Multiparadigm Parallel Programming with Charm++, Featuring ParFUM as a case study

5th Annual Workshop on Charm++ and its Applications

Aaron Becker
abecker3@uiuc.edu
UIUC
18 April 2007

What is Multiparadigm Programming?

There are lots of ways to write a parallel program

BSP

Global Arrays

X10

OpenMP

High Performance Fortran

Fortress

Parallel Matlab

MPI

Charm++

Multiphase Shared Arrays

Unified Parallel C

Chapel

Why are there so many languages?

Why are there so many languages?

Each is good at something different

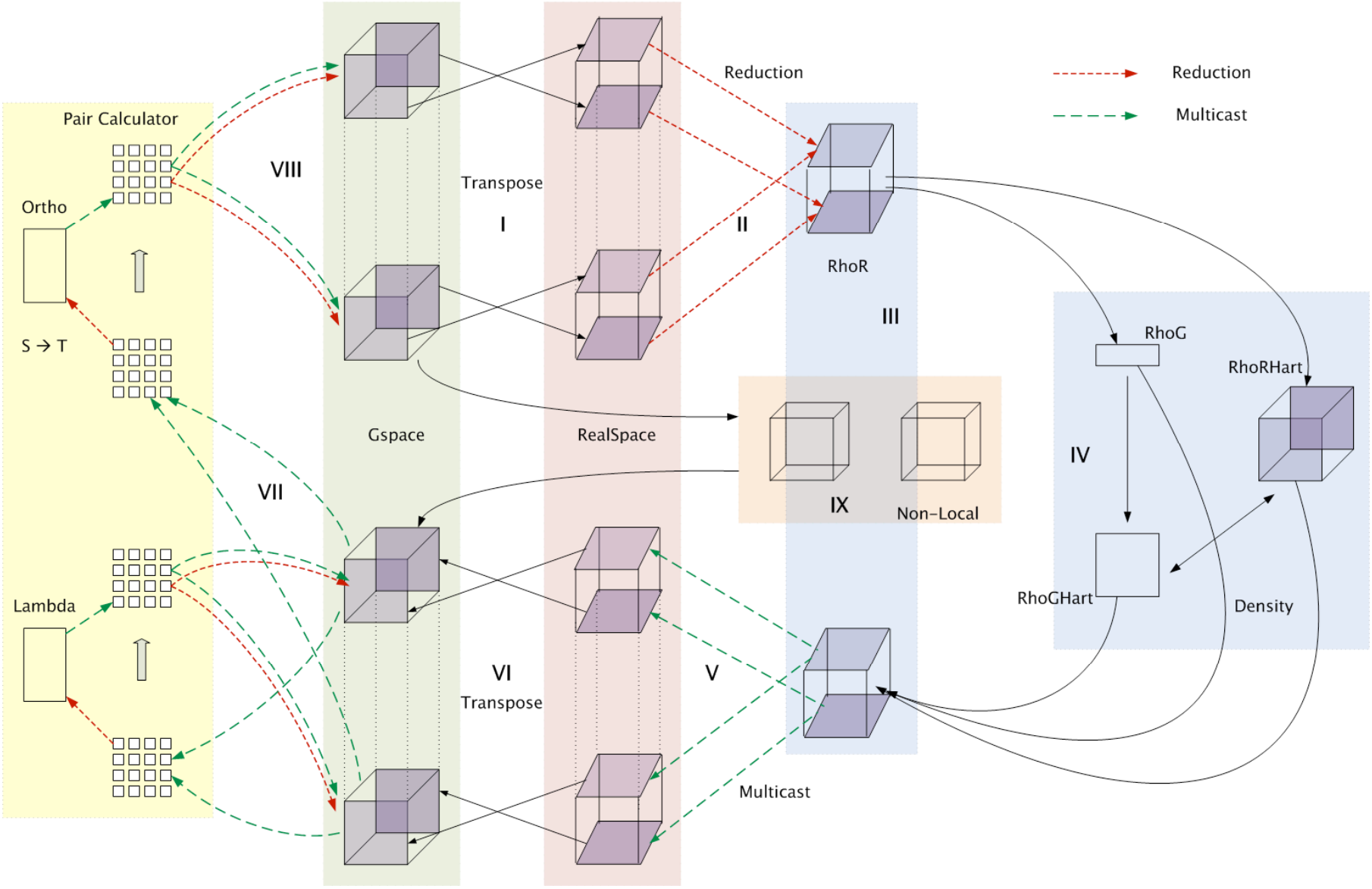
Automatic parallelizing of loops
Fine-grained parallelism
Unpredictable communication patterns

So, what is a multiparadigm program?

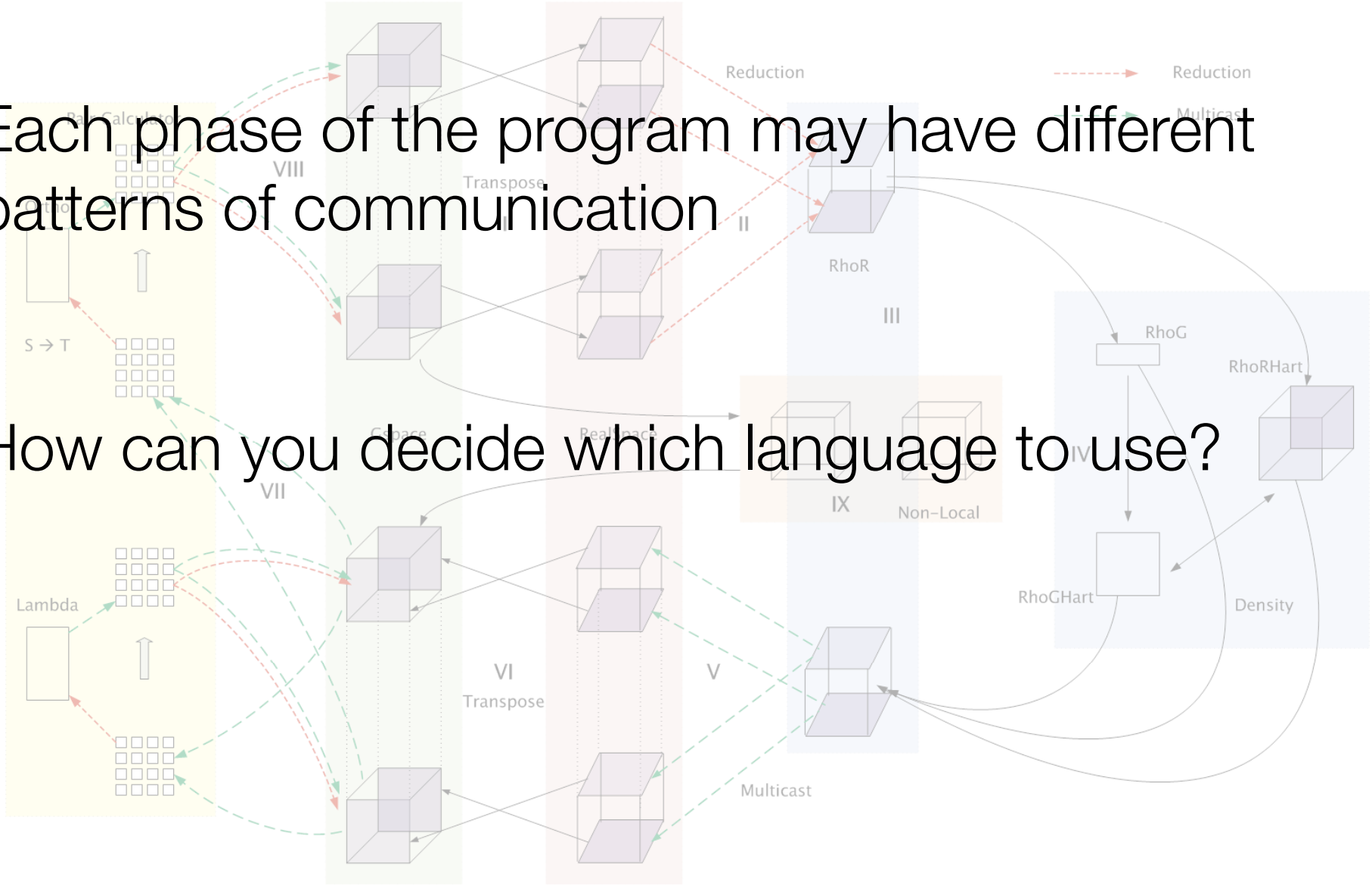
A program composed of modules,
where each module could be written
in a different language

Why would I want a Multiparadigm Program?

Suppose you have a complex program to parallelize



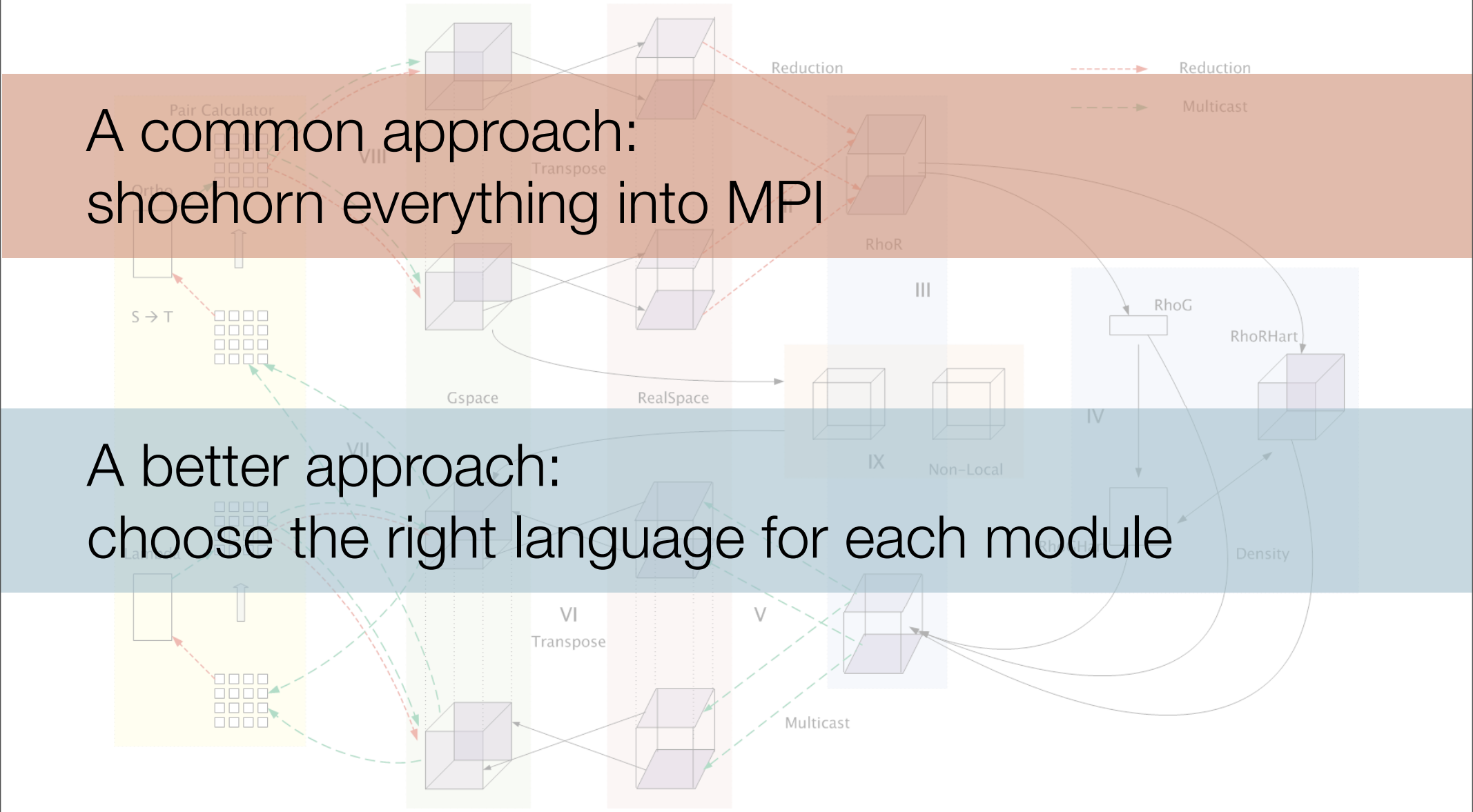
Suppose you have a complex program to parallelize



Each phase of the program may have different patterns of communication

How can you decide which language to use?

Suppose you have a complex program to parallelize



A common approach:
shoehorn everything into MPI

A better approach:
choose the right language for each module

Suppose you have an existing MPI program

You want to add a new module,
but it will be tough to write in MPI

Suppose you have an existing MPI program

You want to add a new module,
but it will be tough to write in MPI

A common approach:
write it in MPI anyway

A better approach:
choose a better suited language

Why aren't multiparadigm programs more common?

Multiparadigm programs are hard to write:
You need a way to stick these modules together

It's relatively simple if you only have one language in use at once: MPI/OpenMP hybrid codes run just one language at a time

For tightly integrated codes with multiple concurrent modules, you need a runtime system to manage them all

Where does Charm++ fit in?

The Charm++ runtime system (RTS) handles most of the difficulties of multiparadigm programming

Modules using different languages are co-scheduled and can integrate tightly with one another

The RTS supports several languages
We are interested in adding more

ParFUM: a Multiparadigm Charm++ Program

What is ParFUM?

ParFUM: a Parallel Framework for Unstructured Meshing

Meant to simplify the development of parallel unstructured mesh codes

Handles partitioning, synchronization, adaptivity, and other difficult parallel tasks

ParFUM is multiparadigm

ParFUM consists of many modules, written in a variety of languages. I will briefly present three examples:

Charm++ for asynchronous adaptivity

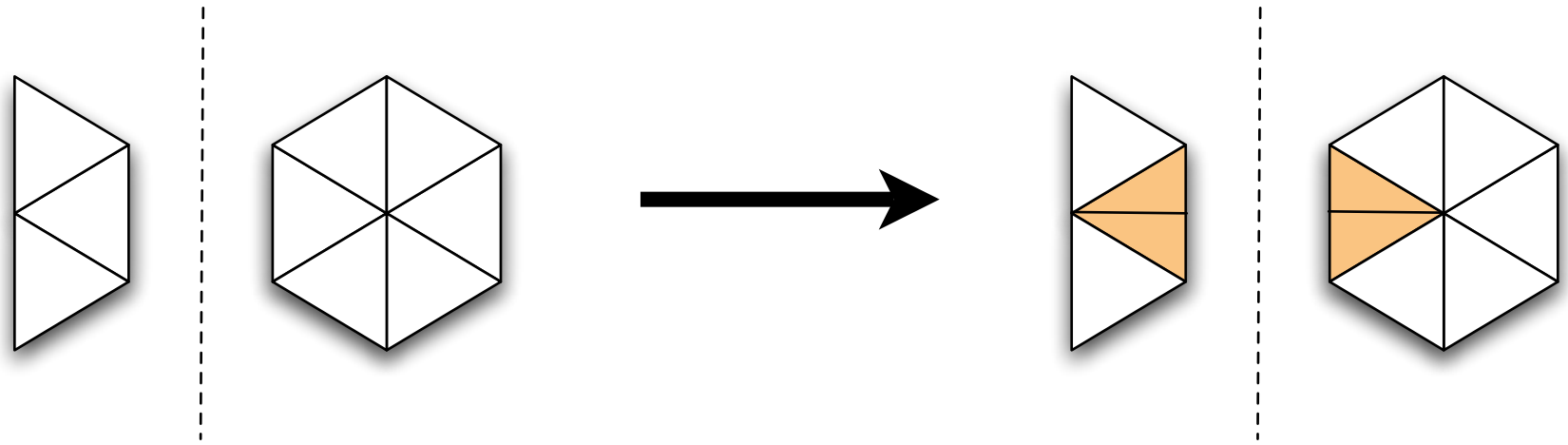
Adaptive MPI for the user's driver code and glue code to connect modules

Multiphase shared arrays (MSA) for data distribution

Charm++ in ParFUM

Asynchronous Incremental Adaptivity

Local refinement or coarsening of the mesh, without any global barriers.



Edge bisection on a processor boundary

What is Charm++?

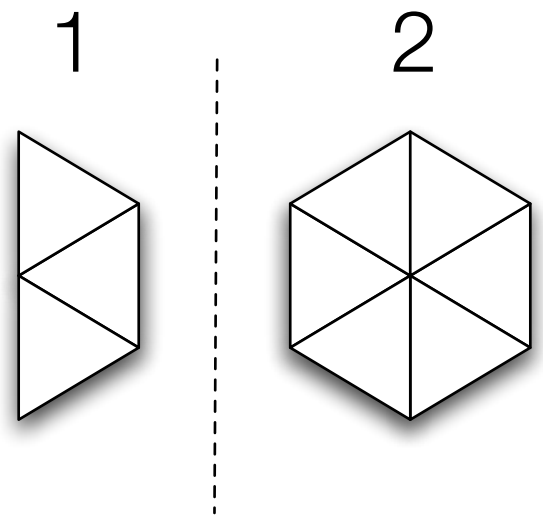
I hope you attended the fine tutorial by Pritish Jetley and Lukasz Wesolowski

In a nutshell, parallel objects which communicate via asynchronous method invocations

Why is Charm++ good for incremental adaptivity?

Incremental adaptivity leads to unpredictable communication patterns.

Suppose a boundary element of partition 1 requests refinement



How will partition 2 know to expect communication from 1? In MPI, this is very hard. In Charm++, it is natural.

Adaptive MPI in ParFUM

What is Adaptive MPI?

For our purposes, it's just an implementation of MPI on top of the Charm++ RTS

For more information, see Celso Medes's tutorial on Friday at 3:10,
How to Write Applications using Adaptive MPI

Why is Adaptive MPI important in ParFUM?

User provided driver code

Glue code between modules

Why is Adaptive MPI important in ParFUM?

User provided driver code

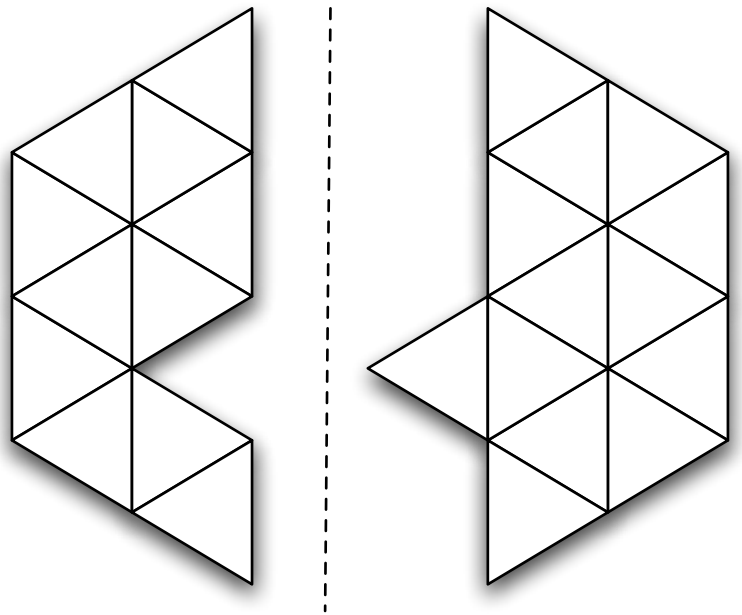
- Popularity
- Legacy

Glue code between modules

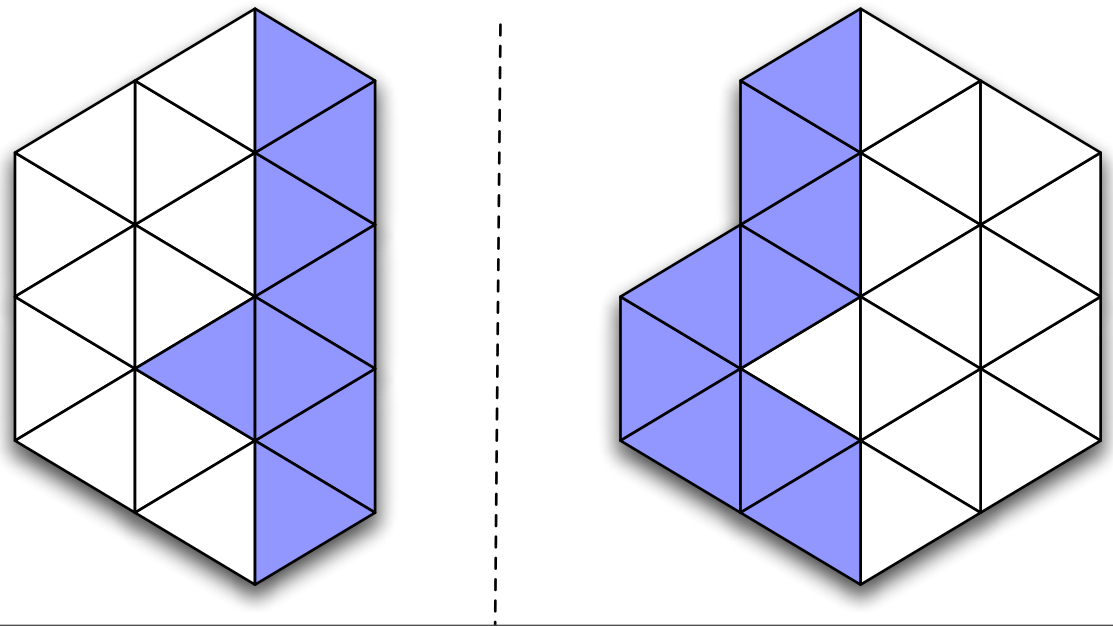
- Simple flow of control

Multiphase Shared Arrays (MSA) in ParFUM

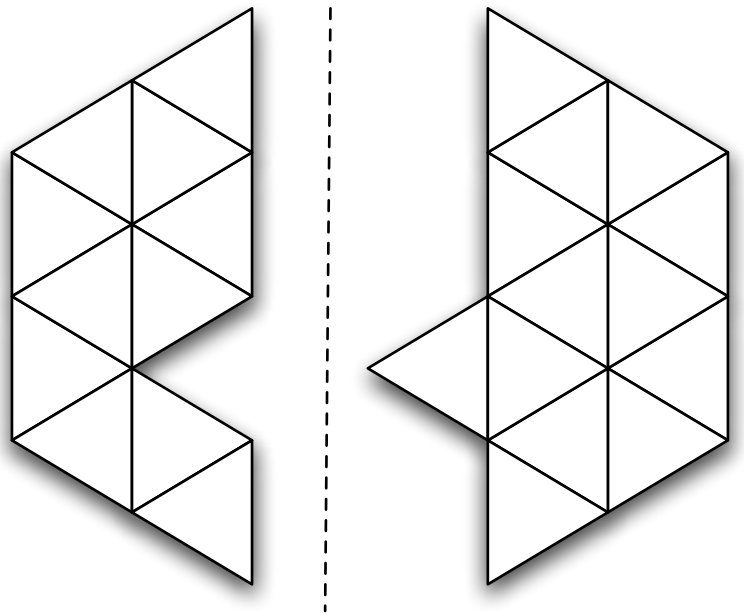
A data distribution problem



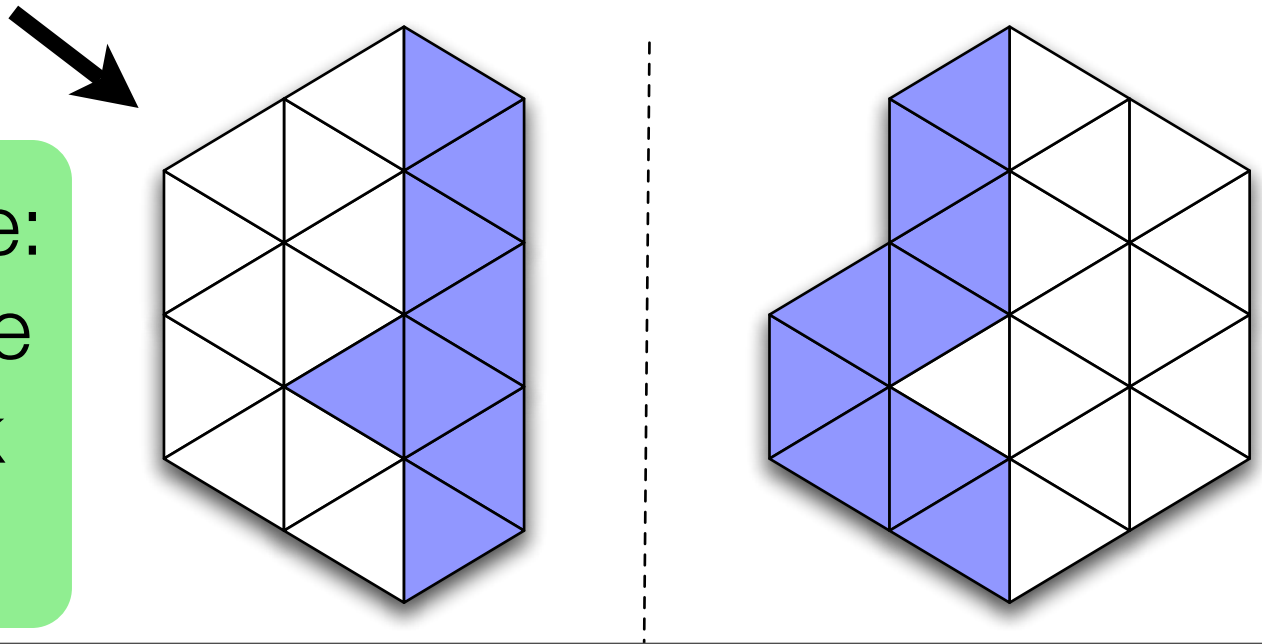
After initial partitioning, we need to determine which boundary elements must be exchanged.



A data distribution problem



After initial partitioning, we need to determine which boundary elements must be exchanged.



What we would like: an easily accessible global table to look up shared edges

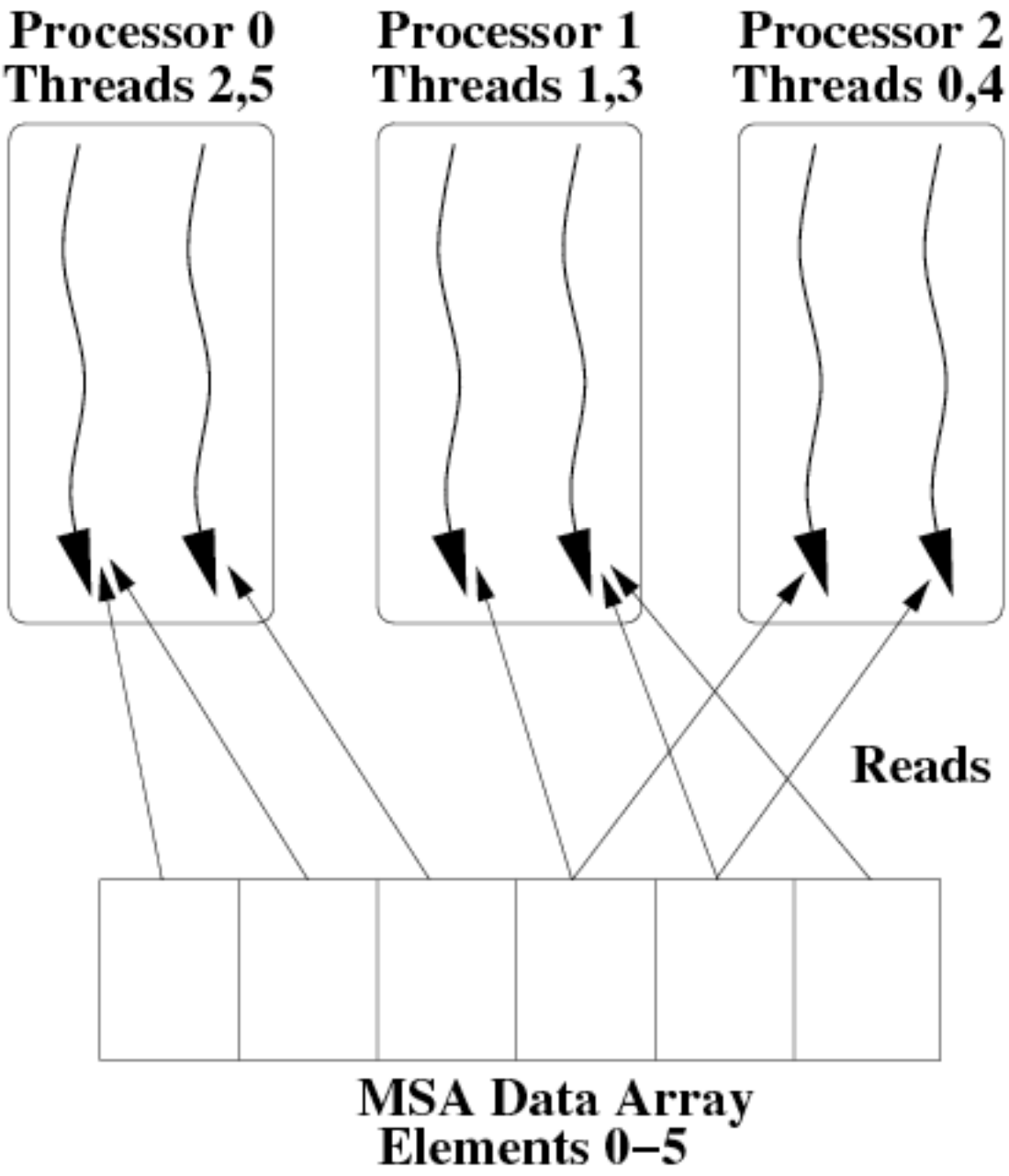
What is MSA?

Idea: shared arrays, where only one type of access is allowed at a time

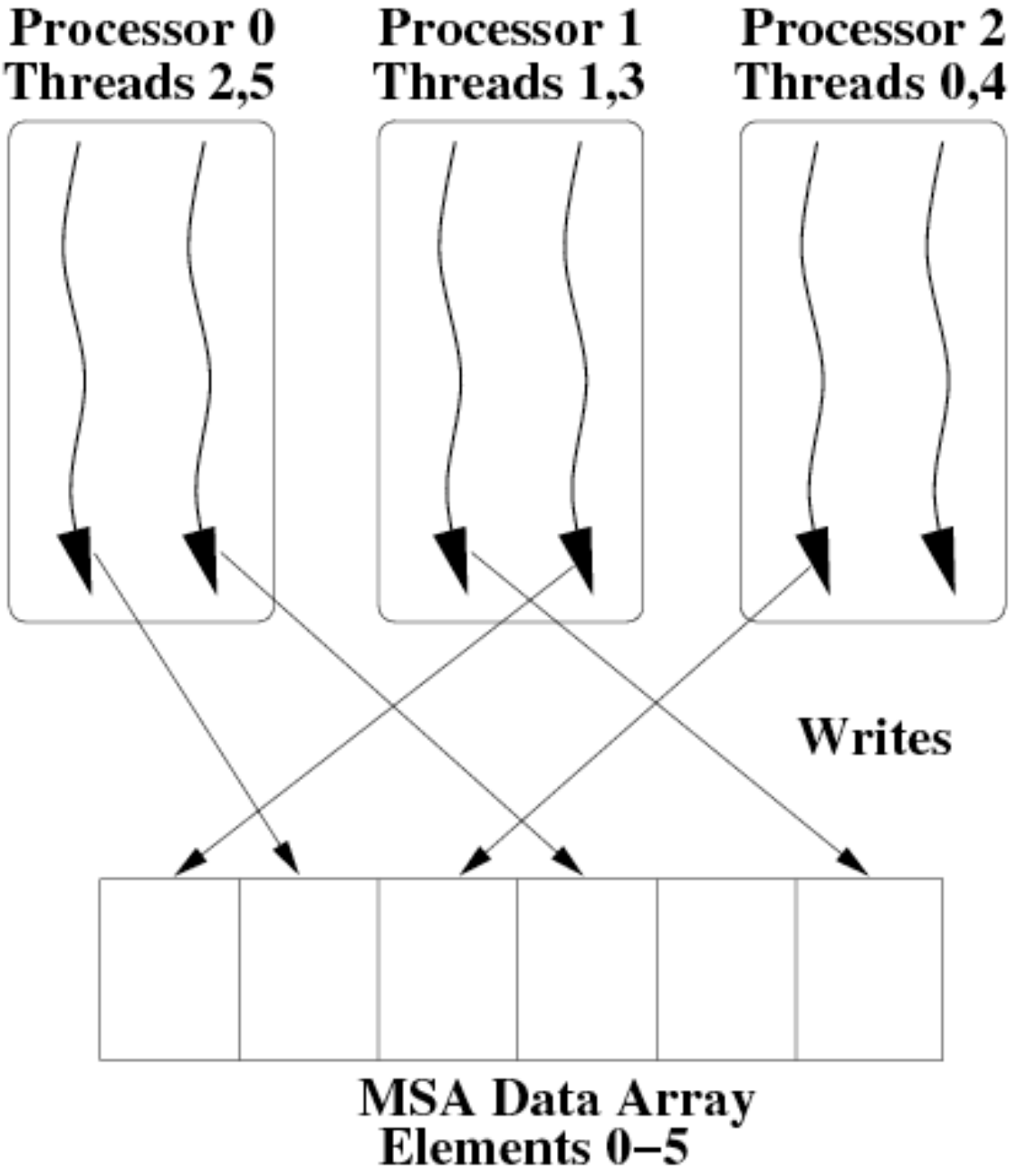
Access type is controlled by the array's *phase*

Phases include:

- read-only
- write-by-one
- accumulate

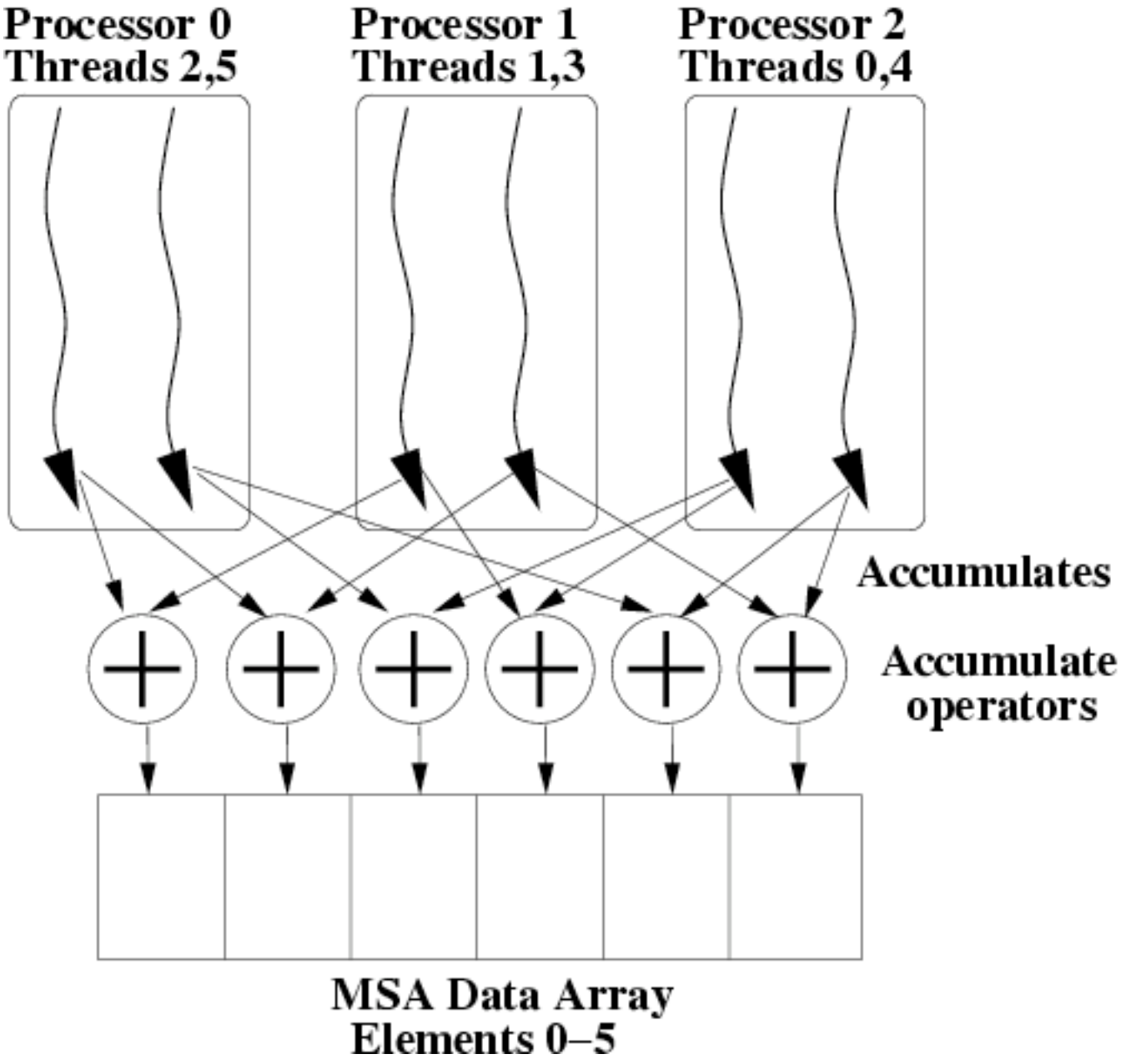


Read-only mode



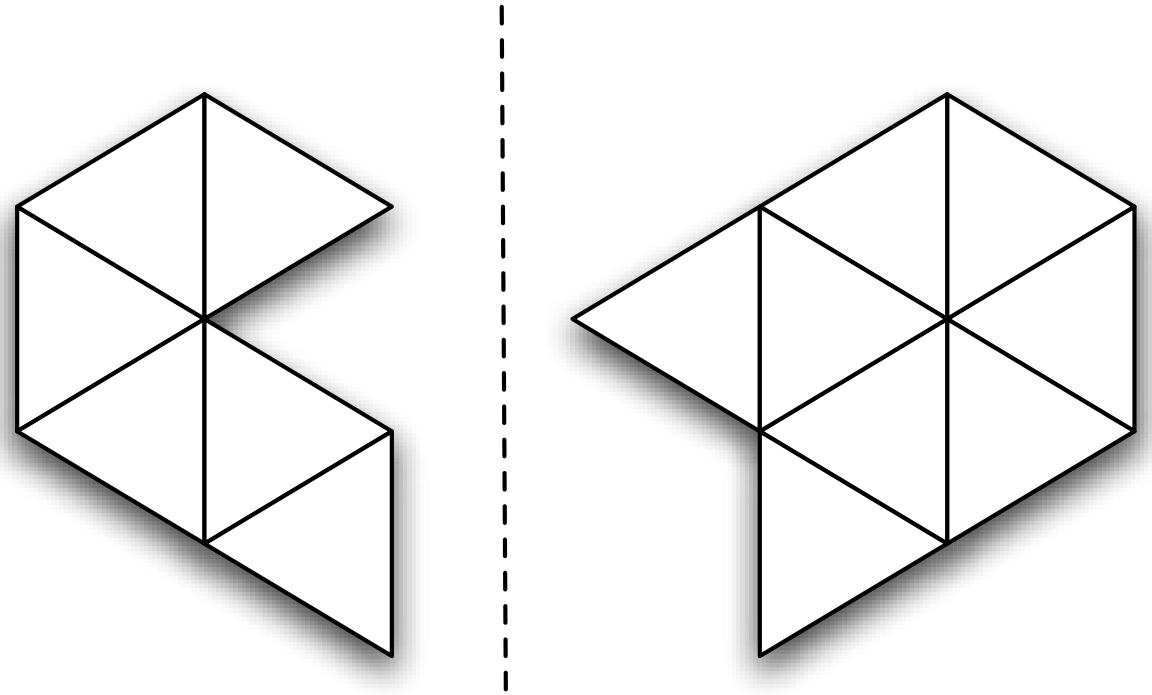
Write-by-one mode

note: one thread could write to many elements

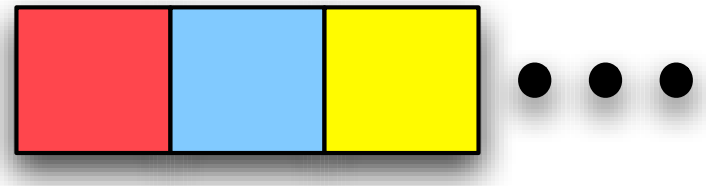


Accumulate mode

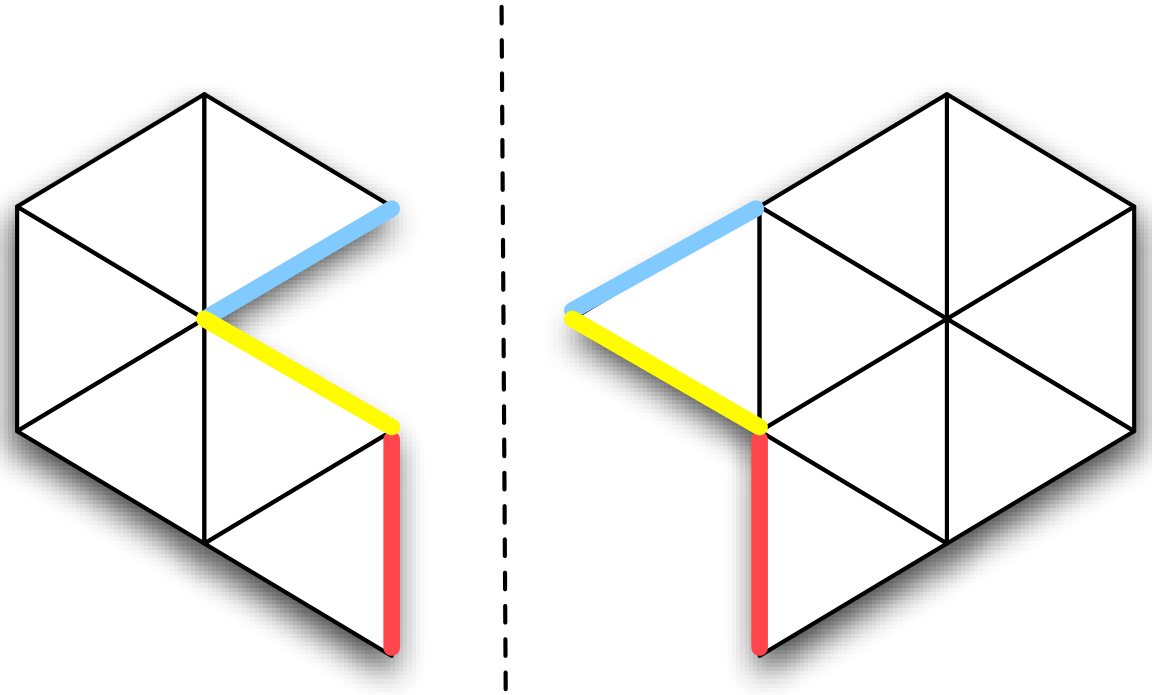
note: accumulation operator must be associative and commutative

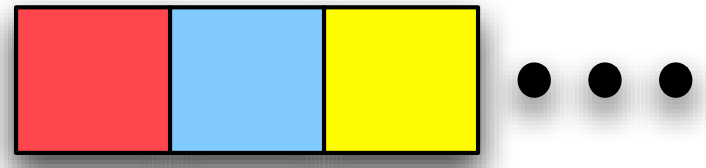


Partitioned Mesh

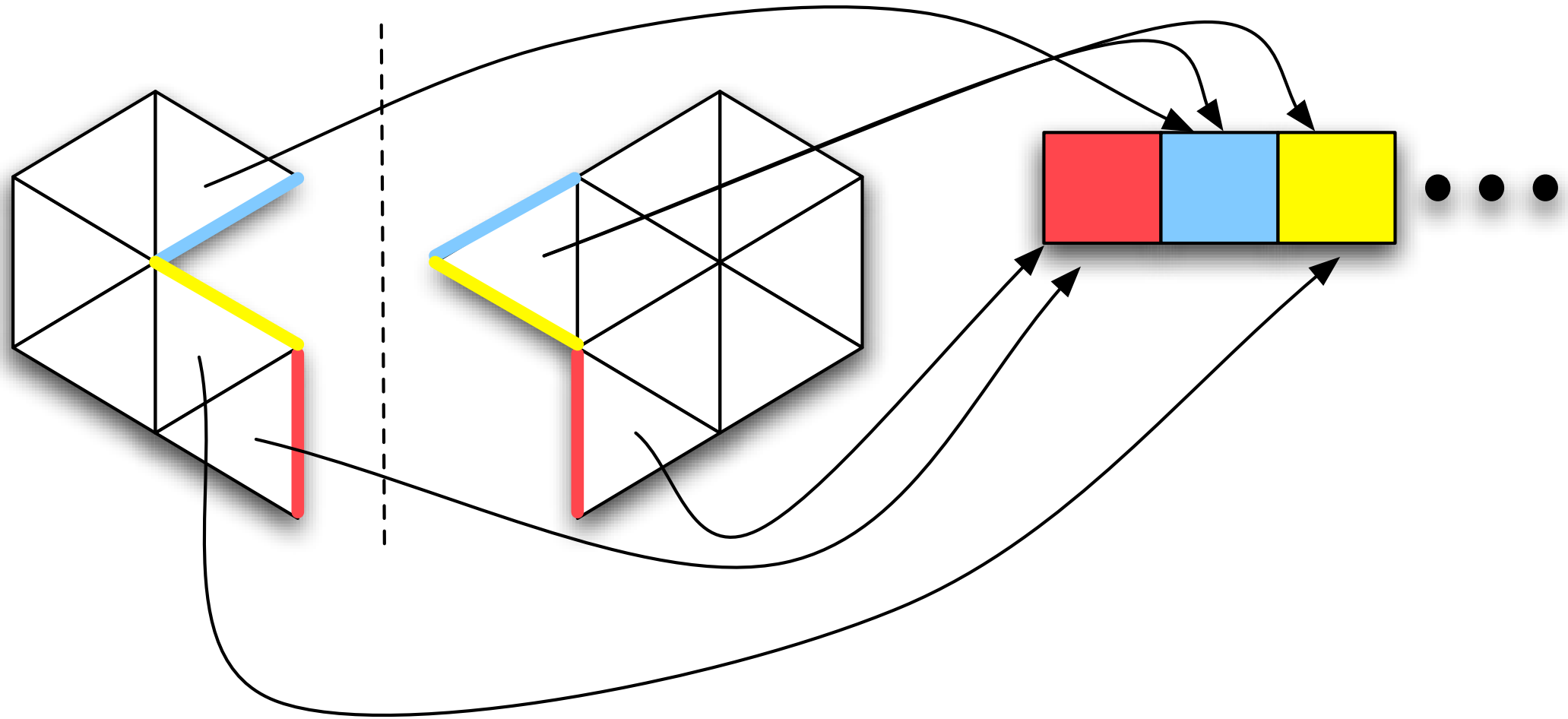


Distributed MSA
Hash Table

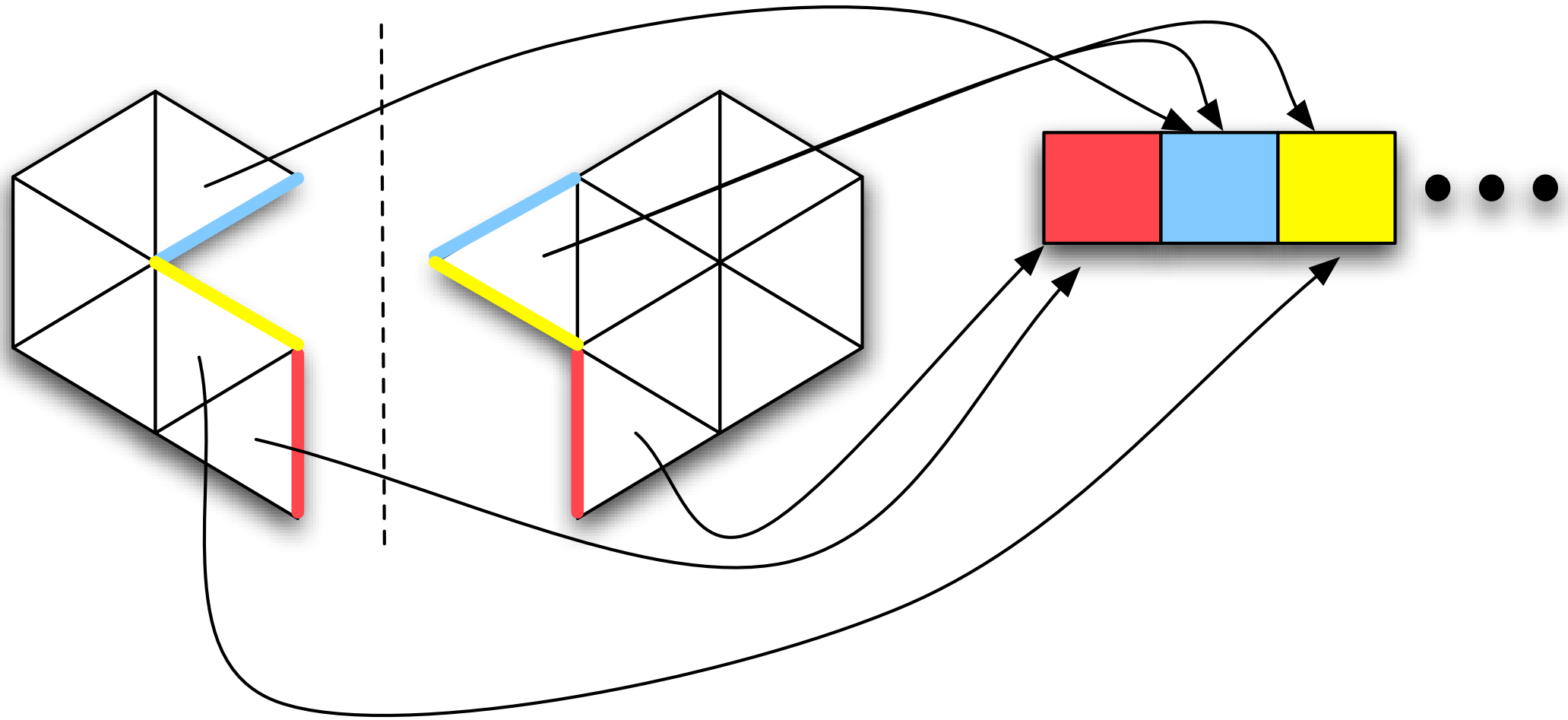




Each shared edge is hashed



Entries are added to the table in accumulate mode



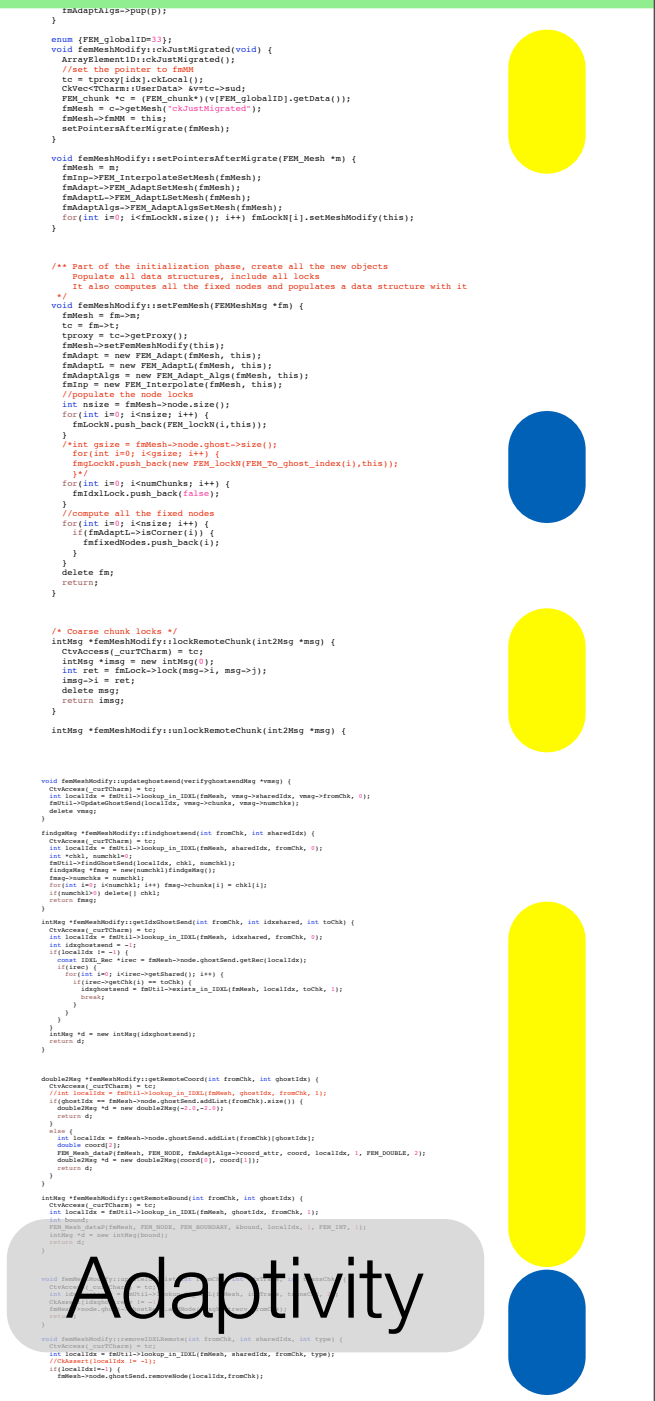
Now elements which collide in the table probably share an edge

Why is MSA good for this application?

Shared access to a global table is convenient when trying to determine which partitions you need to send to or receive from

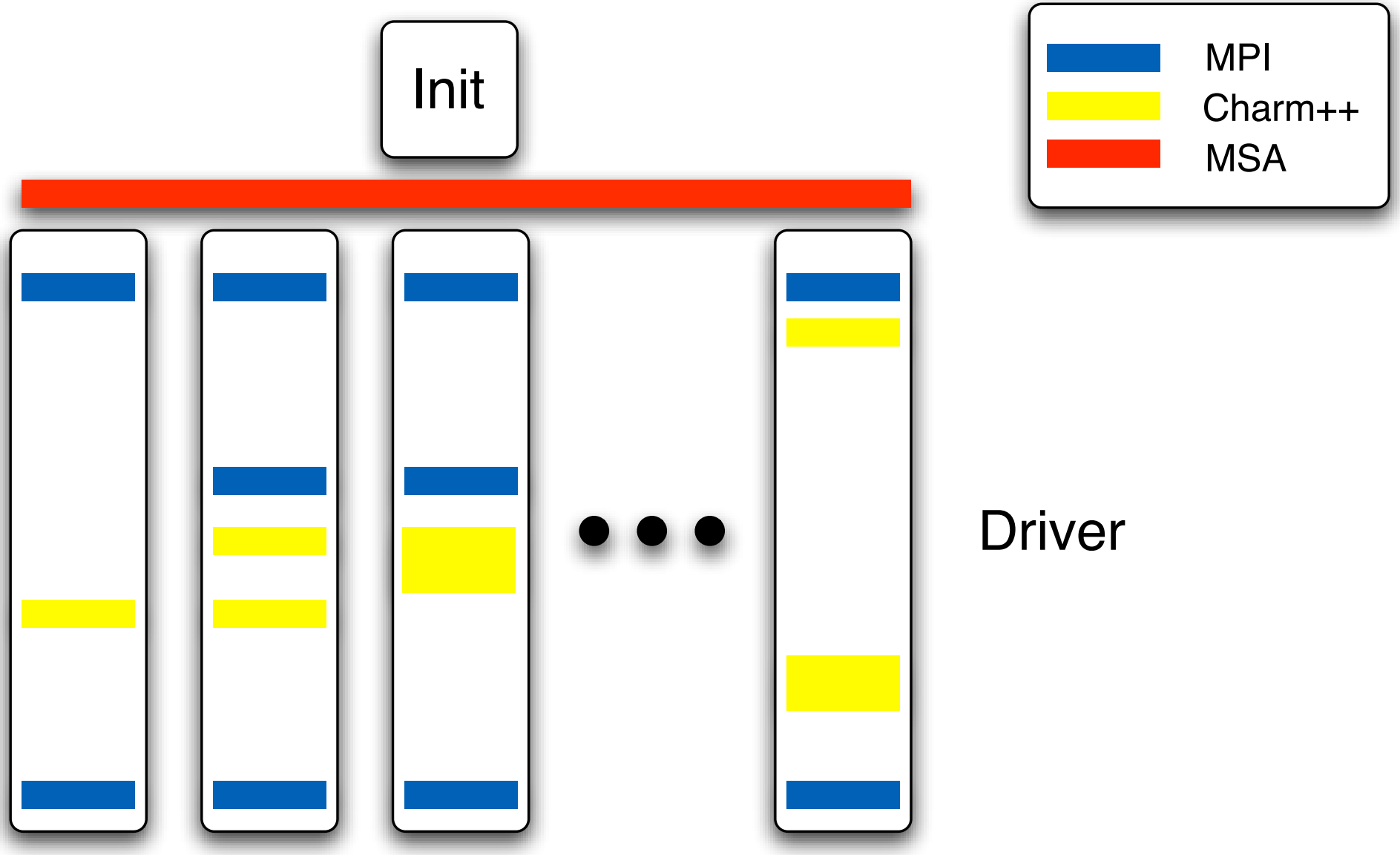
Filling and consulting the array fit neatly into MSA phases

How does this look in practice?



Partitioning

Adaptivity



A Typical ParFUM Program

Final Thoughts

Why should I avoid multiparadigm programming?

You can only program in languages you know

MPI is safe and popular

You need modularity

Language choice is limited by the underlying RTS

Why should I write multiparadigm programs?

Productivity

When adding new functionality to an existing program, you aren’t constrained by past language choices.

Why should I write multiparadigm programs?

Productivity

When adding new functionality to an existing program, you aren't constrained by past language choices.

Why should I write multiparadigm programs?

Productivity

When adding new functionality to an existing program, you aren't constrained by past language choices.

Why should I write multiparadigm programs?

When adding new functionality to an existing program, you aren't constrained by past language choices.

Because it is a multiparadigm program, ParFUM is:

- Easier to develop and easier to understand
- More extensible and flexible
- Still easy to use by MPI programmers

Charm++ is a great platform for multiparadigm programming, and I encourage you to try it out.

Multiparadigm Parallel Programming with Charm++, Featuring ParFUM as a case study

5th Annual Workshop on Charm++ and its Applications

Aaron Becker
abecker3@uiuc.edu
UIUC
18 April 2007