

Parallel I/O on Highly Parallel Systems

SC '95 Tutorial M6

**Bill Nitzberg
and
Sam Fineberg**

**MRJ, Inc.
NASA Ames Research Center
Moffett Field, CA 94035-1000**

December 4, 1995



Outline

- Introduction: The Parallel I/O Problem
- Parallel I/O Hardware and Filesystems
- Efficient Algorithms for Parallel I/O
- Interfaces for Parallel I/O

Copies of these tutorial materials can be found at URL:

<http://www.nas.nasa.gov/home.html>

or by sending email to 'doc-center@nas.nasa.gov'.



Introductions

Name

Affiliation

- Computer Vendor
- University/Research Lab
- Industry

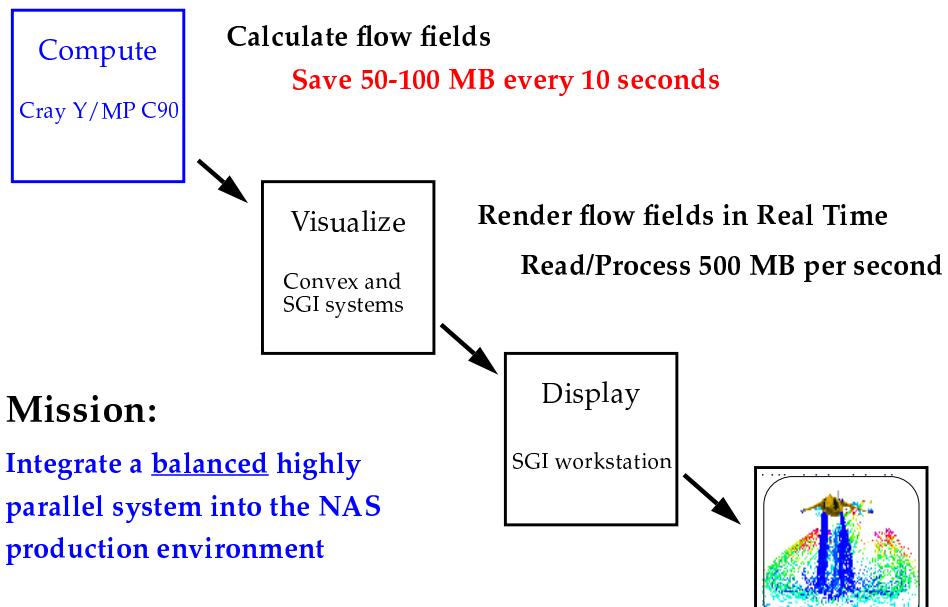
Category

- Learn — Student or Manager
- Manage — System Administration
- Develop — Systems Software
- Use — Scientific Applications

Interesting Parallel I/O Anecdotes



Processing Model at NAS



The Goal

Scientific Computing on Parallel Systems

Why?

- Traditional vector architecture's life is limited
- Commodity "off-the-shelf" pricing

But, Systems Lack:

- Industrial strength systems software
- Resource management (scheduling) facilities
- Parallel I/O performance on scientific applications



Our Focus

Parallel hardware provides substantial I/O capabilities, yet scientific applications' parallel I/O performance is abysmal.

1991—ARC3D [Ryan & Weeratunga]

~100 KB/sec out of 10 MB/sec on iPSC/860 CFS

1995—BTIO (straightforward impl.) [Fineberg]

~ 90 KB/sec out of 40 MB/sec on Paragon PFS

~ 1 MB/sec out of 40 MB/sec on SP2 PIOFS

"Inefficient and immature input/output subsystems have emerged as a major performance bottleneck on scalable parallel systems." [Crandall+, 1995]

"No one does I/O, because it's too slow" [Saphir, 1995]



Parallel I/O

"Single" parallel application run on many nodes
Data is distributed among the node memories
Data is transferred to/from a single (logical) file
The file is itself spread across nodes and disks.

Not: distributed file systems, database systems,
transaction processing, RAIDs, or striped HIPPIs



Highly Parallel System

Distributed Memory Machines

- Commodity CPUs
- High-performance network
- Commodity disks
- 50-2000 processors

Examples

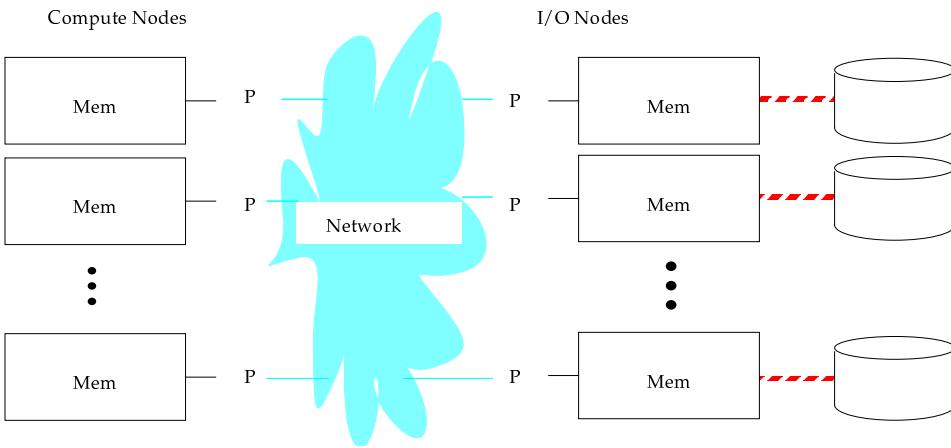
- CS-2, Paragon, SP2, T3D, etc.
- Workstation clusters (SGI, Sun, etc.)

Not traditional vector supercomputers (Crays)

Not shared memory workstations



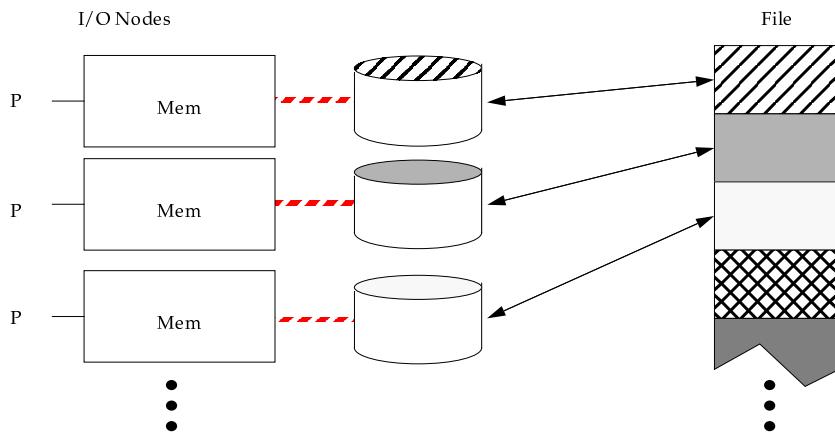
Parallel I/O Hardware



- Applications run on compute nodes
- Files are managed by I/O nodes
- All data is sent in messages over the network



Parallel Filesystem Architecture



Parallel I/O subsystems are modeled after serial systems

- favor large accesses (10s - 100s of kilobytes)
- favor sequential accesses



Scientific Applications

Major classes of scientific application:

- Data Analysis (e.g. seismic processing) READ
- Data/Vis Generation (e.g. CFD) WRITE
- Out-of-core (e.g. large FFTs) READ/WRITE

Common characteristics of “hard” scientific applications:

- use multidimensional arrays
- not embarrassingly parallel
- memory bound



Brief History of Parallel I/O

Spread files across multiple disks (1984-1988)

“Disk Striping”, Salem & Garcia-Molina, Princeton, 1984
“RAID”, Patterson+, UC Berkeley, 1987

First “Parallel File Systems”: interleaved files (1987-89)

“Bridge”, Ellis, Dibble, Scott, Duke & U. Rochester, 1987-89
“DataVault”, Thinking Machines CM-2, 1988
“CFS”, Intel iPSC/2, Pierce, 1989

Experimental studies of parallel file systems (1989-1992)

“iPSC/2 CFS”, Bradley & Reed, UIUC, 1989
“nCUBE/10”, Pratt, French, et. al., U. Virginia, 1989
“DataVault”, Krystynak, NASA Ames, 1992
“iPSC/860 CFS”, Nitzberg, NASA Ames, 1992



New Interfaces (1991-1993)

- "ELFS", Grimshaw & Loyot, U. Virginia, 1991
- "PFS", Intel Paragon, 1992
- "Vesta (PIOFS)", Corbett+, IBM Yorktown, 1993

Studies of Parallel Scientific Applications (1994-1995)

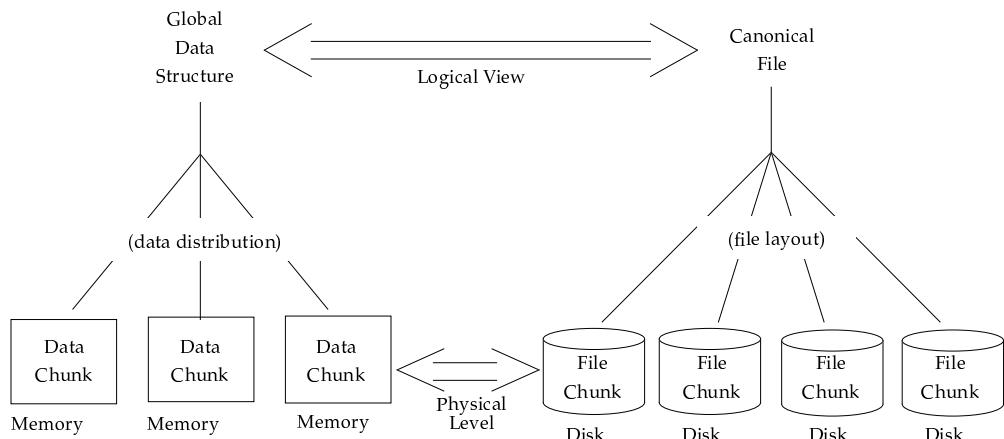
- "CHARISMA", Kotz+, Dartmouth, 1994
- "Scalable I/O Initiative", Crandall+, UIUC, 1995

Compilers, Libraries, Algorithms (1993-1995)

- "Scalable I/O", Kennedy+, Rice, 1993
- "PASSION", Bordawekar+, Syracuse, 1993-1995
- "MPI-IO", Corbett+, IBM & NASA Ames, 1994-1996
- "Jovian", Bennett+, U. Maryland, 1994
- "Panda", Seamons & Winslett, UIUC, 1994
- "PIOUS", Moyer & Sunderam, Emory U., 1994
- "Disk-directed I/O", Kotz, Dartmouth, 1994
- "PPFS", Huber, Chien, et. al., UIUC, 1995

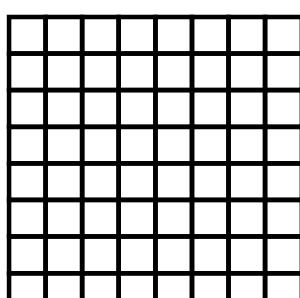


Parallel I/O: The Global Perspective

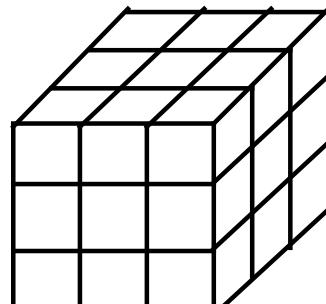


A global data structure is read/written to a file.

Global Data Structure



2D Array



3D Array

Scientific applications typically operate on multidimensional arrays



Data Distribution

Data is distributed among compute nodes for performance

- increase parallelism
- improve load balance
- minimize communication overhead

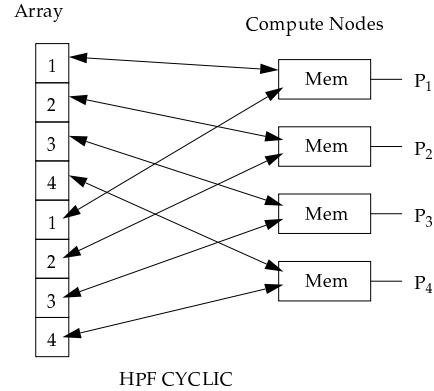
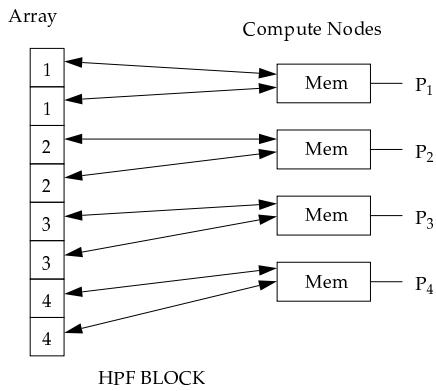
We use HPF distribution directives as a descriptive tool:

- convenient for describing distributions in general
- but, not general enough to describe everything

Two directives: BLOCK and CYCLIC



HPF Distributions



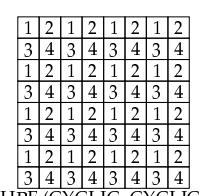
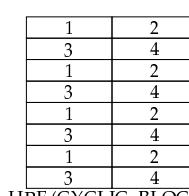
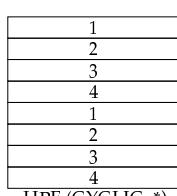
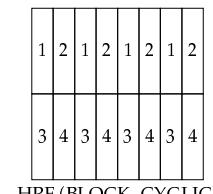
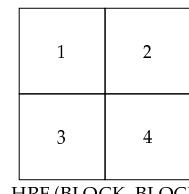
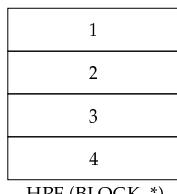
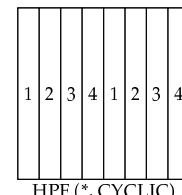
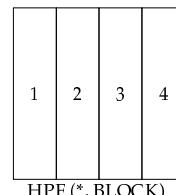
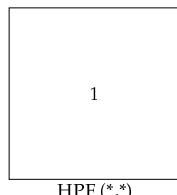
BLOCK—cut data into equal-sized chunks

CYCLIC—deal out small pieces round-robin

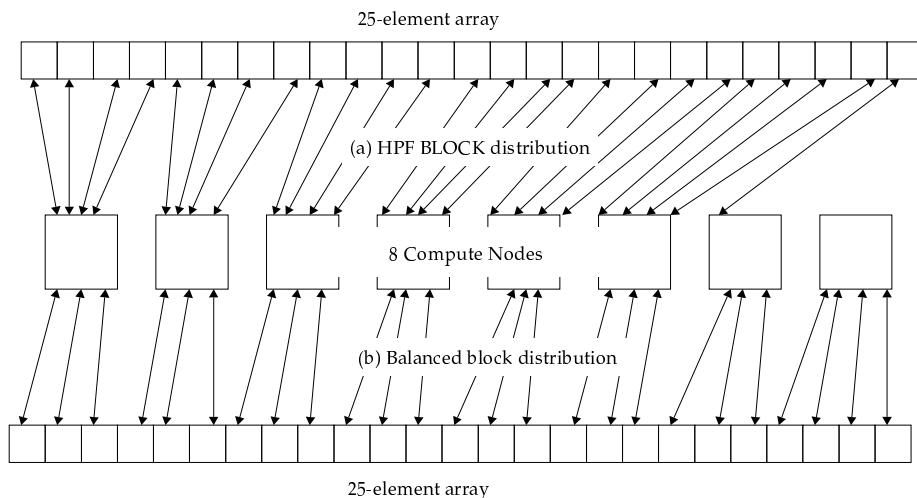
The size of the pieces (d) is specified as an argument: CYCLIC(d)



HPF 2D Distributions



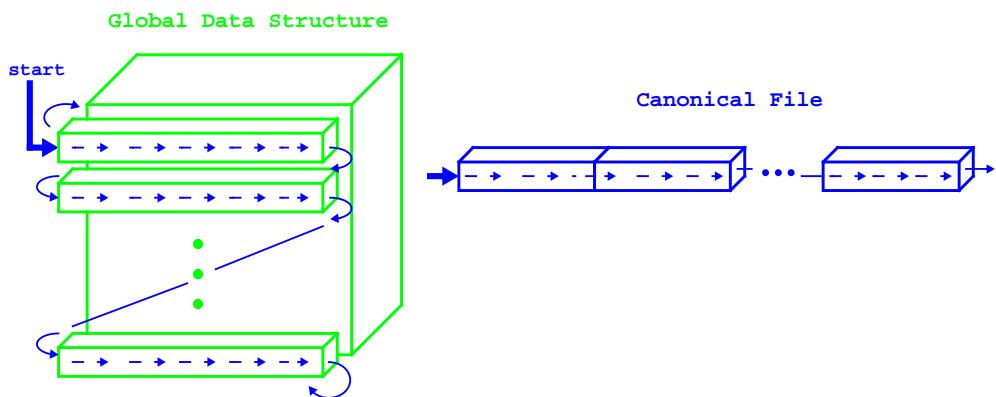
HPF BLOCK vs. Balanced



Straight HPF BLOCK distributions can lead to serious load imbalance, especially in multi-dimensional arrays



Canonical File

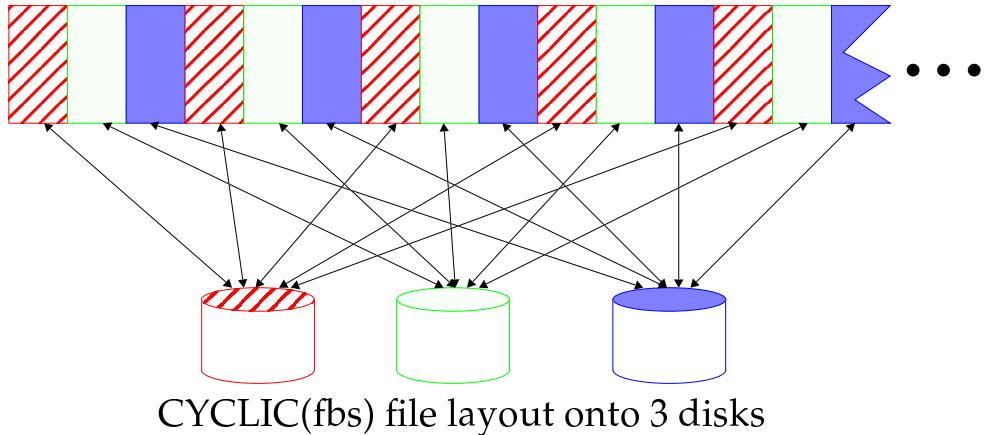


- one dimensional linearization of the global data structure (e.g. row-major ordering of a 3D array)
- basis for parallel I/O interfaces and file layout



File Layout

Canonical File—a sequence of file blocks...



All commercial parallel file systems use an
HPF CYCLIC(fbs) file layout of the canonical file



BT: An Example Scientific Application

NAS Parallel Benchmarks [Bailey+]

- de facto standard for supercomputer performance
- “pencil and paper” benchmarks

BT—the block-tridiagonal pseudo-application

- simulates high speed compressible airflow by solving a system of partial differential equations using the ADI method
- general structure represents over 20% of the CFD workload at NAS



BT Algorithm

BT simulates airflow in timesteps:

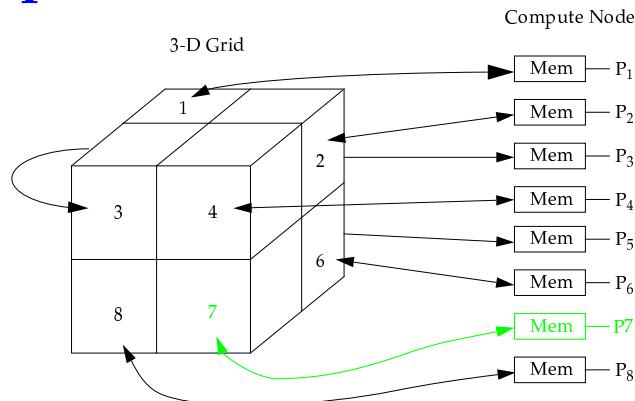
- Air is modeled in a 3-D array of (velocity, temperature, pressure) values.
- For each timestep, airflow is approximated by traversing the array in the X, Y, then Z dimension.
- For each traversal, all lines can be done in parallel.

Three common methods for parallel implementation:

- Pipelined Gaussian Elimination
- Dynamic Block-Cartesian Decomposition
- Multi-partition



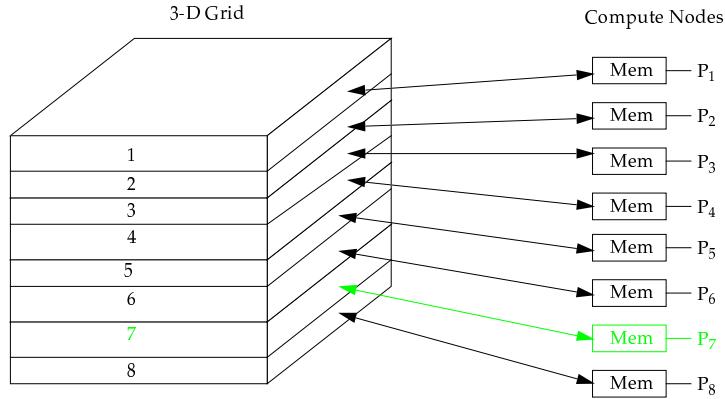
BT: Pipelined Gaussian Elimination



- Subcubes minimize communication overhead
- Requires pipeline-fill for every traversal
- Best when P is a perfect cube



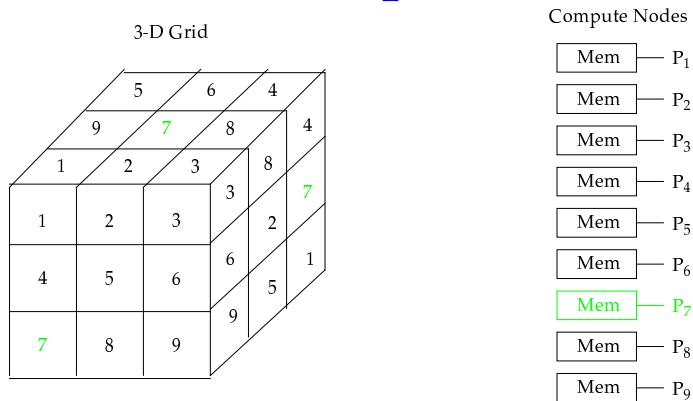
BT: Dynamic Block-Cartesian Decomp.



- Requires 3 full transposes for every 2 timesteps
- Fully load balanced
- Best when P divides grid dimensions



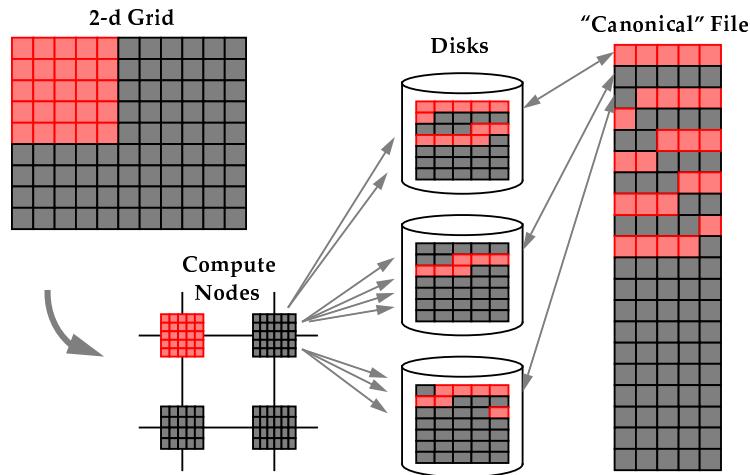
BT: Multi-partition



- Subcube minimize communication overhead
- Fully load balanced
- Best when P is a perfect square



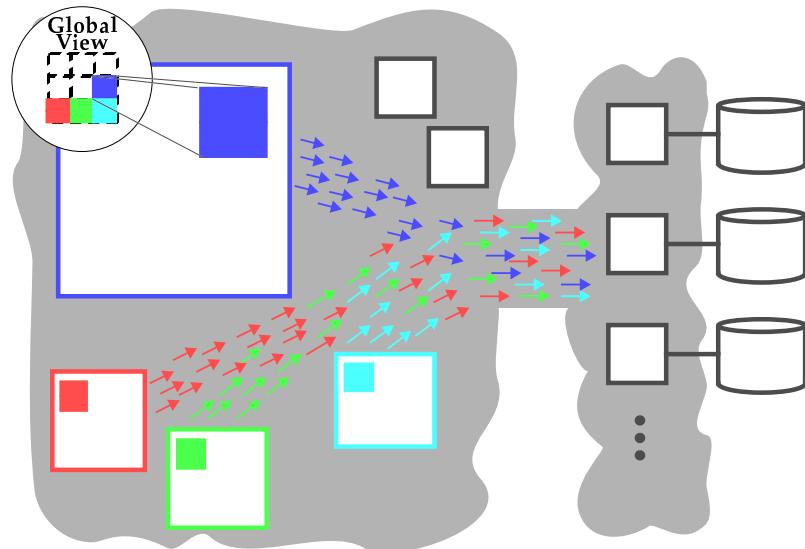
Multi-dimensional Data Distributions



Small data blocks (100 - 1000 bytes) are an inherent property of parallelizing scientific applications.



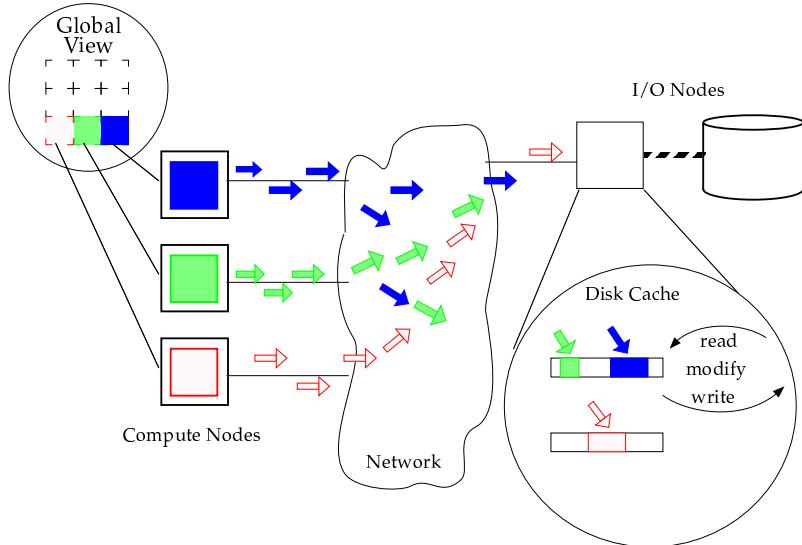
"Naive" Algorithm



A straightforward implementation of file I/O using current parallel I/O interfaces results in **uncoordinated accesses**.



Parallel Filesystems



Numerous small uncoordinated accesses result in significant overhead and potential thrashing.



Parallel I/O: The Low Level Perspective

"Good" data distributions inherently lead to: [CHARISMA]

- small (< 1000 bytes) data blocks
- non-sequentiality

The "Naive" algorithm leads to

- uncoordinated access
- read/modify/write or read/read overhead
- potential thrashing

Current filesystems want large sequential accesses

Mismatch results in abysmal performance

