Constructing Parallel Data Distributions

Matthew Knepley, Michael Lange, and Gerard Gorman

Computational and Applied Mathematics
Rice University

Applied Modeling and Computation Group Seminar
Earth Science and Engineering
Imperial College, London October 10, 2016

Mesh and Field Distribution

Suppose I want to distribute a mesh...

Mesh and Field Distribution

- Decide which cells go to which processes (partition)
- Tell processes how many cells are coming
- Send cell descriptions
- Mark *ghost* vertices/edges/faces
- Locally renumber/reorder

Suppose I want to distribute a field...

Mesh and Field Distribution

- Associate field values with parts of mesh
- Tell processes how many field values are coming
- Send field values
- Mark *ghost* values
- Locally renumber/reorder

Suppose I want to communicate ghost values...

Mesh and Field Distribution

- Construct map from owned values to ghost values
- Tell processes how many owned values are coming
- Send field values

Suppose I want to use a cell overlap...

Suppose I want many-to-many instead of one-to-many...

Suppose I want
Jacobian preallocation
instead of ghost values...

Parallel data distributions

can use a single interface

to first class objects.

Main Point

Parallel data distributions

can use a single interface

to first class objects.

Main Point

Parallel data distributions

can use a single interface

to first class objects.

Outline

Communication Automation

Point Space	**Dof Space**	**Section**	**SF**
Solution Dofs	Adjacent Dofs	Jacobian Layout →	Shared Adjacency
Mesh Points	Solution Dofs	Data Layout →	Shared Dofs
Mesh Points	Mesh Points	Topology →	Shared Topology
Processes	Mesh Points	Point Partition →	Shared Points
	Processes		Neighbors

Distribution Automation

We have two main tools:

PetscSection and PetscSF

Distribution Automation

We have two main tools:

PetscSection and PetscSF

PetscSection

A PetscSection is a multimap

$$\mathrm{int} \longrightarrow \{\mathrm{int}\}$$

PetscSection

A PetscSection is a multimap

$$\mathrm{int} \longrightarrow \{\mathrm{int}\}$$

which can represent **mesh topology**

$$\text{mesh point} \longrightarrow \{\text{mesh point}\}$$

PetscSection

A PetscSection is a multimap

$$\text{int} \longrightarrow \{\text{int}\}$$

which can represent **mesh partition**

$$\text{process} \longrightarrow \{\text{mesh point}\}$$

PetscSection

A PetscSection is a multimap

$$\text{int} \longrightarrow \{\text{int}\}$$

which can represent **data layout**

$$\text{mesh point} \longrightarrow \{\text{dof}\}$$

PetscSection

A PetscSection is a multimap

$$\text{int} \longrightarrow \{\text{int}\}$$

which can represent **Jacobian adjacency**

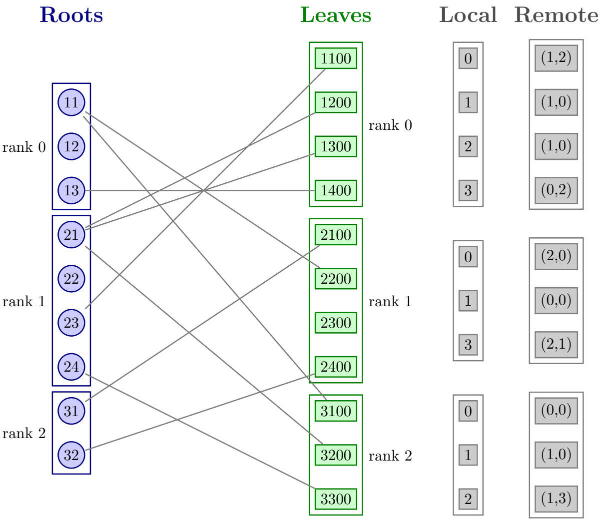
$$\text{dof} \longrightarrow \{\text{dof}\}$$

PetscSF

A PetscSF is a *star-forest*

$$\text{local dof (root)} \longrightarrow \{(\text{local dof},\ \text{process})\ (\text{leaf})\}$$

PetscSF


Roots
Leaves
Local
Remote
rank 0
11
12
13
rank 1
21
22
23
24
rank 2
31
32
1100
1200
1300
1400
rank 0
2100
2200
2300
2400
rank 1
3100
3200
3300
rank 2
0
1
2
3
0
1
3
0
1
2
(1,2)
(1,0)
(1,0)
(0,2)
(2,0)
(0,0)
(2,1)
(0,0)
(1,0)
(1,3)

Outline

Data Description

All data is understood as a function with

a domain consisting of *points*

and

a range consisting of *dofs*

Moving Data

In order to move data, we need

- a migration PetscSF
  - roots are old point distribution
  - leaves are new point distribution
- a PetscSection mapping points to dofs
  - in the old point distribution

Moving Data

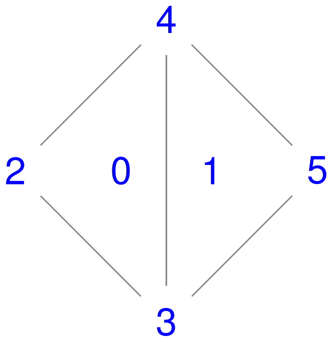
To redistribute the data, we

- redistribute the section
- derive a dof migration SF
  - Pushfoward of point migration SF over Section
- redistribute the values

Outline

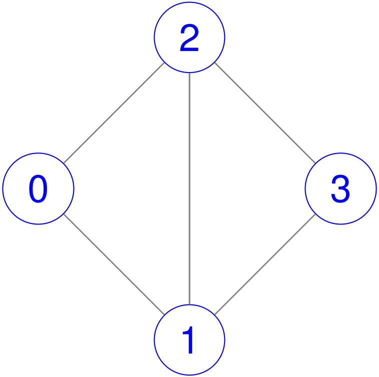
Example
Field Data

Suppose that we have a P_1 discretization and want to distribute the data.



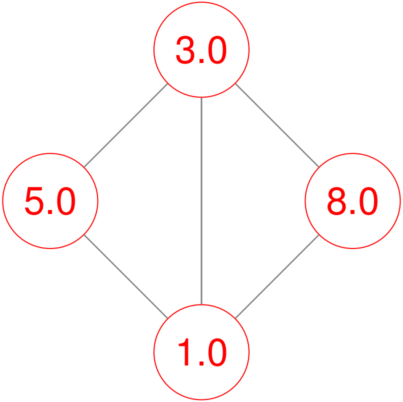
Example
Field Data

Suppose that we have a P_1 discretization
and want to distribute the data.



Example
Field Data

Suppose that we have a P_1 discretization and want to distribute the data.



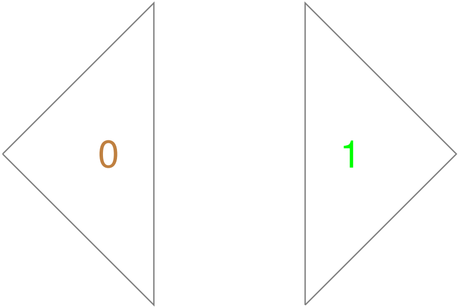
Example
Field Data

Suppose that we have a P_1 discretization
and want to distribute the data.

	Section		Vec
Proc 0	$2 \to \{1, 0\}$	$3 \to \{1, 1\}$	$\{5.0, 1.0, 3.0, 8.0\}$
	$4 \to \{1, 2\}$	$5 \to \{1, 3\}$	
Proc 1			$\{\}$

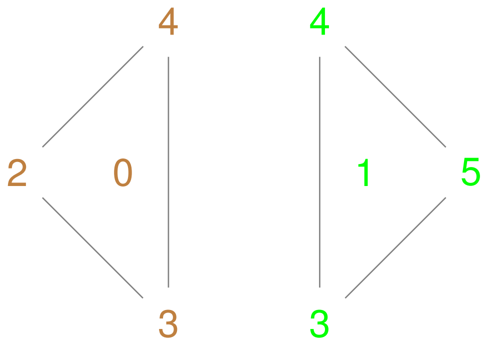
Partition

We create a partition of the cells,



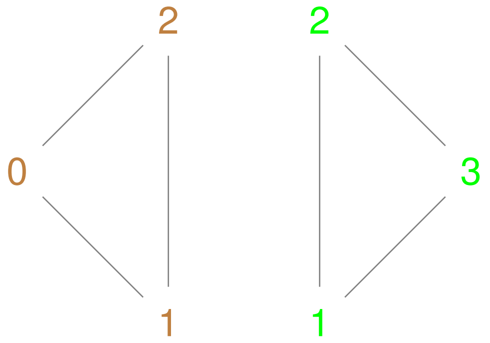
Partition

which induces a vertex partition



Partition

which induces a dof partition



Partition

and can be represented by a Section and IS

	Section		IS
Proc 0	$0 \rightarrow \{3, 0\}$	$1 \rightarrow \{3, 3\}$	$\{0, 1, 2, 1, 2, 3\}$
Proc 1			$\{\}$

Bootstrap

Create a PetscSF that assumes everyone communicates:

Proc 0	Proc 1
$0 \to (p0, 0)$	$0 \to (p0, 1)$
$1 \to (p1, 0)$	$1 \to (p1, 1)$

Bootstrap

Create a PetscSF that assumes a broadcast from Proc 0:

Proc 0	Proc 1
$0 \to (p0, 0)$	$0 \to (p0, 1)$

Partition Inversion

Start with partition info at senders,

	Section		IS
Proc 0	$0 \to \{3, 0\}$	$1 \to \{3, 3\}$	$\{0, 1, 2, 1, 2, 3\}$
Proc 1	$0 \to \{0, 0\}$	$1 \to \{0, 0\}$	$\{\}$

but need partition info at receivers.

	Section		IS
Proc 0	$0 \to \{3, 0\}$	$1 \to \{0, 0\}$	$\{0, 1, 2\}$
Proc 1	$0 \to \{3, 0\}$	$1 \to \{0, 0\}$	$\{1, 2, 3\}$

```
DistributeData(procSF, partSec, MPIU_2INT, part, invpartSec, invpart);
```

Partition Inversion

Start with partition info at senders,

	Section		IS
Proc 0	$0 \rightarrow \{3, 0\}$	$1 \rightarrow \{3, 3\}$	$\{0, 1, 2, 1, 2, 3\}$
Proc 1	$0 \rightarrow \{0, 0\}$	$1 \rightarrow \{0, 0\}$	$\{\}$

but need partition info at receivers.

	Section		IS
Proc 0	$0 \rightarrow \{3, 0\}$	$1 \rightarrow \{0, 0\}$	$\{0, 1, 2\}$
Proc 1	$0 \rightarrow \{3, 0\}$	$1 \rightarrow \{0, 0\}$	$\{1, 2, 3\}$

```
DistributeData(procSF, partSec, MPIU_2INT, part, invpartSec, invpart);
```

Partition Inversion

Start with partition info at senders,

	Section		IS
Proc 0	$0 \rightarrow \{3, 0\}$	$1 \rightarrow \{3, 3\}$	$\{0, 1, 2, 1, 2, 3\}$
Proc 1	$0 \rightarrow \{0, 0\}$	$1 \rightarrow \{0, 0\}$	$\{\}$

but need partition info at receivers.

	Section		IS
Proc 0	$0 \rightarrow \{3, 0\}$	$1 \rightarrow \{0, 0\}$	$\{0, 1, 2\}$
Proc 1	$0 \rightarrow \{3, 0\}$	$1 \rightarrow \{0, 0\}$	$\{1, 2, 3\}$

```
DistributeData(procSF, partSec, MPIU_2INT, part, invpartSec, invpart);
```

Data Distribution

Now the serial data

	Section		Vec
Proc 0	$2 \to \{1, 0\}$	$3 \to \{1, 1\}$	$\{5.0, 1.0, 3.0, 8.0\}$
	$4 \to \{1, 2\}$	$5 \to \{1, 3\}$	
Proc 1			$\{\}$

can be distributed

	Section		Vec
Proc 0	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$\{5.0, 1.0, 3.0\}$
o	$3 \to \{1, 2\}$		
Proc 1	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$\{1.0, 3.0, 8.0\}$
	$3 \to \{1, 2\}$		

Data Distribution

using the same call on a new SF,

```
DistributeData(pointSF, dataSec, MPIU_SCALAR, data, newdataSec, newdata)
```

Notice the local renumbering of vertices.

can be distributed

	Section		Vec
Proc 0	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$\{5.0, 1.0, 3.0\}$
o	$3 \to \{1, 2\}$		
Proc 1	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$\{1.0, 3.0, 8.0\}$
	$3 \to \{1, 2\}$		

Data Distribution

Data distribution has three phases:

1. Distribute the Section
2. Create a new SF for the data
3. Distribute the data with SFBcast()

```
DistributeData(sf, secSource, dtype, dataSource, secTarget, dataTarget) {
  PetscSFDistributeSection(sf, secSource, remoteOff, secTarget);
  PetscSFCreateSectionSF(sf, secSource, remoteOff, secTarget, sfDof);
  PetscSFBcast(sfDof, dtype, dataSource, dataTarget);
}
```

Data Distribution
Phase I

The serial Section

	Serial Section	
Proc 0	$2 \to \{1, 0\}$	$3 \to \{1, 1\}$
	$4 \to \{1, 2\}$	$5 \to \{1, 3\}$
Proc 1		

is pushed over the point SF

Proc 0		Proc 1	
$0 \to (p0, 0)$	$1 \to (p0, 2)$	$0 \to (p0, 1)$	$1 \to (p0, 3)$
$2 \to (p0, 3)$	$3 \to (p0, 4)$	$2 \to (p0, 4)$	$3 \to (p0, 5)$

Data Distribution
Phase I

to give the parallel Section

Serial Section		Parallel Section	
$2 \rightarrow \{1, 0\}$	$3 \rightarrow \{1, 1\}$	$1 \rightarrow \{1, 0\}$	$2 \rightarrow \{1, 1\}$
$4 \rightarrow \{1, 2\}$	$5 \rightarrow \{1, 3\}$	$3 \rightarrow \{1, 2\}$	
		$1 \rightarrow \{1, 0\}$	$2 \rightarrow \{1, 1\}$
		$3 \rightarrow \{1, 2\}$	

is pushed over the point SF

Proc 0		Proc 1	
$0 \rightarrow (p0, 0)$	$1 \rightarrow (p0, 2)$	$0 \rightarrow (p0, 1)$	$1 \rightarrow (p0, 3)$
$2 \rightarrow (p0, 3)$	$3 \rightarrow (p0, 4)$	$2 \rightarrow (p0, 4)$	$3 \rightarrow (p0, 5)$

Data Distribution
Phase II

The point SF

Proc 0		Proc 1	
$0 \to (p0, 0)$	$1 \to (p0, 2)$	$0 \to (p0, 1)$	$1 \to (p0, 3)$
$2 \to (p0, 3)$	$3 \to (p0, 4)$	$2 \to (p0, 4)$	$3 \to (p0, 5)$

is expanded using the Section

	Section		
Proc 0	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$3 \to \{1, 2\}$
Proc 1	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$3 \to \{1, 2\}$

Data Distribution
Phase II

to give a dof SF

Proc 0		Proc 1	
$0 \to (p0, 0)$	$1 \to (p0, 1)$	$0 \to (p0, 1)$	$1 \to (p0, 2)$
$2 \to (p0, 2)$		$2 \to (p0, 3)$	

is expanded using the Section

	Section		
Proc 0	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$3 \to \{1, 2\}$
Proc 1	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$3 \to \{1, 2\}$

Data Distribution
Phase III

Now the serial data

	Serial Vec
Proc 0	$\{5.0, 1.0, 3.0, 8.0\}$
Proc 1	$\{\}$

is sent using the dof SF

Proc 0		Proc 1	
$0 \to (p0, 0)$	$1 \to (p0, 1)$	$0 \to (p0, 1)$	$1 \to (p0, 2)$
$2 \to (p0, 2)$		$2 \to (p0, 3)$	

Data Distribution
Phase III

to the parallel data distribution.

	Serial Vec	Parallel Vec
Proc 0	$\{5.0, 1.0, 3.0, 8.0\}$	$\{5.0, 1.0, 3.0\}$
Proc 1	$\{\}$	$\{1.0, 3.0, 8.0\}$

is sent using the dof SF

Proc 0		Proc 1	
$0 \rightarrow (p0, 0)$	$1 \rightarrow (p0, 1)$	$0 \rightarrow (p0, 1)$	$1 \rightarrow (p0, 2)$
$2 \rightarrow (p0, 2)$		$2 \rightarrow (p0, 3)$	

PetscSection Distribution

Section distribution has three phases:

1. Distribute the number of point dofs
2. Distribute the old dof offsets (used in dof SF)
3. Construct the Section locally

```
DistributeSection(sf, secSource, remoteOff, secTarget) {
  /* Calculate domain (chart) from local SF points */
  PetscSFBcast(sf, secSource.dof, secTarget.dof);
  /* Use remote offsets in PetscSFCreateSectionSF() */
  PetscSFBcast(sf, secSource.off, remoteOff);
  PetscSectionSetUp(secTarget);
}
```

PetscSF Distribution

```
DistributeSF(sfSource, sfMig, sfTarget) {
  PetscSFGetGraph(sfMig, Nr, Nl, leaves, NULL);
  for (p = 0; p < Nl; ++p) { /* Make bid to own all points we received */
    lowners[p].rank  = rank;
    lowners[p].index = leaves ? leaves[p] : p;
  }
  for (p = 0; p < Nr; ++p) { /* Flag so that MAXLOC ignores root value */
    rowners[p].rank  = -3;
    rowners[p].index = -3;
  }
  PetscSFReduce(sfMig, lowners, rowners, MAXLOC);
  PetscSFBcast(sfMig, rowners, lowners);
  for (p = 0, Ng = 0; p < Nl; ++p) {
    if (lowners[p].rank != rank) {
      ghostPoints[Ng] = leaves ? leaves[p] : p;
      remotePoints[Ng].rank  = lowners[p].rank;
      remotePoints[Ng].index = lowners[p].index;
      Ng++;
    }
  }
  PetscSFSetGraph(sfTarget, Np, Ng, ghostPoints, remotePoints);
}
```

Example
Jacobian Connectivity

We can use mesh topology to locally determine the nonzero pattern of the Jacobian,

	Serial Section		IS
Proc 0	$0 \to \{3, 0\}$	$1 \to \{4, 3\}$	{0, 1, 2, 0, 1, 2, 3,
	$2 \to \{4, 7\}$	$3 \to \{3, 11\}$	0, 1, 2, 3, 1, 2, 3}
Proc 1			

and using the same call on the dof SF,

```
DistributeData(dofSF, adjSec, MPIU_INT, adj, newadjSec, newadj);
```

Example
Jacobian Connectivity

it can be distributed.

	Serial Section		IS
Proc 0	$0 \rightarrow \{3, 0\}$	$1 \rightarrow \{4, 3\}$	{0, 1, 2, 0, 1, 2, 3, 0, 1, 2}
	$2 \rightarrow \{4, 7\}$		
Proc 1	$0 \rightarrow \{4, 0\}$	$1 \rightarrow \{4, 4\}$	{0, 1, 2, 3, 0, 1, 2, 3, 1, 2, 3}
	$2 \rightarrow \{3, 7\}$		

and using the same call on the dof SF,

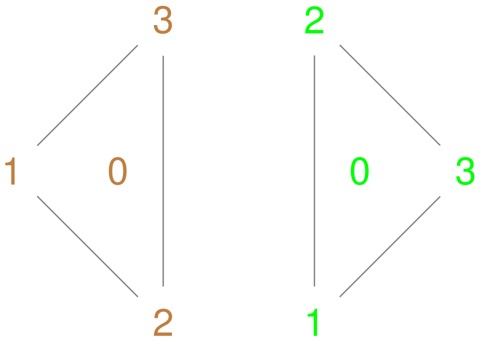
```
DistributeData(dofSF, adjSec, MPIU_INT, adj, newadjSec, newadj);
```

Outline

2 Moving Data
- P_1 Example
- Redistribution Example

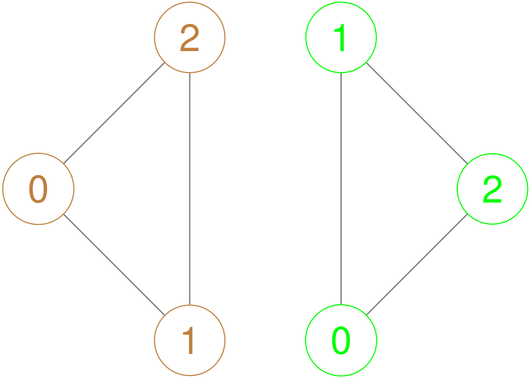
Example
Redistribution

Suppose that we start with distributed P_1 and want to redistribute the data.



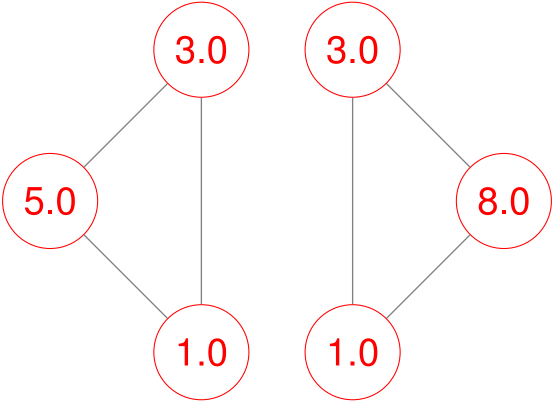
Example
Redistribution

Suppose that we start with distributed P_1 and want to redistribute the data.



Example
Redistribution

Suppose that we start with distributed P_1 and want to redistribute the data.



Example
Redistribution

Suppose that we start with distributed P_1
and want to redistribute the data.

	Local Section		Vec
Proc 0	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$\{5.0, 1.0, 3.0\}$
	$3 \to \{1, 2\}$		
Proc 1	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$\{1.0, 3.0, 8.0\}$
	$3 \to \{1, 2\}$		

Example
Redistribution

Suppose that we start with distributed P_1 and want to redistribute the data.

	Global Section		Vec
Proc 0	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$\{5.0, 1.0, 3.0\}$
	$3 \to \{1, 2\}$		
Proc 1	$1 \to \{1, -2\}$	$2 \to \{1, -3\}$	$\{8.0\}$
	$3 \to \{1, 3\}$		

Example
Redistribution

Suppose that we start with distributed P_1
and want to redistribute the data.

Point SF

Proc 0	Proc 1	
	$1 \to (p0, 2)$	$2 \to (p0, 3)$

Example
Redistribution

Suppose that we start with distributed P_1 and want to redistribute the data.

Dof SF

Proc 0	Proc 1
$0 \to (p0, 1)$	$1 \to (p0, 2)$

Partitioning

Again start with partition info at senders,

	Section		IS
Proc 0	$0 \to \{0, 0\}$	$1 \to \{4, 0\}$	$\{0, 1, 2, 3\}$
Proc 1	$0 \to \{4, 0\}$	$1 \to \{0, 4\}$	$\{0, 1, 2, 3\}$

and get partition info at receivers.

	Section		IS
Proc 0	$0 \to \{0, 0\}$	$1 \to \{4, 0\}$	$\{0, 1, 2, 3\}$
Proc 1	$0 \to \{4, 0\}$	$1 \to \{0, 4\}$	$\{0, 1, 2, 3\}$

using the same call and generic bootstrap SF.

```
DistributeData(procSF, partSec, MPIU_2INT, part, invpartSec, invpart);
```

Partitioning

Again start with partition info at senders,

	Section		IS
Proc 0	$0 \to \{0, 0\}$	$1 \to \{4, 0\}$	$\{0, 1, 2, 3\}$
Proc 1	$0 \to \{4, 0\}$	$1 \to \{0, 4\}$	$\{0, 1, 2, 3\}$

and get partition info at receivers.

	Section		IS
Proc 0	$0 \to \{0, 0\}$	$1 \to \{4, 0\}$	$\{0, 1, 2, 3\}$
Proc 1	$0 \to \{4, 0\}$	$1 \to \{0, 4\}$	$\{0, 1, 2, 3\}$

using the same call and generic bootstrap SF.

```
DistributeData(procSF, partSec, MPIU_2INT, part, invpartSec, invpart);
```

Partitioning

Again start with partition info at senders,

	Section		IS
Proc 0	$0 \to \{0, 0\}$	$1 \to \{4, 0\}$	$\{0, 1, 2, 3\}$
Proc 1	$0 \to \{4, 0\}$	$1 \to \{0, 4\}$	$\{0, 1, 2, 3\}$

and get partition info at receivers.

	Section		IS
Proc 0	$0 \to \{0, 0\}$	$1 \to \{4, 0\}$	$\{0, 1, 2, 3\}$
Proc 1	$0 \to \{4, 0\}$	$1 \to \{0, 4\}$	$\{0, 1, 2, 3\}$

using the same call and generic bootstrap SF.

```
DistributeData(procSF, partSec, MPIU_2INT, part, invpartSec, invpart);
```

Data Distribution
Phase I

The local Section

$$\begin{array}{ll} & \text{Local Section} \\ \text{Proc 0} & 1 \to \{1, 0\} \quad 2 \to \{1, 1\} \\ & 3 \to \{1, 2\} \\ \text{Proc 1} & 1 \to \{1, 0\} \quad 2 \to \{1, 1\} \\ & 3 \to \{1, 2\} \end{array}$$

is pushed over the migration point SF

$$\begin{array}{cc|cc} \multicolumn{2}{c}{\text{Proc 0}} & \multicolumn{2}{c}{\text{Proc 1}} \\ \hline 0 \to (p1, 0) & 1 \to (p1, 1) & 0 \to (p0, 0) & 1 \to (p0, 1) \\ 2 \to (p1, 2) & 3 \to (p1, 3) & 2 \to (p0, 2) & 3 \to (p0, 3) \end{array}$$

Data Distribution
Phase I

to give the new local Section

Old Section		New Section	
$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$
$3 \to \{1, 2\}$		$3 \to \{1, 2\}$	
$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$
$3 \to \{1, 2\}$		$3 \to \{1, 2\}$	

is pushed over the migration point SF

Proc 0		Proc 1	
$0 \to (p1, 0)$	$1 \to (p1, 1)$	$0 \to (p0, 0)$	$1 \to (p0, 1)$
$2 \to (p1, 2)$	$3 \to (p1, 3)$	$2 \to (p0, 2)$	$3 \to (p0, 3)$

Data Distribution
Phase II

The point SF

Proc 0		Proc 1	
$0 \to (p1, 0)$	$1 \to (p1, 1)$	$0 \to (p0, 0)$	$1 \to (p0, 1)$
$2 \to (p1, 2)$	$3 \to (p1, 3)$	$2 \to (p0, 2)$	$3 \to (p0, 3)$

is expanded using the Section

	Local Section		
Proc 0	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$3 \to \{1, 2\}$
Proc 1	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$3 \to \{1, 2\}$

Data Distribution

Phase II

to give a dof SF

Proc 0		Proc 1	
$0 \to (p1, 0)$	$1 \to (p1, 1)$	$0 \to (p0, 0)$	$1 \to (p0, 1)$
$2 \to (p1, 2)$		$2 \to (p0, 2)$	

is expanded using the Section

Local Section

Proc 0	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$3 \to \{1, 2\}$
Proc 1	$1 \to \{1, 0\}$	$2 \to \{1, 1\}$	$3 \to \{1, 2\}$

Data Distribution
Phase III

Now the original data

	Old Vec
Proc 0	$\{5.0, 1.0, 3.0\}$
Proc 1	$\{1.0, 3.0, 8.0\}$

is sent using the dof SF

Proc 0		Proc 1	
$0 \rightarrow (p1, 0)$	$1 \rightarrow (p1, 1)$	$0 \rightarrow (p0, 0)$	$1 \rightarrow (p0, 1)$
$2 \rightarrow (p1, 2)$		$2 \rightarrow (p0, 2)$	

Data Distribution

Phase III

to the new data distribution.

	Old Vec	New Vec
Proc 0	$\{5.0, 1.0, 3.0\}$	$\{1.0, 3.0, 8.0\}$
Proc 1	$\{1.0, 3.0, 8.0\}$	$\{5.0, 1.0, 3.0\}$

is sent using the dof SF

Proc 0		Proc 1	
$0 \to (p1, 0)$	$1 \to (p1, 1)$	$0 \to (p0, 0)$	$1 \to (p0, 1)$
$2 \to (p1, 2)$		$2 \to (p0, 2)$	

Automation

We can automatically generate complex, parallel communication patterns for structured data

Automation

$$\text{Section} + \text{SF} \Longrightarrow \text{SF}$$

Automation

Partition + Bootstrap SF $\Longrightarrow$ Migration Point SF

Automation

Dof Section + Point SF $\Longrightarrow$
Migration Dof SF

Automation

Adjacency + Dof SF $\Longrightarrow$ Migration Adjacency SF

Automation

Topology + Point SF $\Longrightarrow$
Migration Topology SF

Automation

Simple Spec (FE, FV, FD) $\Longrightarrow$ Dof Section

Communication Automation

Point Space	**Dof Space**	**Section**	**SF**
Solution Dofs	Adjacent Dofs	Jacobian Layout →	Shared Adjacency
Mesh Points	Solution Dofs	Data Layout →	Shared Dofs
Mesh Points	Mesh Points	Topology →	Shared Topology
Processes	Mesh Points	Point Partition →	Shared Points
	Processes		Neighbors

Advantages

- Composable Abstractions
- Independent of
  - Dimension
  - Cell Shape
  - Discretization
- Localizes Optimization
- Extensible by User

Enabled Features

- Parallel Mesh Loads
- Parallel Load Balancing
- Arbitrary Mesh Overlap
- Arbitrary datatype support (DMNetwork)

Thank You!

http://www.caam.rice.edu/~mk51