

Macroservers: A New High-Level Programming and Execution Model for PIM-Based Scalable Architectures

Hans P. Zima

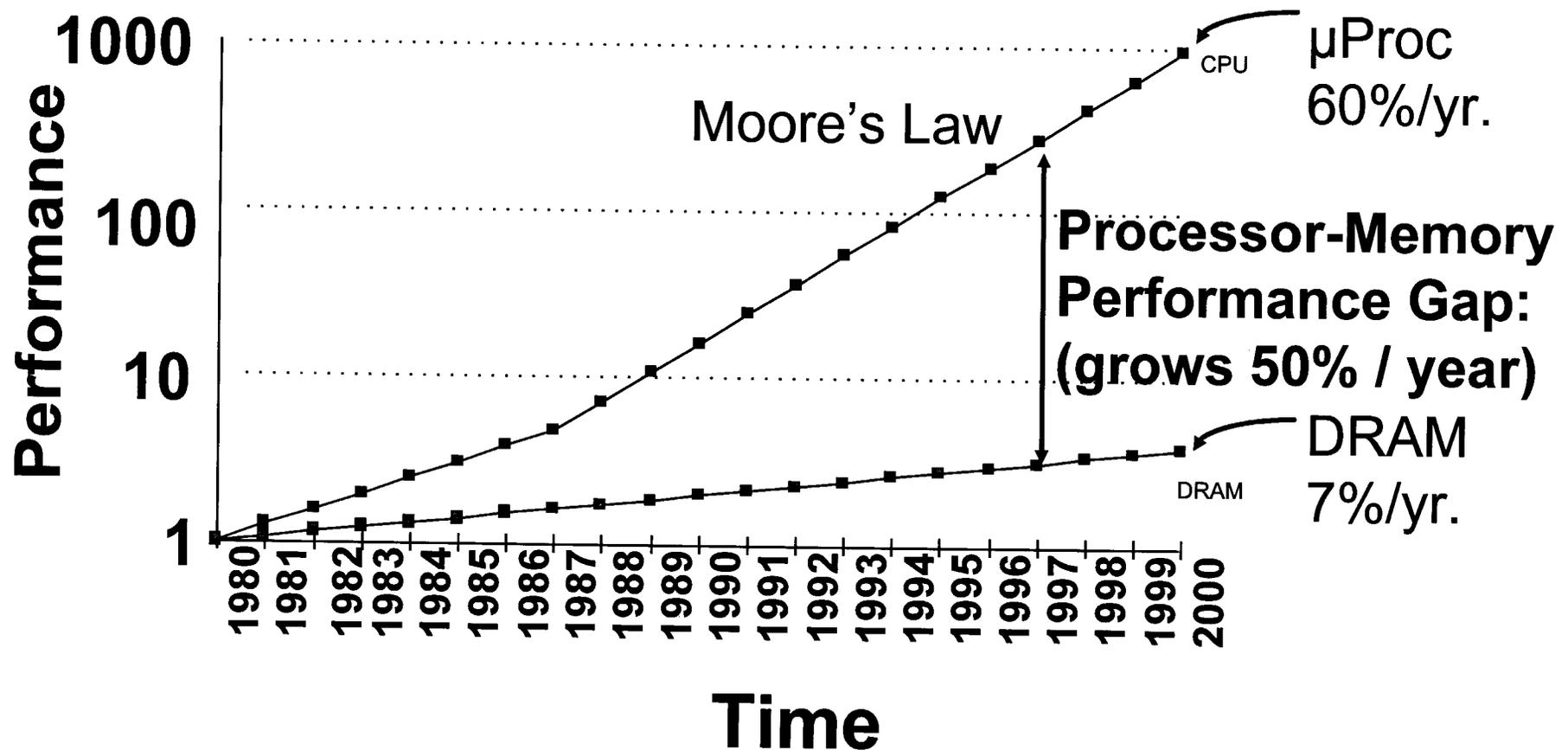
Engineering and Communications Infrastructure Section (366)

zima@jpl.nasa.gov

OUTLINE

- 1 Introduction
- 2 Processor-in-Memory (PiM) Systems
- 3 Macroserver
- 4 Research Issues
- 5 Conclusion

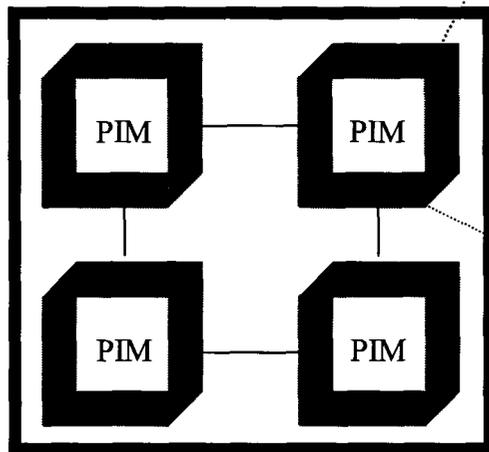
The Gap Between Processor and DRAM Performance



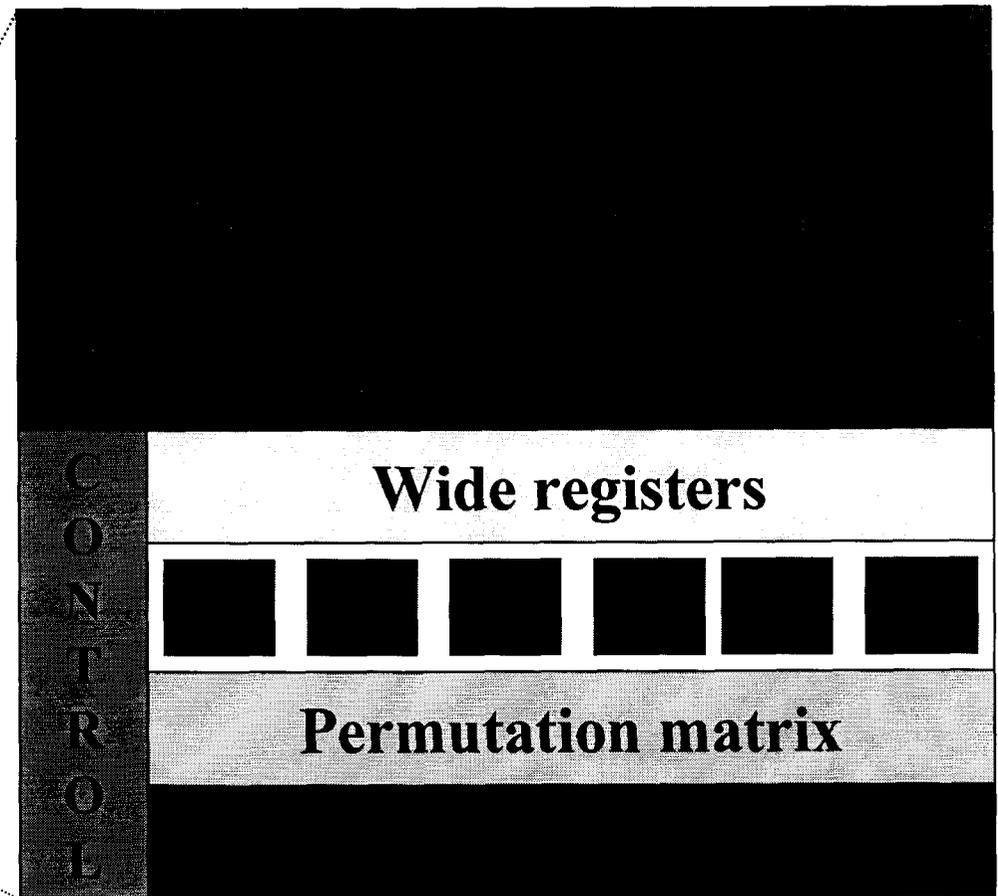
Source: D.A. Patterson "New directions in Computer Architecture" Berkeley, June 1998

Processor-in-Memory

- *Integration of CMOS logic/DRAM memory*
- *Replication of PIM nodes across module*
- *Multithreaded processors*
- *Huge improvement of on-chip bandwidth*
- *Lightweight threads in memory*
- *Lightweight communication via parcels*
- *Efficient memory operations*
- *No data caches*



PIM module (chip)

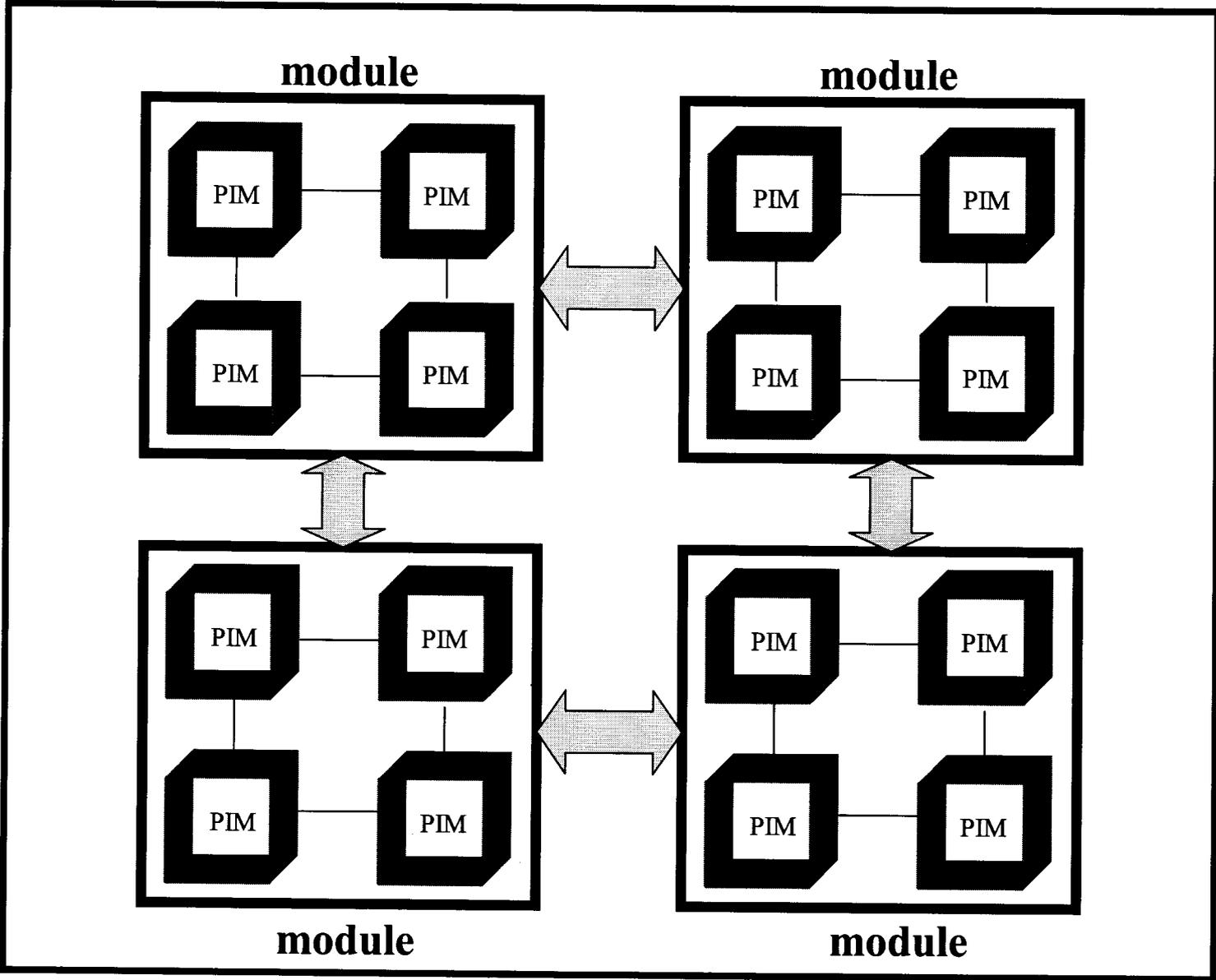


PIM node

PIM-Based System Configurations

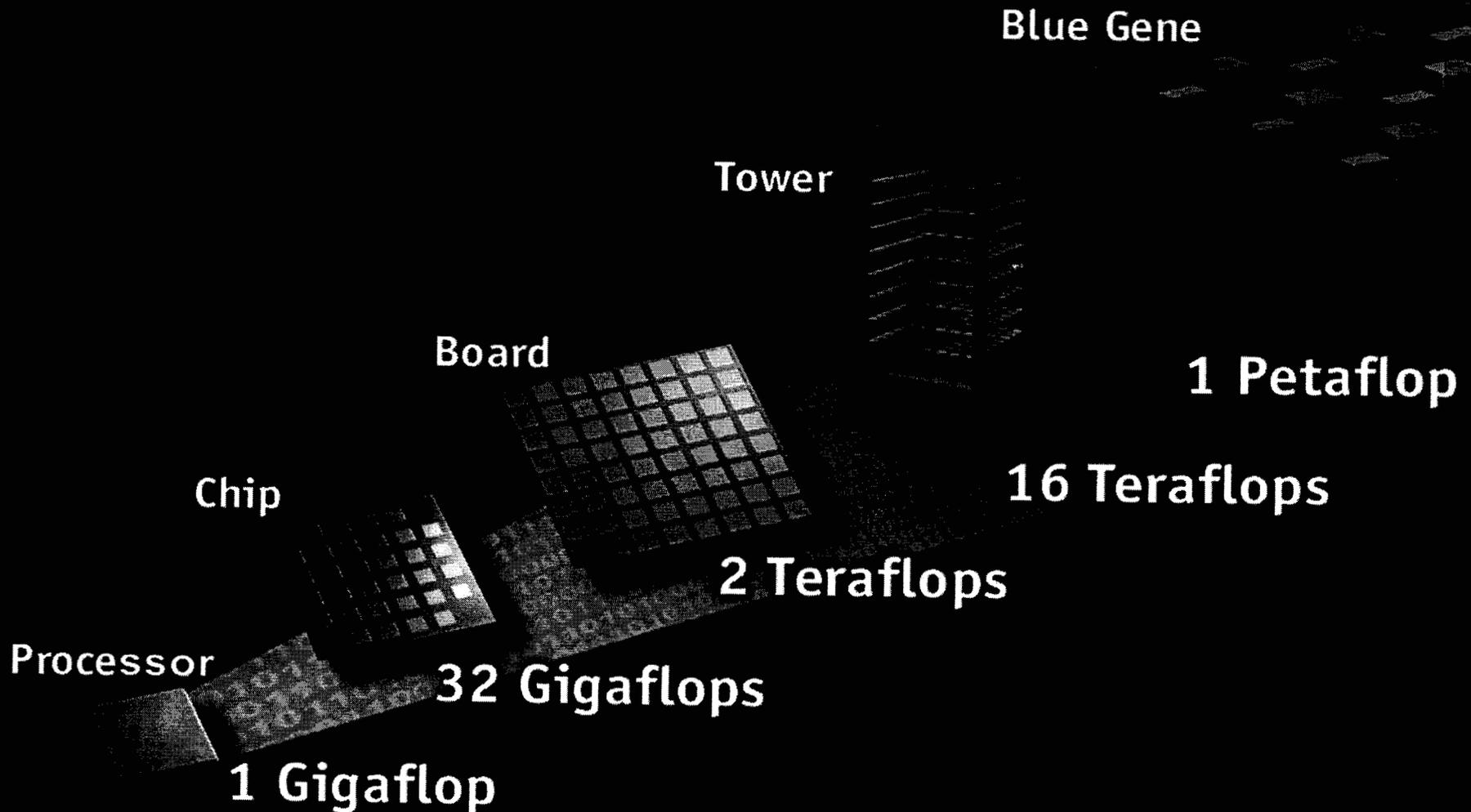
- ***PiM as a special-purpose processor:***
Mitsubishi M32 R/D (1996), IRAM (1997): embedded systems on a chip with vector processing logic
- ***PiM as memory in a conventional system:***
FlexRAM (UIUC, 1997), DIVA (University of Notre Dame, USC/ISI, Caltech, since 1998)
- ***PiM as a part of a memory hierarchy:*** HTMT (1998-), Cascade (2002-)
- ***PiM array:*** Blue Gene (IBM, since 1999), PIM-Lite (Notre Dame, 2002), Gilgamesh (Caltech/JPL)

Homogeneous PIM Array



IBM Blue Gene BG/C

Five Steps to a Petaflop Computer

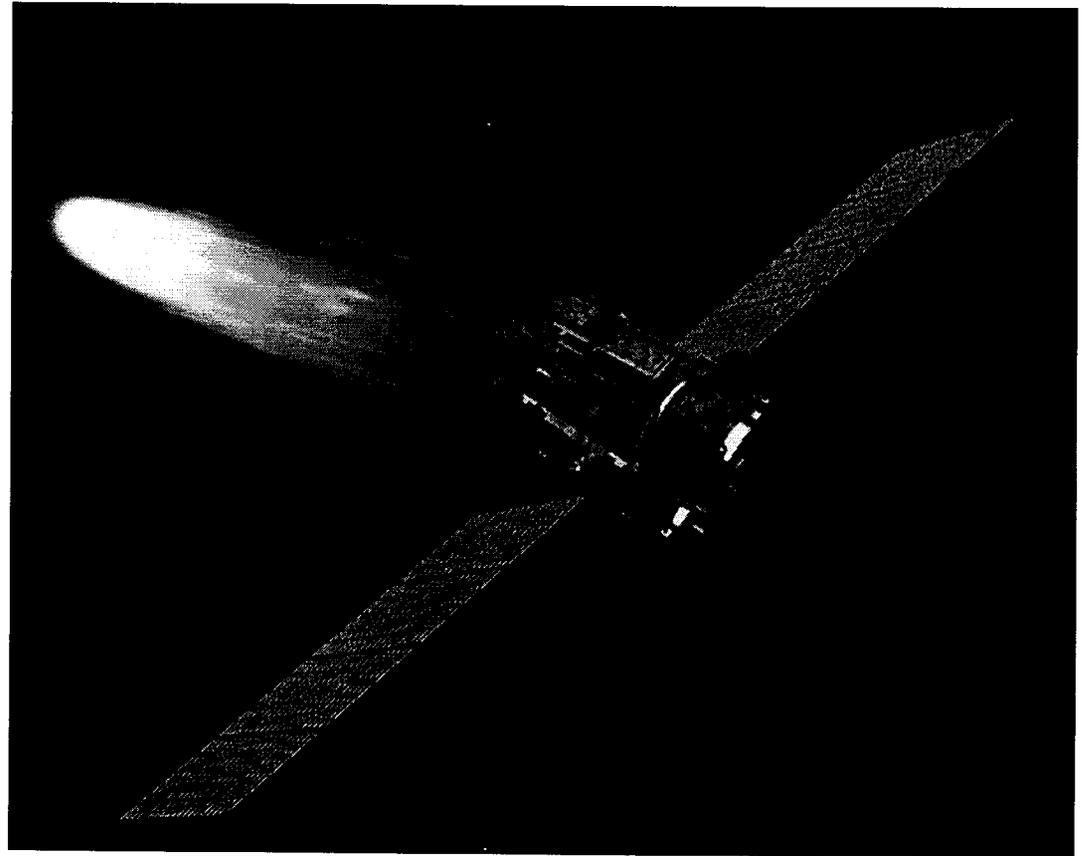


Application Domains

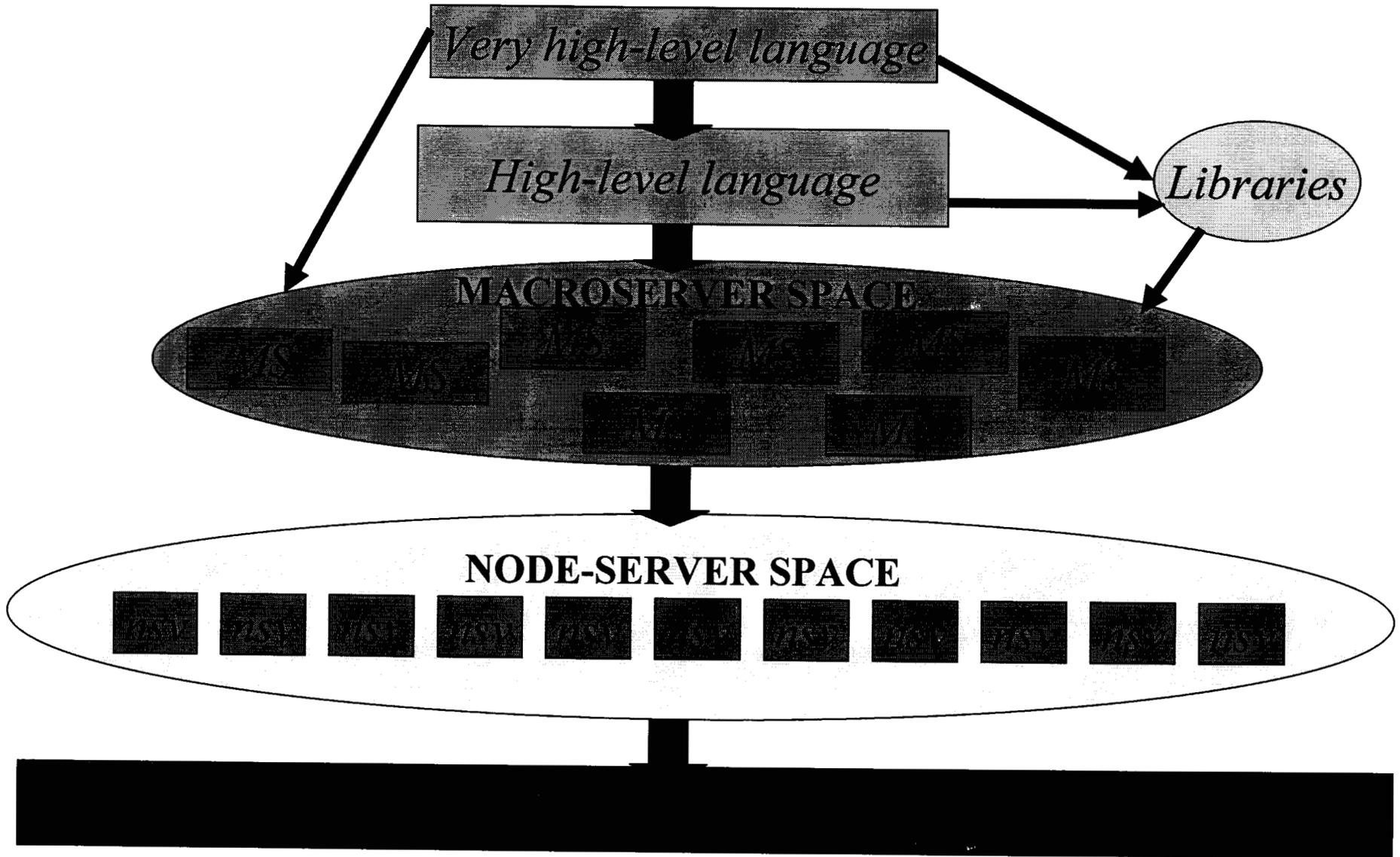
- Large-scale ground-based numerical computation
- Spaceborne systems (including autonomy)
- Monitoring and control (real-time)

Strategy

- Highly replicated fine-grain processors
- Virtual paged memory management
- Message-driven execution
- Multithreading
- Support for irregular data structures



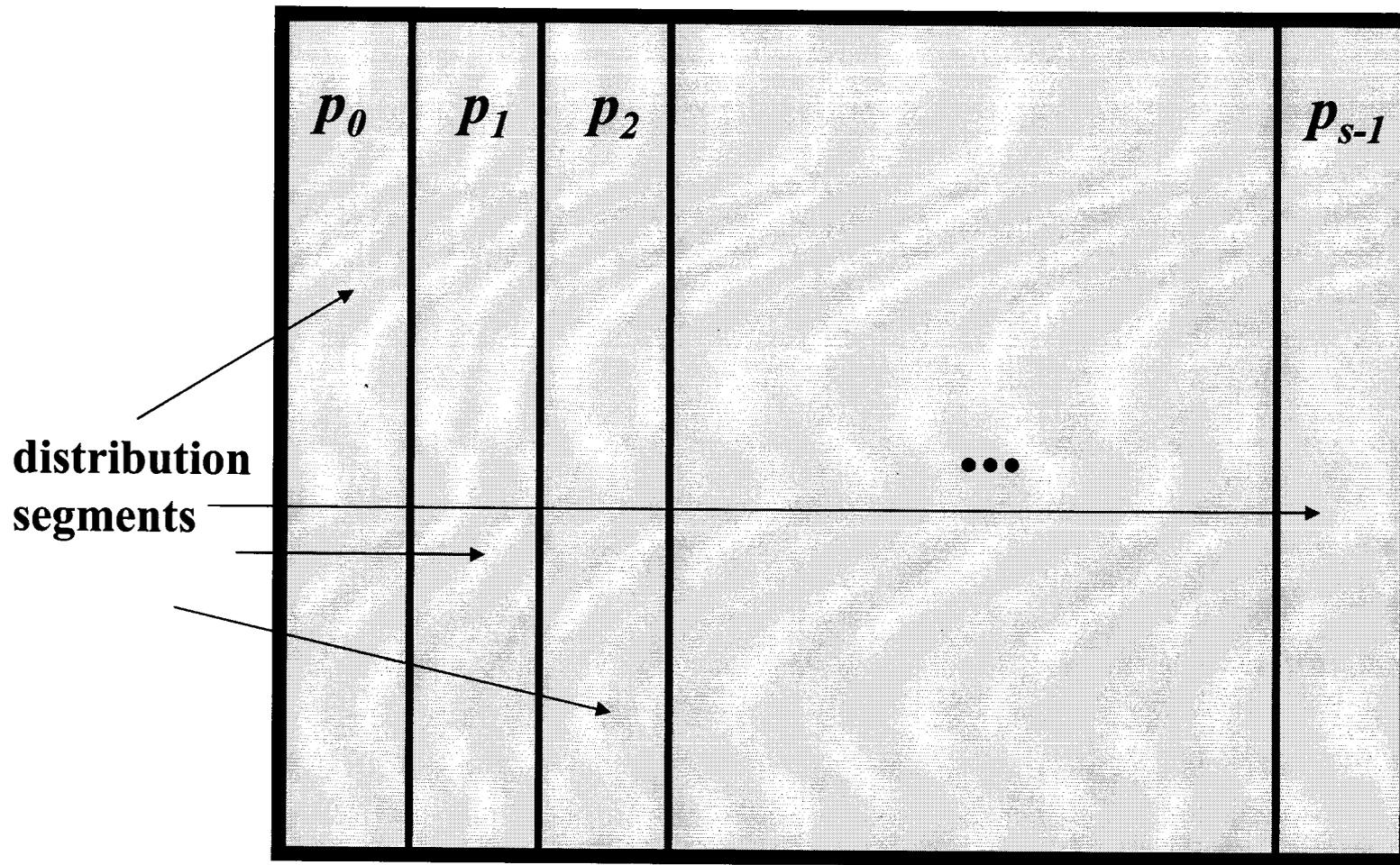
A Software Architecture for PIM-Based Systems



Macroservers

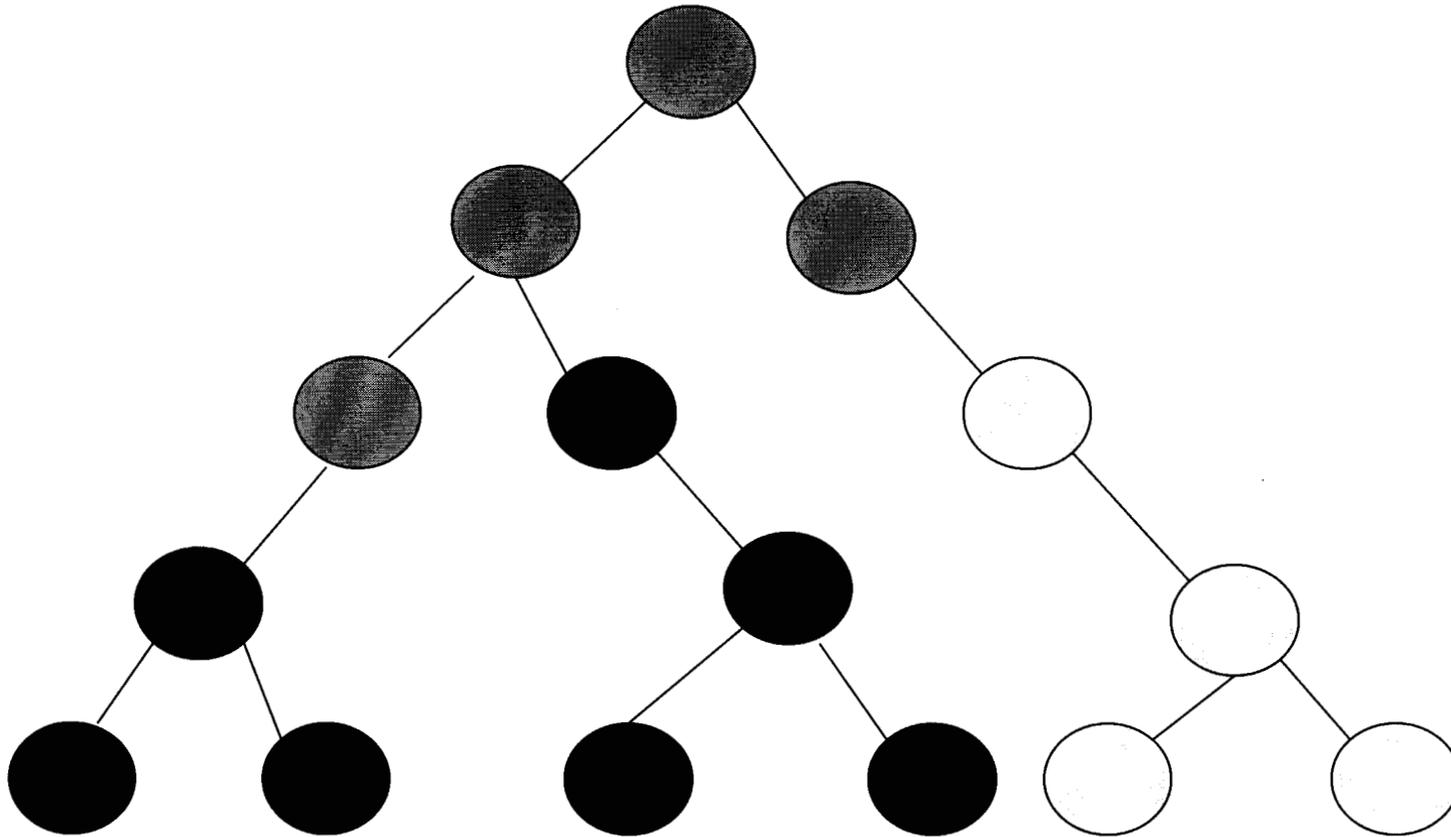
- **Middleware for object-based runtime management of data and threads; *target for high-level languages or very high-level specification systems***
- **Distributed collections *are a key concept: they consist of a data structure, together with a specification of a distribution (across “processors”).***
- **Specialization: *intrinsic object classes, distributions, and compilation methods (e.g., for sparse matrices)***
- **Application of special operations to distributed collections or sets of collections: *standard operators, reductions, prefix operations, filter***
- **Futures**
 - *spawning of structured data parallelism*
 - *creation of skeletons for the coordination of (data parallel or heterogeneous) tasks [Opus]*
 - *arbitrarily recursive task structures*

Column-block distribution of a 2D-matrix

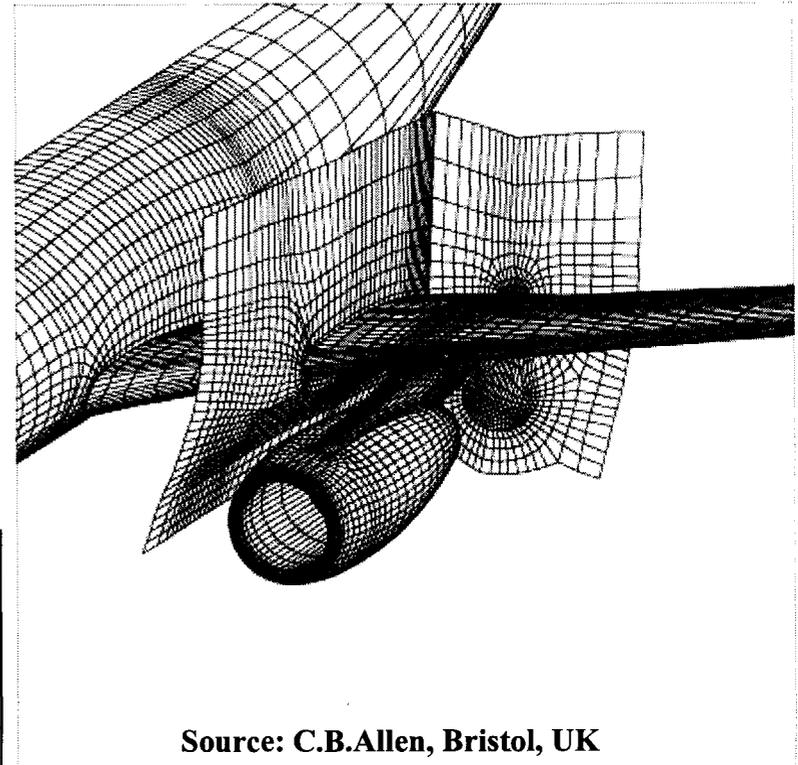
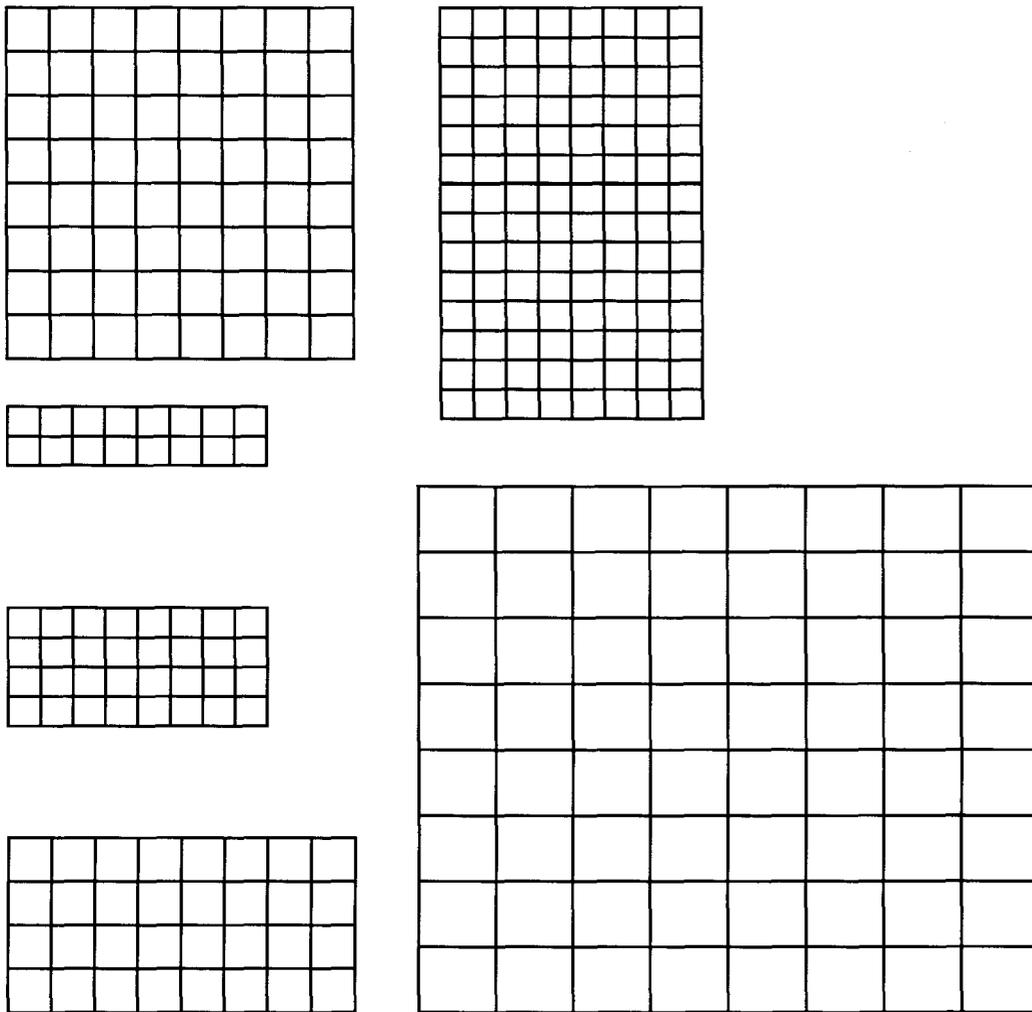


Regular distributions such as this can be easily handled in the compiler/runtime system

Example: Distribution of a Tree Structure



Example: A Multiblock Grid Collection



- *define partition of “processor” set*
- *distribute grids*
- *process grids in parallel*
- *run individual solvers in parallel*

Example: MRD/CRS Sparse Matrix Distribution

0	53	0	0	0	0	0	0	
0	0	0	0	0	0	21	0	
19	0	0	0	0	0	0	16	
0	0	0	0	0	72	0	0	
0	0	0	17	0	0	0	0	
0	0	0	0	93	0	0	0	
0	0	0	0	0	0	13	0	
					44	0	0	19
					37	0	0	0
					0	0	64	0

D ⁰	C ⁰	R ⁰
53	2	1
19	1	2
17	4	2
93	5	3
		3
		4
		5
		5

D ¹	C ¹	R ¹
21	2	1
16	3	1
72	1	2
13	2	3
		4
		4
		4
		5

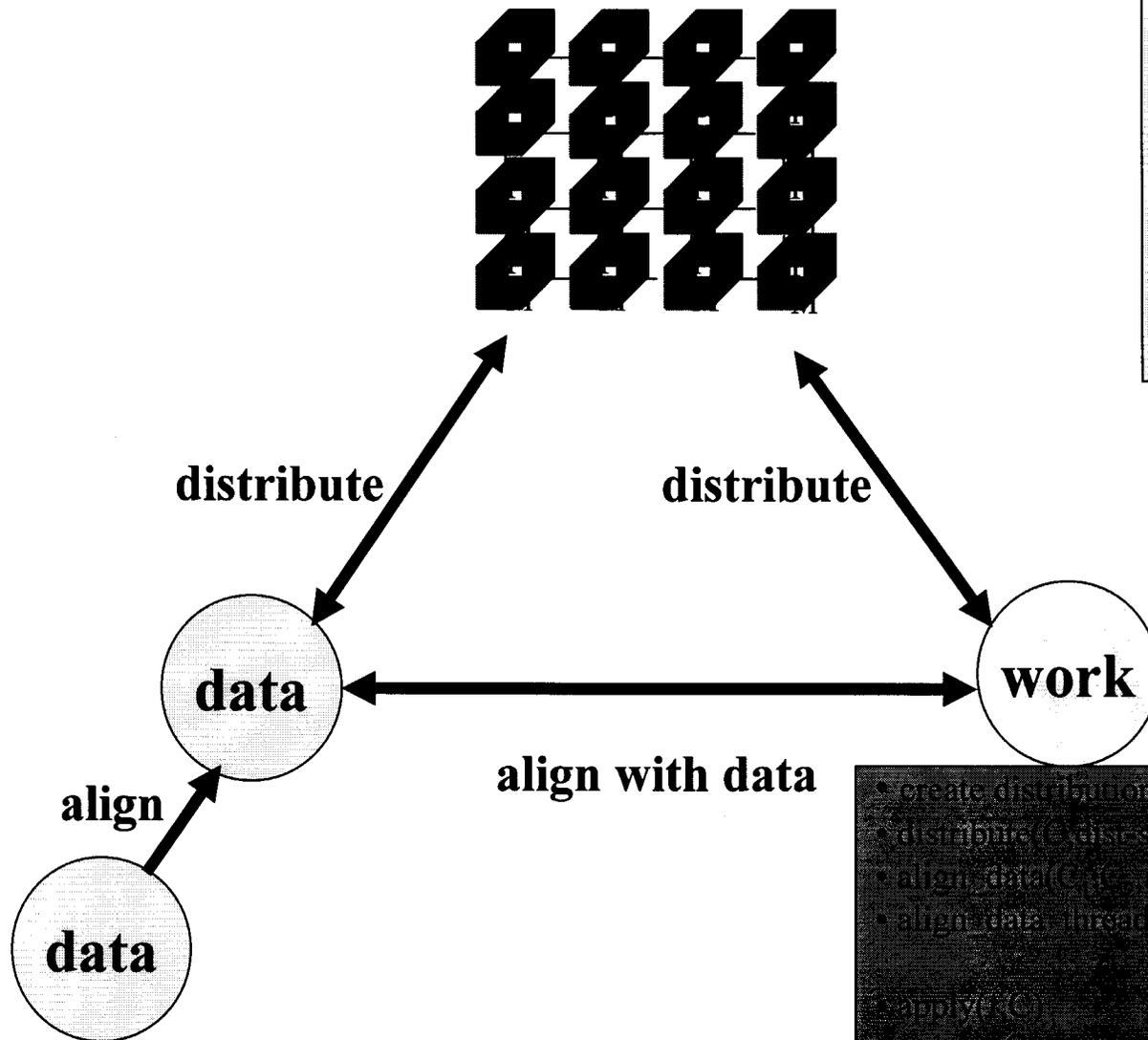
--	--	--

D ³	C ³	R ³
44	1	1
19	4	3
37	1	4
64	3	5

Requirements for Irregular and Dynamic Applications

- ***General data structures***
- ***General methods for distributing and aligning data***
(regular distributions may not reflect locality in physical space)
- ***General mechanisms for data/thread affinity***
(allow a dynamic mapping of thread groups to memory segments associated with a data partition)
- ***Dynamic manipulation of data distributions, alignments, and affinity must be efficient***
(apart from adaptive problems such as SAMR dynamic redistributions are even needed for regular problems such as ADI)

Distribution and Alignment Control



* Data distribution:

- mapping data to “processors”
- standard or user-defined
- first-class distribution objects
- dynamic redistribution

* Data alignment:

- establishing mapping between data domains

* Work distribution:

- mapping sets of threads to “processors”

* Work alignment:

- mapping sets of threads to distribution segments

```
• create_distribution_object(dist_spec, template(...))  
• distribute(C, dist_spec, target)  
• align_data(C, C, align_spec)  
• align_data_threads(C, C, align_spec)  
  
• apply(C)  
• reduce(C, operation)  
• reduce(C, operation, init)  
• apply(C, operation, init)
```

Intelligent Software

- **Self-adapting software**
- **Automatic, application-guided data and work distribution**
- **Feedback-oriented compilation: *interaction of compiler with dynamic performance analysis subsystem***
- **Verification and validation support**
- **Fault tolerance support**

Conclusion

- **Macroservers: *middleware for object-based runtime management of data and threads in homogeneous PIM-based architectures***
- ***Target for high-level languages or very high-level specification systems***
- ***Full control of locality and parallelism***
- ***Mapped to local node-servers on a PIM array***