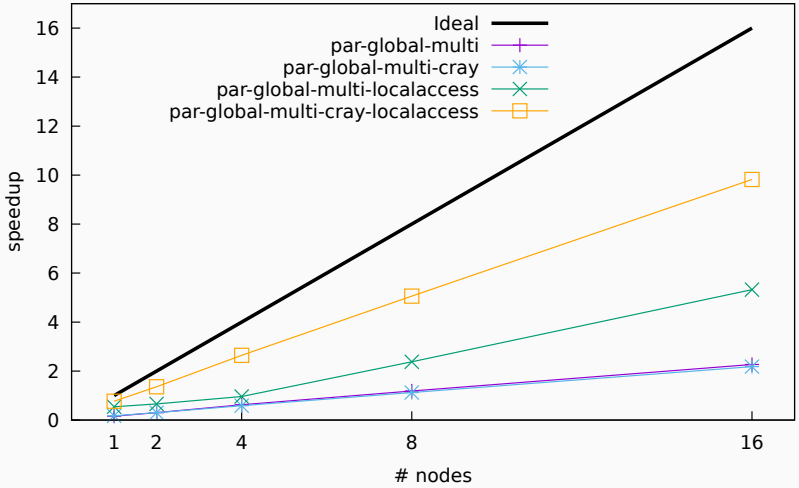
```
for iteration in 1..I {

  ref s = src.temperature;
  ref d = dst.temperature;

  // exchange columns

  forall (i, j) in CylinderDomain {
    local {
      var weight = conductivity[i, j];
      var remaining_weight = 1 - weight;

      d[i, j] =
        weight * s[i, j] +
        // neighbor computations
    }
   }
   d.updateFluff();

  max_difference = max reduce [ij in CylinderDomain]
      abs(d[ij] - s[ij]);

  if max_difference < E break;

  src <=> dst;
}
```

```
var tmp : [CylinderDomain] real;

local {
  tmp = [ij in CylinderDomain] abs(d[ij] - s[ij]);
}

max_difference = max reduce tmp;
```

Local View Multi Locale



All Attempts Multi Locale Global View (speculative)



All Attempts Multi Locale Local View (speculative)