A Parallel I/O Runtime System for Irregular Applications and Its Optimizations

Sung-Soon Park
Anyang University

sspark@aycc.anyang.ac.kr

April 24, 1999
KISS Spring Conference

Outline

Parallel I/O

- Why, What, and How ….

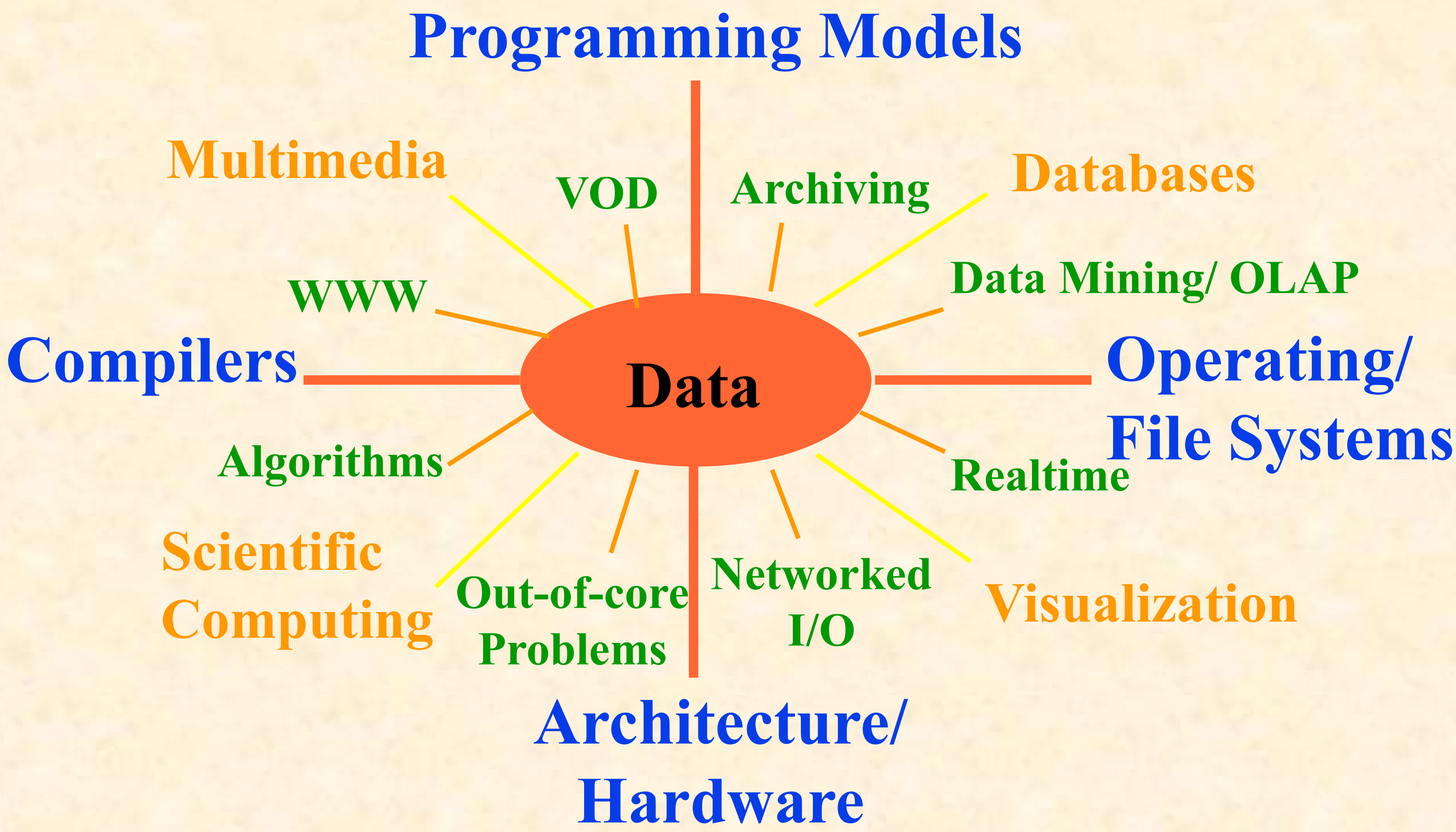
I/O work for Irregular Application

- Irregular application and its Optimizations

Future Trends of I/O in Software

Further Information

Parallel I/O : Data-centric Computing Universe



Parallel I/O : The I/O Problem

- Tremendous increase in the CPU performance
- Sequential machines becoming powerful
- Parallel machines becoming even more powerful
- Applications: more complex and data-intensive
- Programming models
  - More abstract
  - Difficult to optimize
- Applications tend to become I/O-bound
- **Need: To balance I/O with CPU performance**

Parallel I/O :

Four Techniques for improving I/O performance

- Locality optimizations and effective data caching
- Overlapping data accesses with computations
- Aggressive scheduling of data accesses
- **Parallelizing data accesses**
  - Exploiting user-level parallelism (Software)
  - Increasing storage device parallelism (Hardware)

Parallel I/O : Concepts

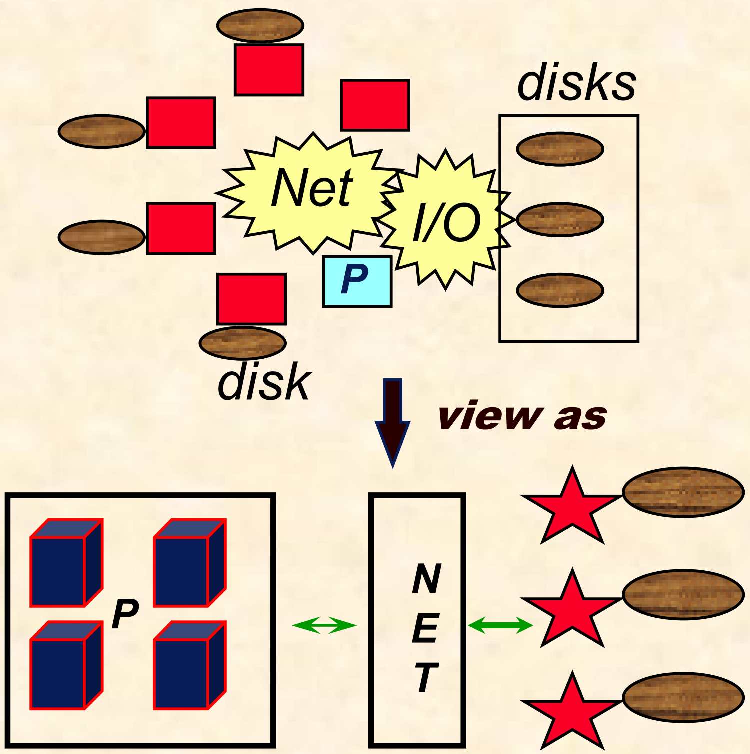
- Multiple processes participate
- Application is aware of parallelism
- Preferably the “file” is itself stored on a parallel file system with multiple disks.
- That is, I/O is parallel at both ends:
  - application program
  - I/O hardware

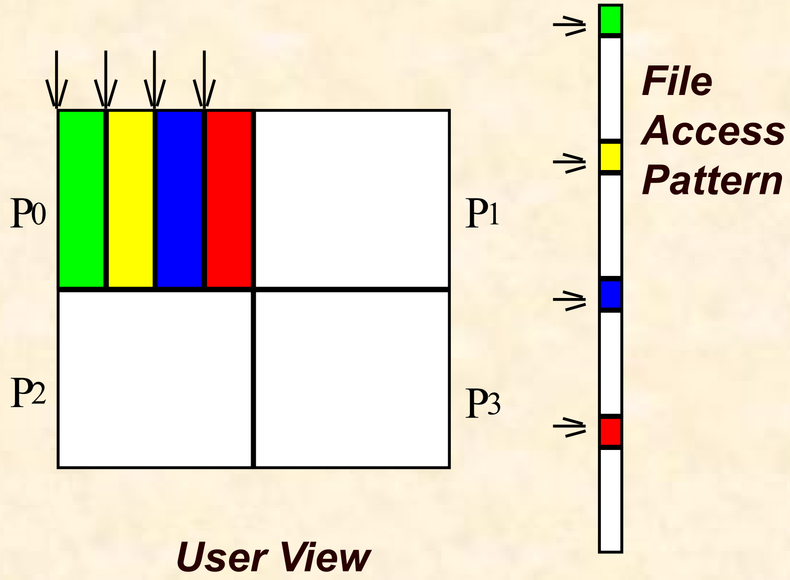
Parallel I/O : Requirements / Characteristics

- Compulsory (read/write, initialization..)
- Check-pointing
- Out-of-Core I/O (disk resident data)
- Visualization
- QoS/Guaranteed Performance
- Analysis/Mining of large disk-resident data
- High throughput transactions oriented-accesses

Problem: Support needed for different types of I/O

Parallel I/O : Issues





Problem: Exploiting I/O parallelism in a balanced manner

Problem: Interface and coordinating I/O from distributed structures

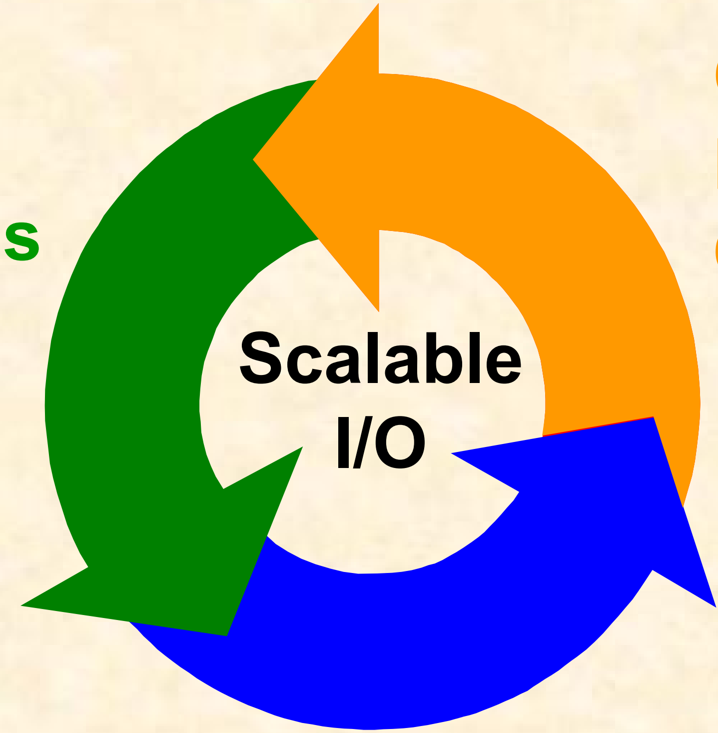
Parallel I/O : Problems of I/O

- Applications exhibit different I/O characteristics
- Different views in user and I/O spaces
- Different solutions at different levels
  - Architecture
  - Software
  - Storage Hardware
- Performance depends on how these solutions interact
- **Ping-pong effect**
  - Bottle-neck shifts between hardware and software
- **Need: Make parallel I/O scalable**

Parallel I/O : An Integrated Solution for Scalability

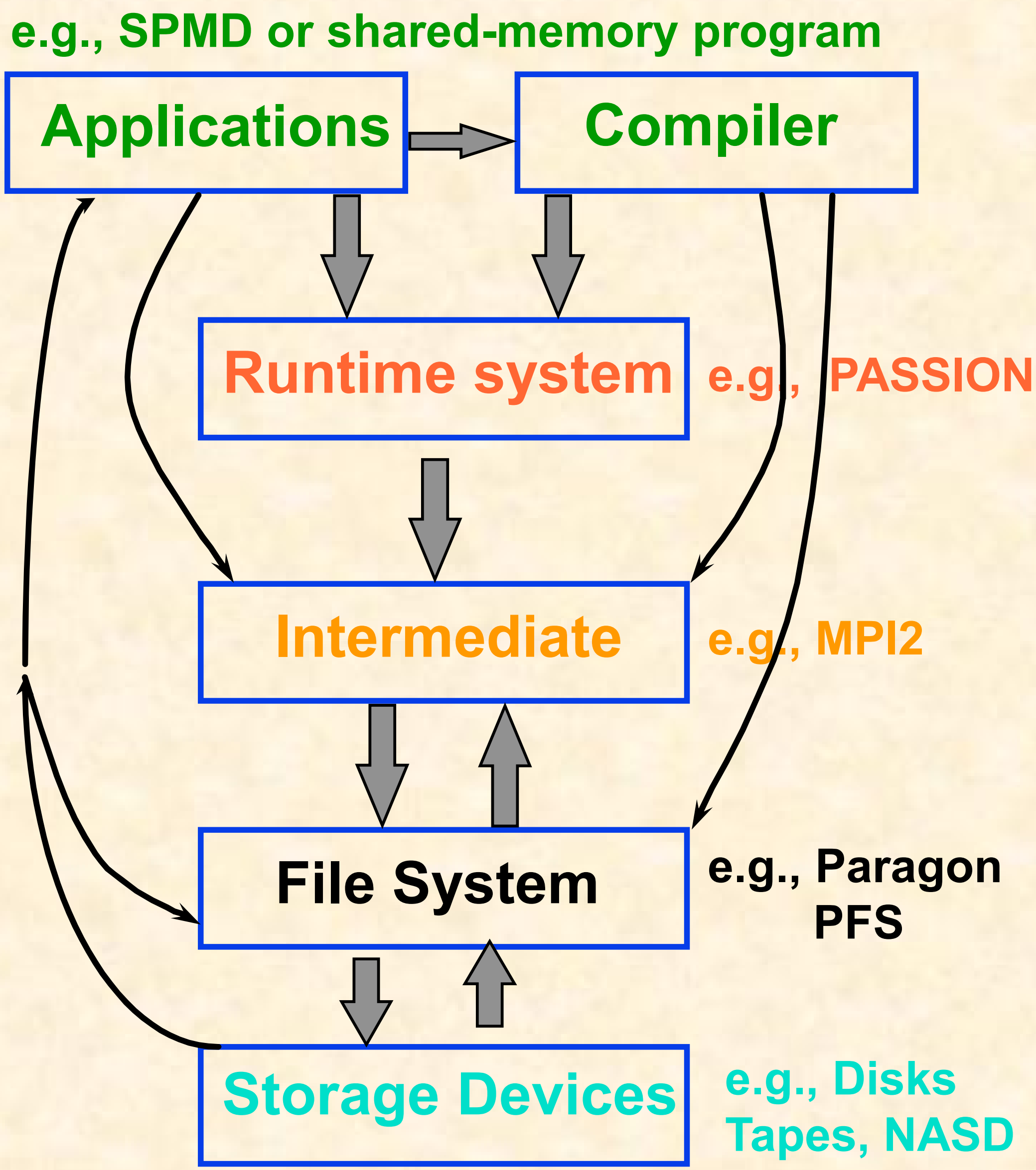
Compilers
Runtime Systems
OS/File Systems

Applications



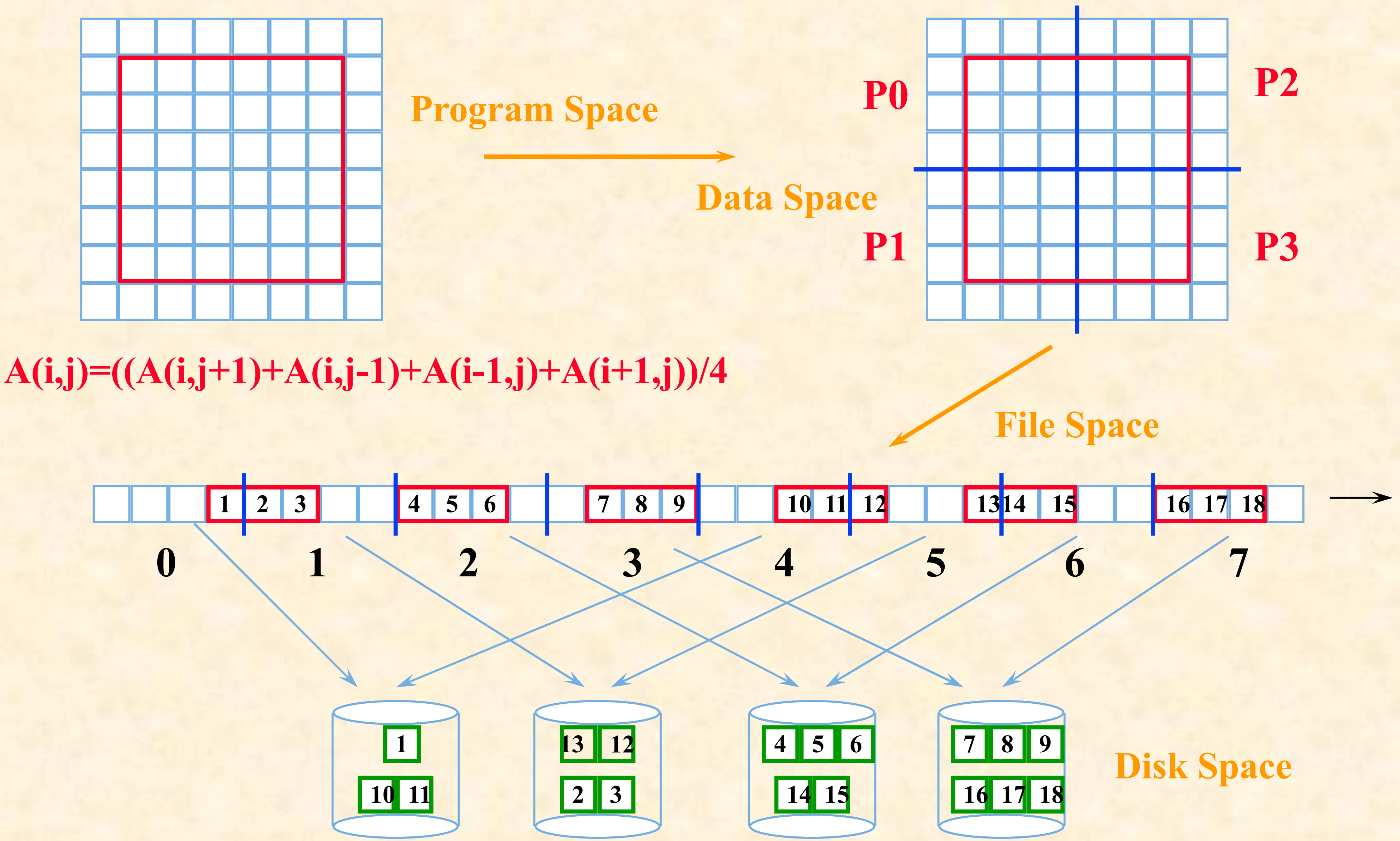
Architecture
Interconnection Networks
Storage Hardware

Parallel I/O : Typical Software Layers for Scalable I/O



- Domain and semantic info available (checkpoint array, read mesh)
- Semantic info preserved and used (read/write section of an array)
- Some data type info available, But, linear domain
- Stream of bytes, no semantic or data type information,
- Stream of bytes, data placement issues

Parallel I/O :
Operating Spaces in Parallel I/O-intensive Applications



Parallel I/O : Background of Collective I/O

Compute nodes or **I/O servers** share

data access and *storage information*

and use it to make

independent,

non-overlapping

and *coordinated* accesses that

conform with the data storage pattern

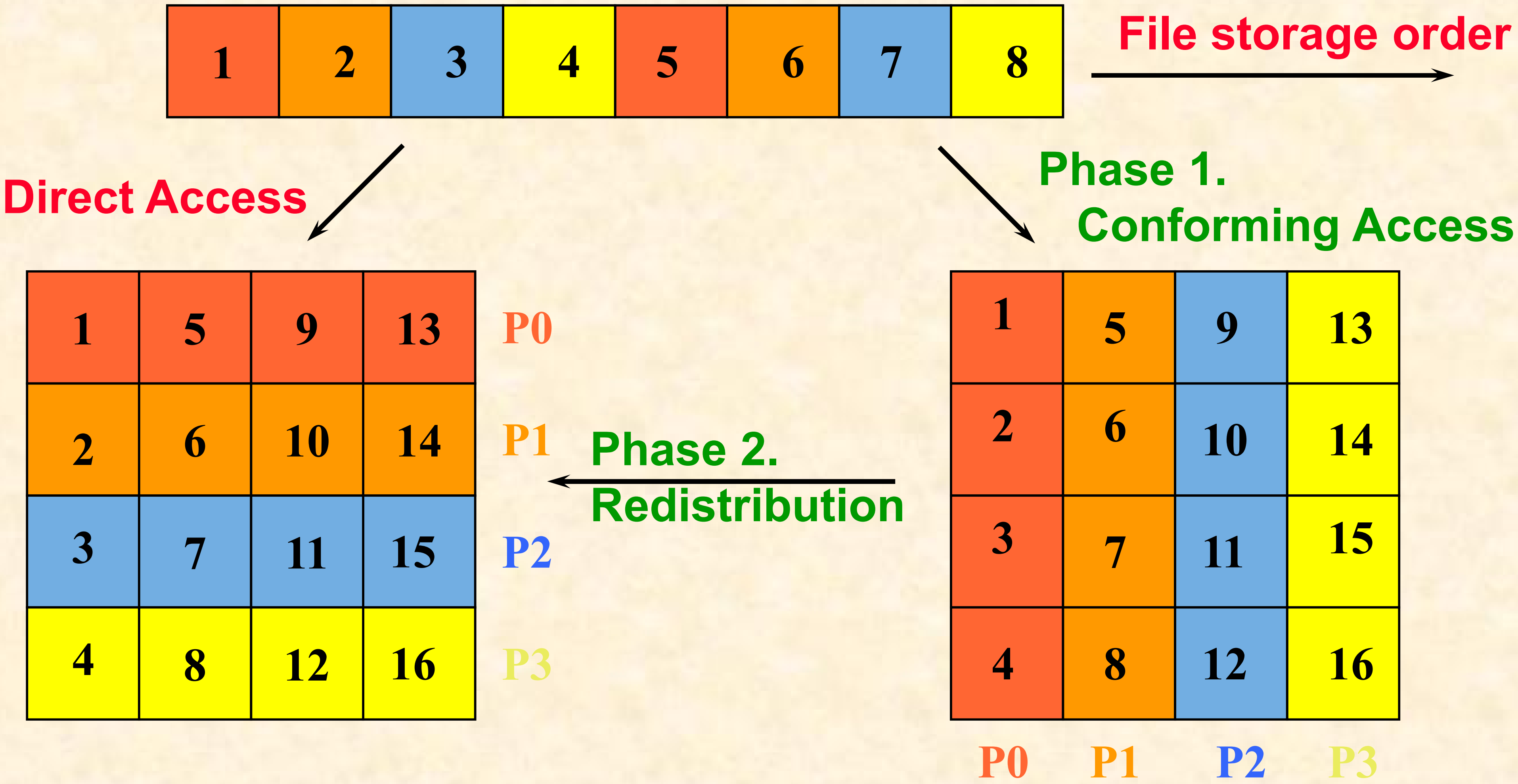
Parallel I/O : Semantics of Collective I/O

- Collective I/O
  - Generates a small number of contiguous accesses
  - Collective accesses conform with the data storage pattern
- Overall operation can be partitioned into
  - *Prologue*
    - *Global.* Involves coordination among participants
  - *Body*
    - *Local.* Independent non-overlapping accesses
  - *Epilogue* (Optional)
    - *Local.* May involve data distribution.
- Does not honor order among individual accesses

Parallel I/O : Implementations of Collective I/O
:-> Two-phase access strategy

- *Useful for accessing distributed arrays from striped files*
- Basic assumption
  - For given parameters (striping, storage order), there exists an array distribution pattern that requires the minimum I/O. This distribution is called the conforming distribution.
- Two-phased approach
  - Phase 1 : Conforming accesses
    * Access data from files according to the conforming distribution
  - Phase 2 : Redistribution
    * If required, redistribute data among processors to match application's desired distribution pattern

Parallel I/O : Directed Access vs. Two-phase Access



I/O requests in direct access = N*N
I/O requests in collective access = P

I/O for Irregular Applications : Irregular Application

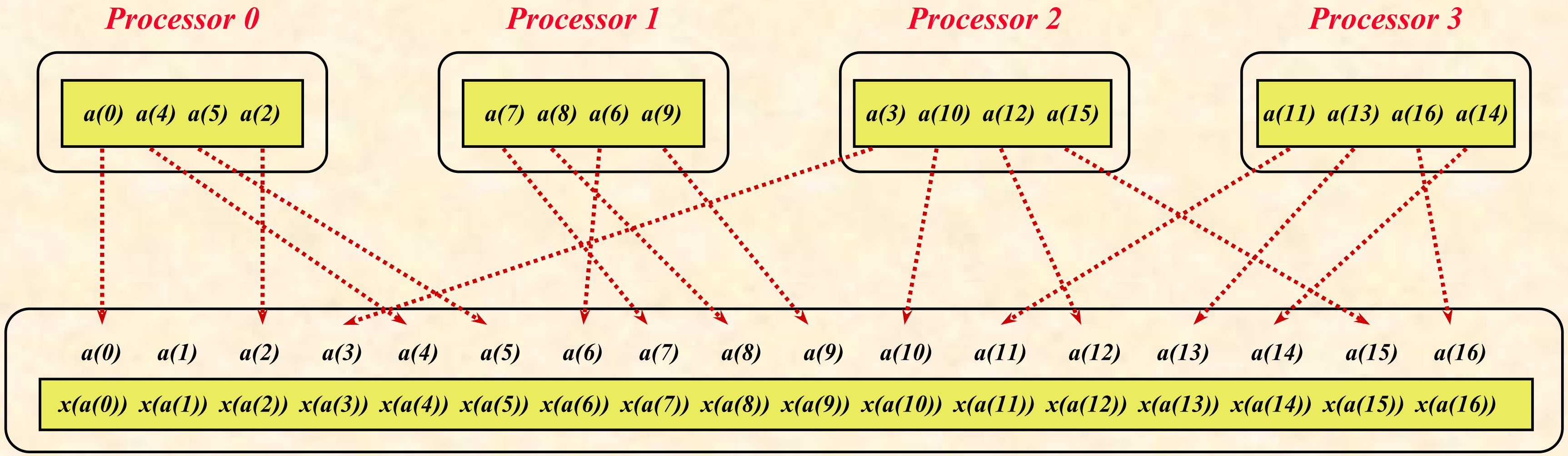
- One or more level of **indirections**
  - data domain is decided by computing its indirection values
  - e.g., sparse matrix computations,

    particle codes,

    CFD applications,…

```
do i = lb1, ub1
    x[a[i]] = F(x[a[i]], y[b[i]])
enddo
do j = lb2, ub2
    x[c[j]] = G(x[c[j]], y[d[j]])
enddo
```

- An Example

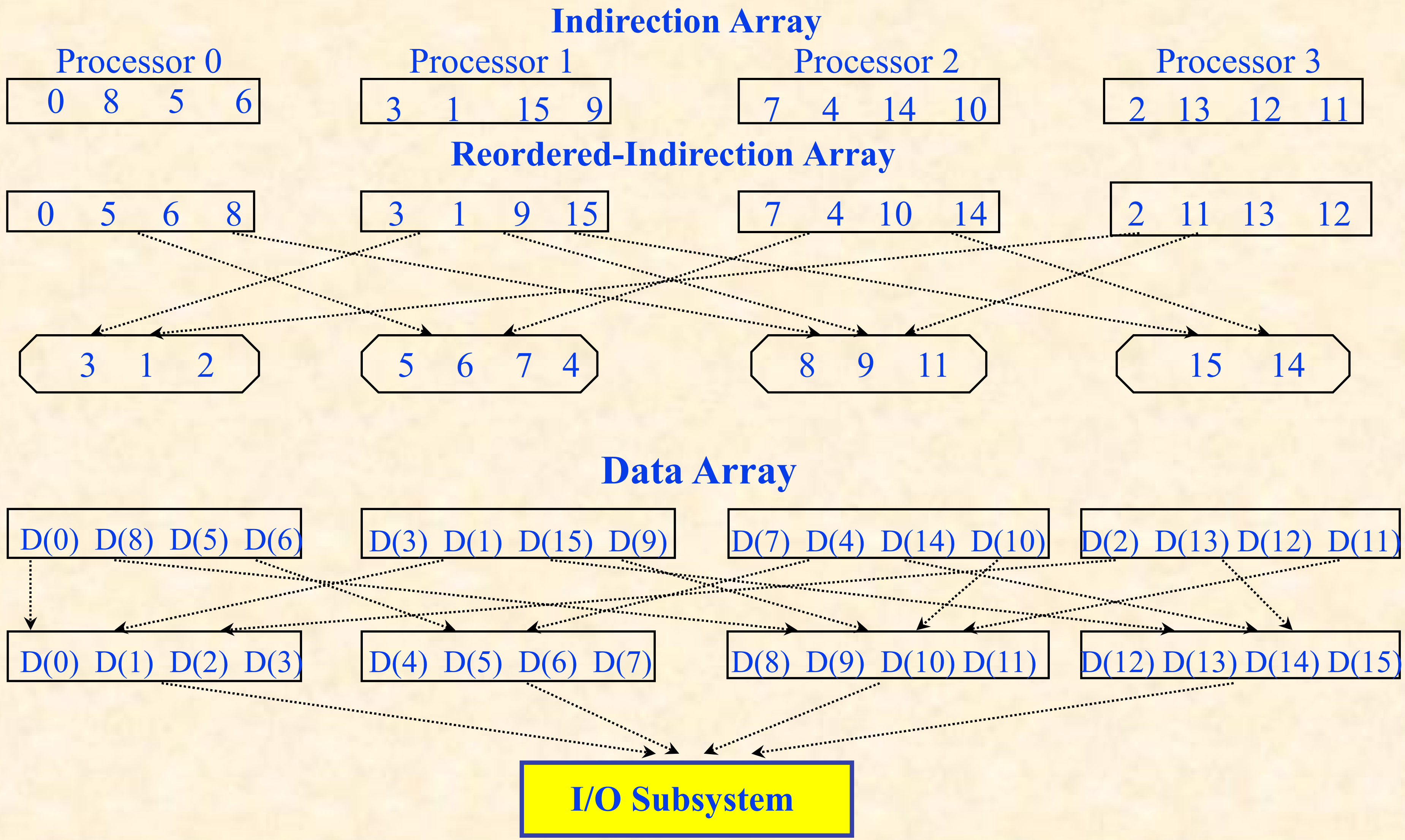


I/O for Irregular Applications : Collective I/O operations

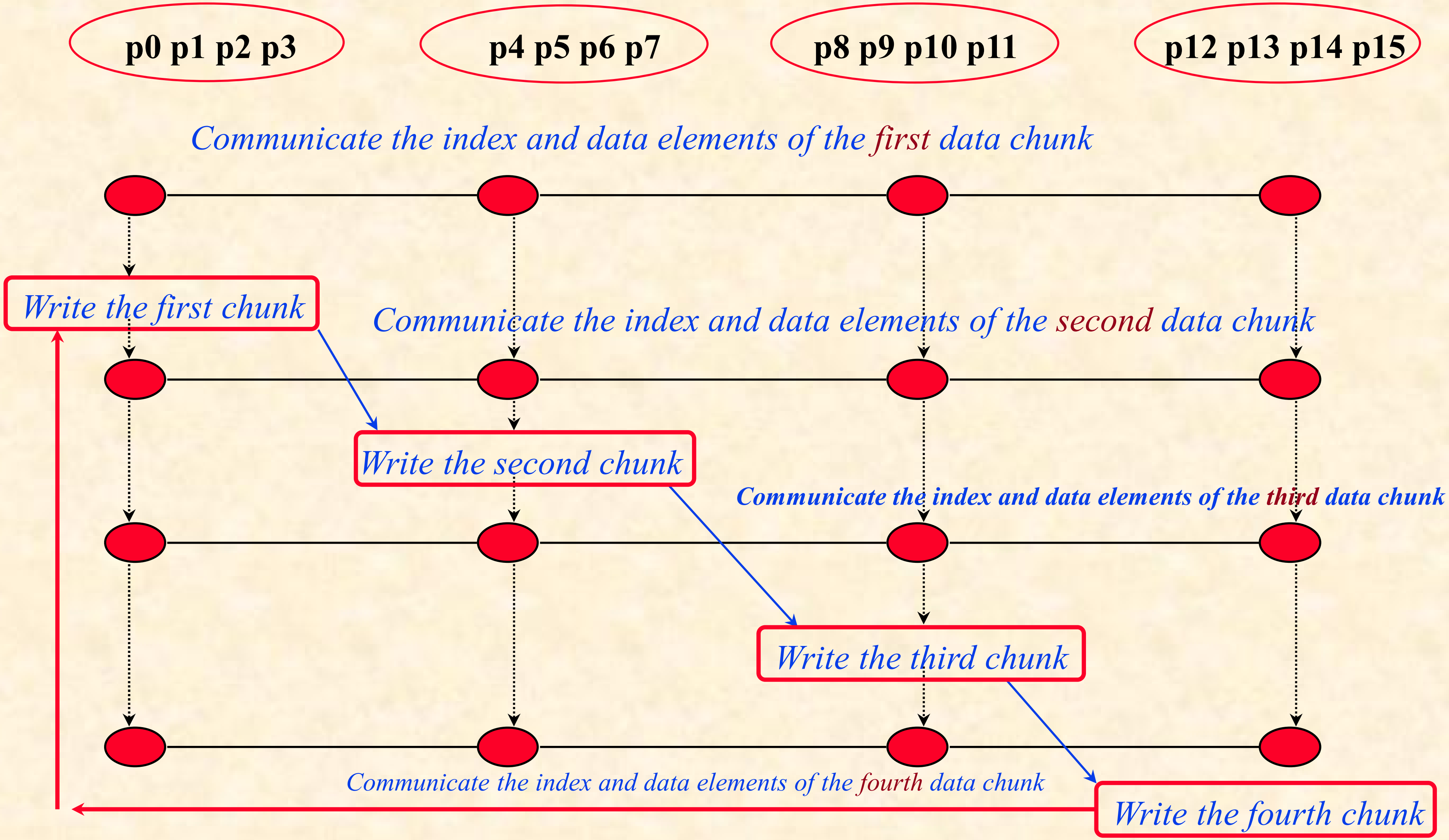
- Collective I/O
  - A processor involved in the computation is also responsible for reading data from files or writing data into files
- Pipelined Collective I/O
  - Divide processors into multiple processor groups
  - Only processors in a group issue I/O request simultaneously to reduce I/O contention
  - While one group of processors is performing I/O operation, another group performs communication in order to collect (redistribute) data from write (read)

I/O for Irregular Applications :

Design for Irregular Accesses

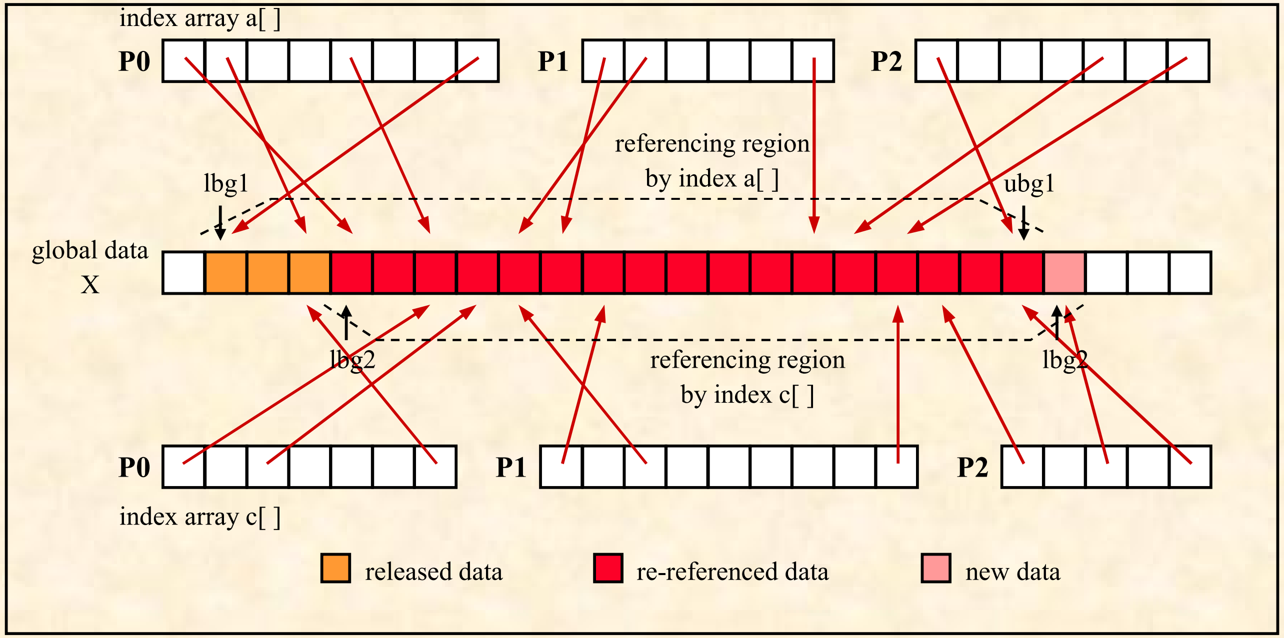


I/O for Irregular Applications :
Dynamic Contention Management



I/O for Irregular Applications: Optimizations - S/W caching

- Data Reuse
  - Irregular applications spend too much time in accessing
  - Same data may be accessed repeatedly during the execution of subsequent irregular loops
  - An example



I/O for Irregular Applications: Optimizations - S/W caching

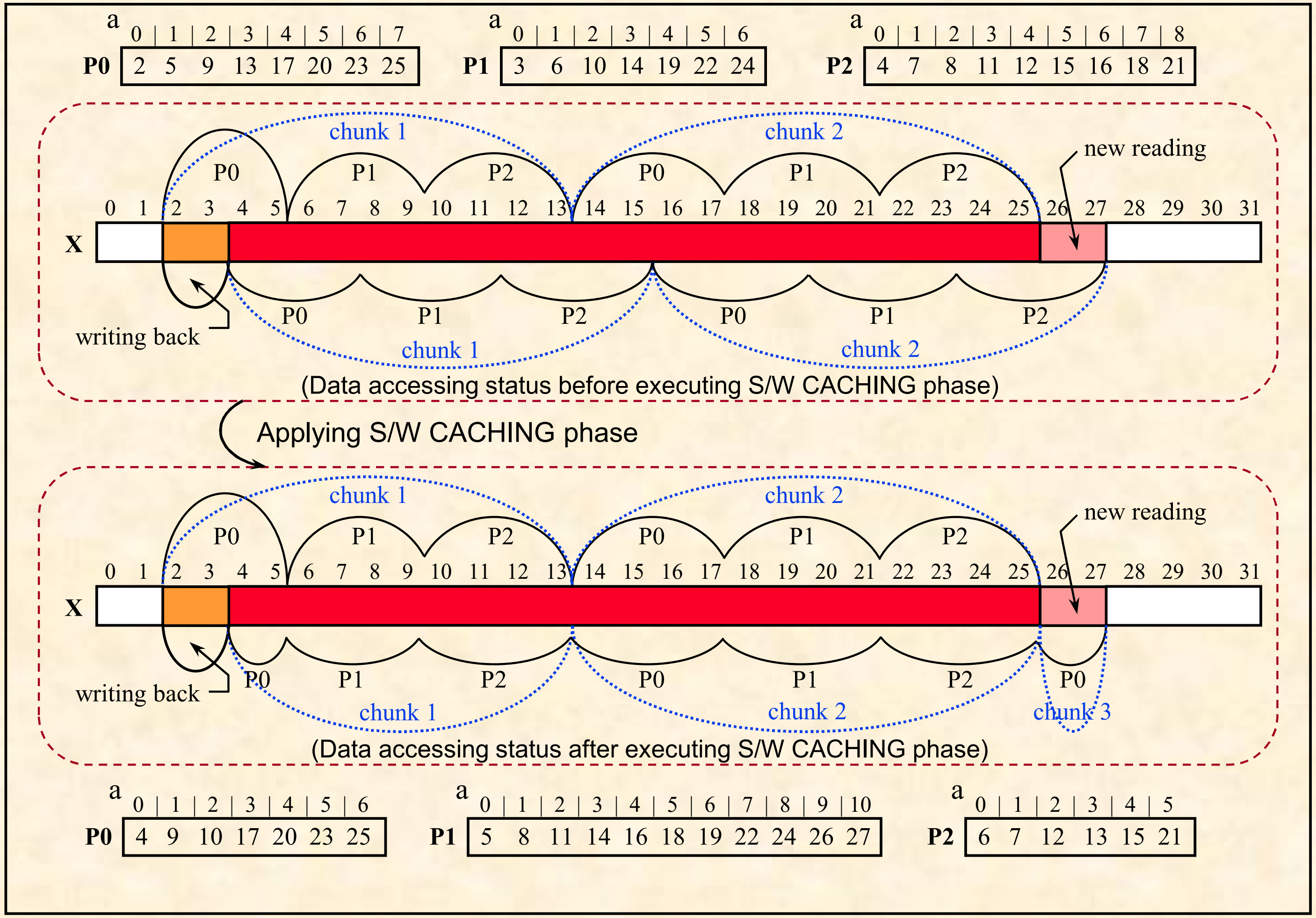
- Basic Goals
  - reduce I/O to the maximum extent possible
  - reuse the schedule information constructed in the beginning
  - build only incremental schedule
- Two steps are added to basic collective ***read/write*** operations

  (read phase case)

  - read data partially from files and redistribute it into appropriate locations of each processor
  - perform s/w caching phase to modify schedule information

I/O for Irregular Applications: Optimizations - S/W caching

- Data Access Pattern before and after Executing S/W Caching Phase



I/O for Irregular Applications:

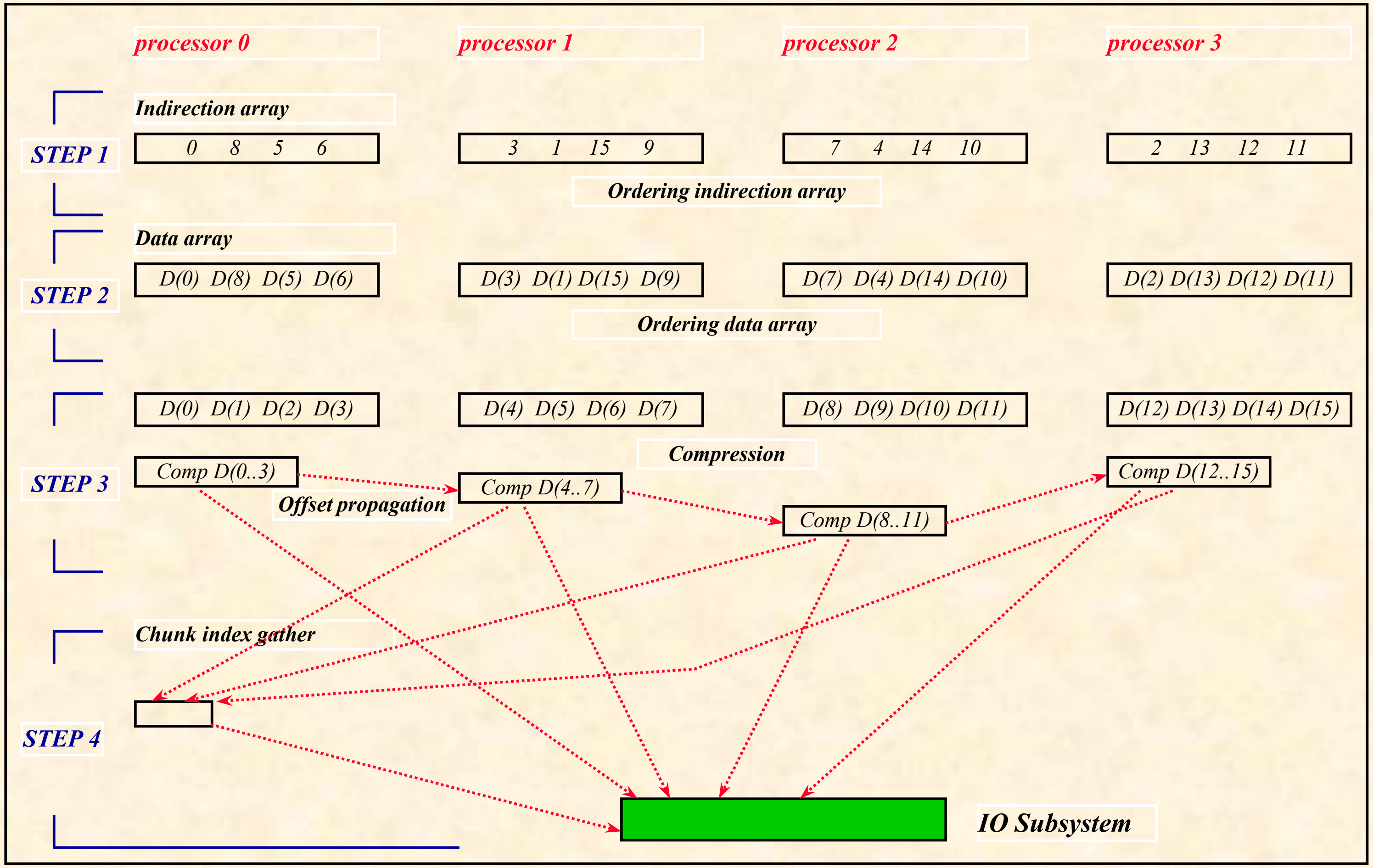
Optimizations - Compression

- Design Hints
  - (Un)compressing the I/O buffers in-memory before (reading) writing them
  - Each buffer considered as an independent ***chunk*** to manage only the subset of the array that can be stored in memory
  - Reducing execution time and storage space requirements, without loosing generality
  - Compression algorithms tradeoff :- good compression ratio usually means high compression time
  - We looked for an algorithm with a reasonable compression ration and execution time : *lzwr3*

I/O for Irregular Applications:

Optimizations - Compression

- Steps in Collective Write with Compression



I/O for Irregular Applications : Performance Evaluation

- Performance Parameters

  - Total data size : *256KB - 64MB*
  - Number of Processors : *32 - 128 computing nodes*
  - Chunk size, which represents the amount of buffer space available to the runtime library
  - Stripe Unit, representing the amount of logically contiguous data read/written from each I/O node
  - Number of processor groups (for the pipelined collective I/O)
  - Overlap range (for software caching), which is the data area overlapped between two irregular loops
  - Compression ratio (for compression)

I/O for Irregular Applications : Performance Evaluation

- System Environment

  - Intel Paragon at Caltech (TREX)

    > 16 and 64 I/O nodes partition, each I/O node having a 4GB Seagate disk

    > PFS file system used

  - ASCI/Red Teraflops (Sandia National Lab.)

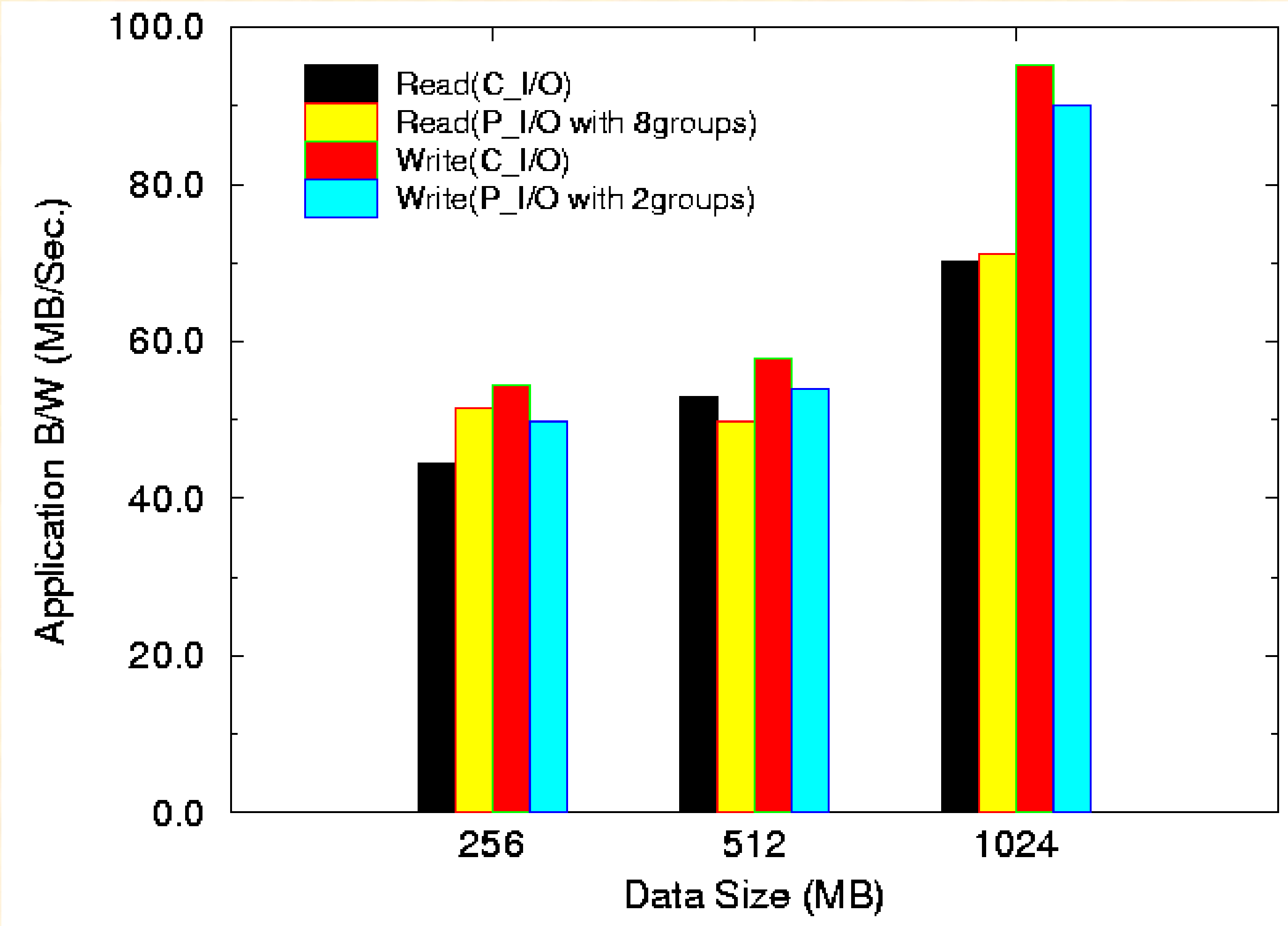
    > 9 I/O nodes partition, each I/O node having two 32GB RAID

    > PFS file system used

    > Default stripe unit is 512KB

I/O for Irregular Applications : Performance Evaluation

- Application B/W for Collective and Pipelined Collective I/O as a function of data size on ASCI/Red

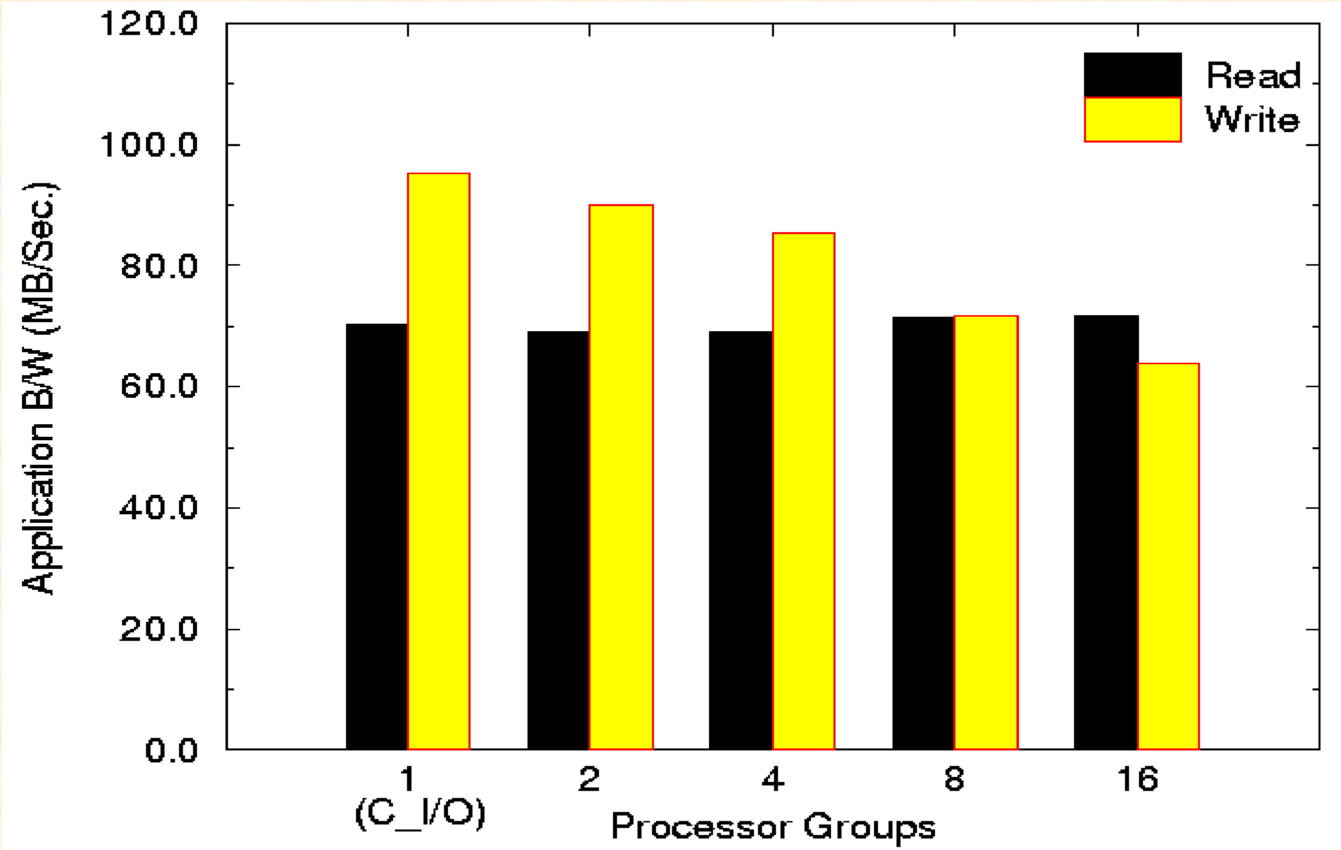


- The # of compute node 64(DS=256MB) and 128(DS=512,1024MB)

- Stripe unit is 512KB

I/O for Irregular Applications : Performance Evaluation

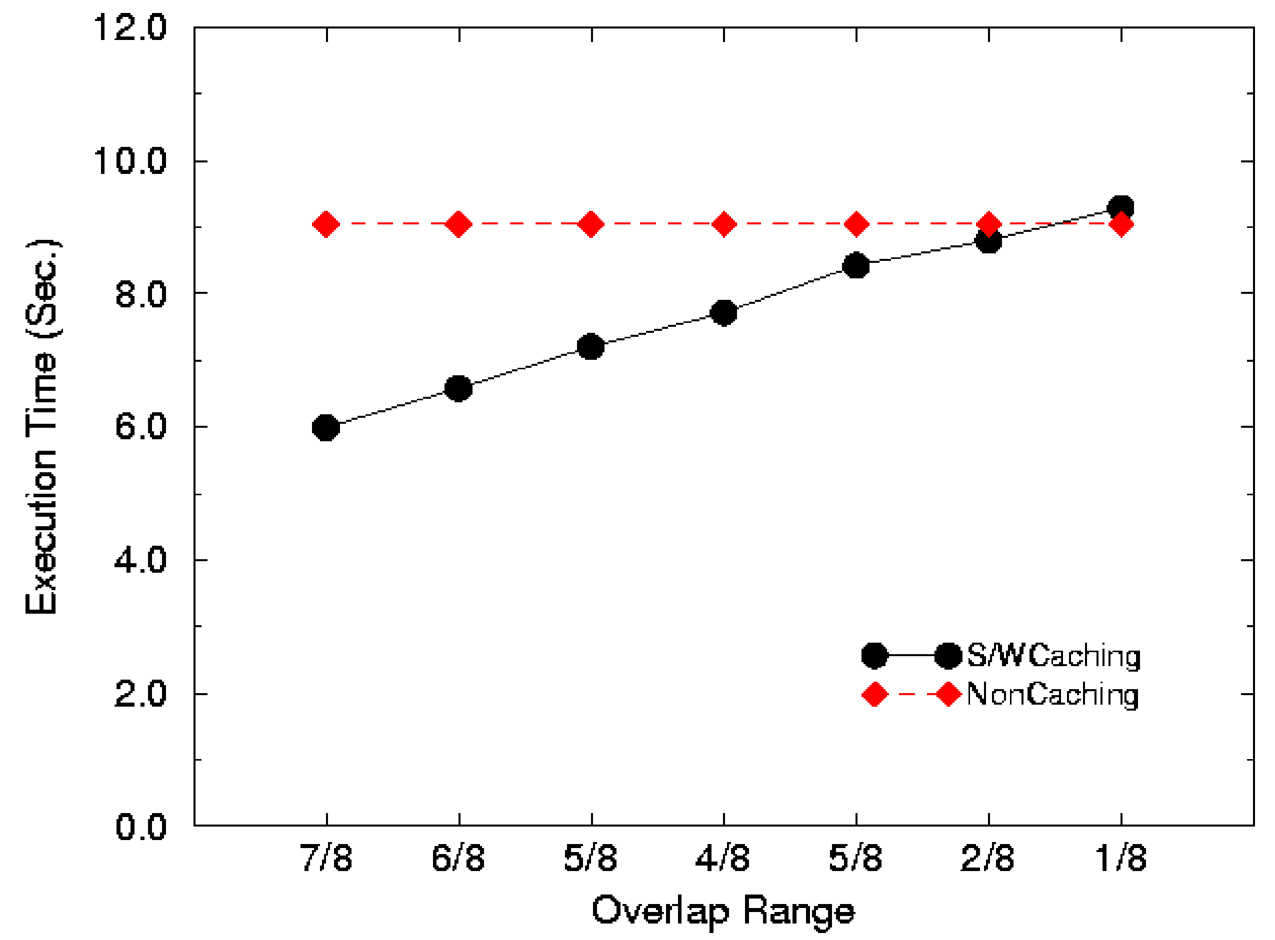
- Application B/W for Collective and Pipelined Collective I/O as a function of processor groups on ASCI/Red



- The # of compute nodes is 128
- The total data size is 1024MB
- Stripe unit is 512KB

I/O for Irregular Applications : Performance Evaluation

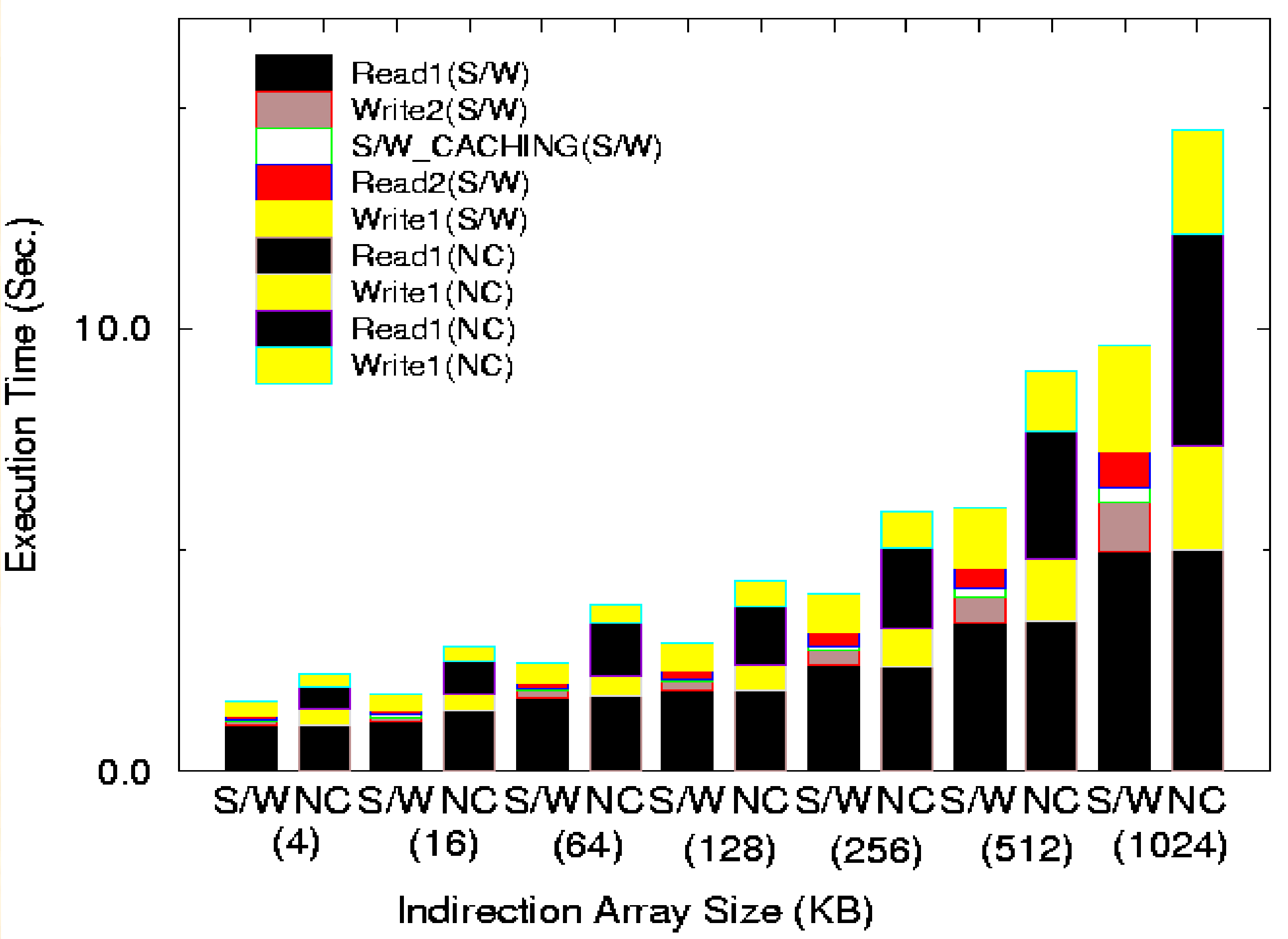
- Execution Time for S/W Caching Method compared with for Non-caching Method on TREX



- 16 I/O nodes partition
- The # of compute nodes is 32
- The total data size is 128MB
- Stripe unit is 64KB

I/O for Irregular Applications : Performance Evaluation

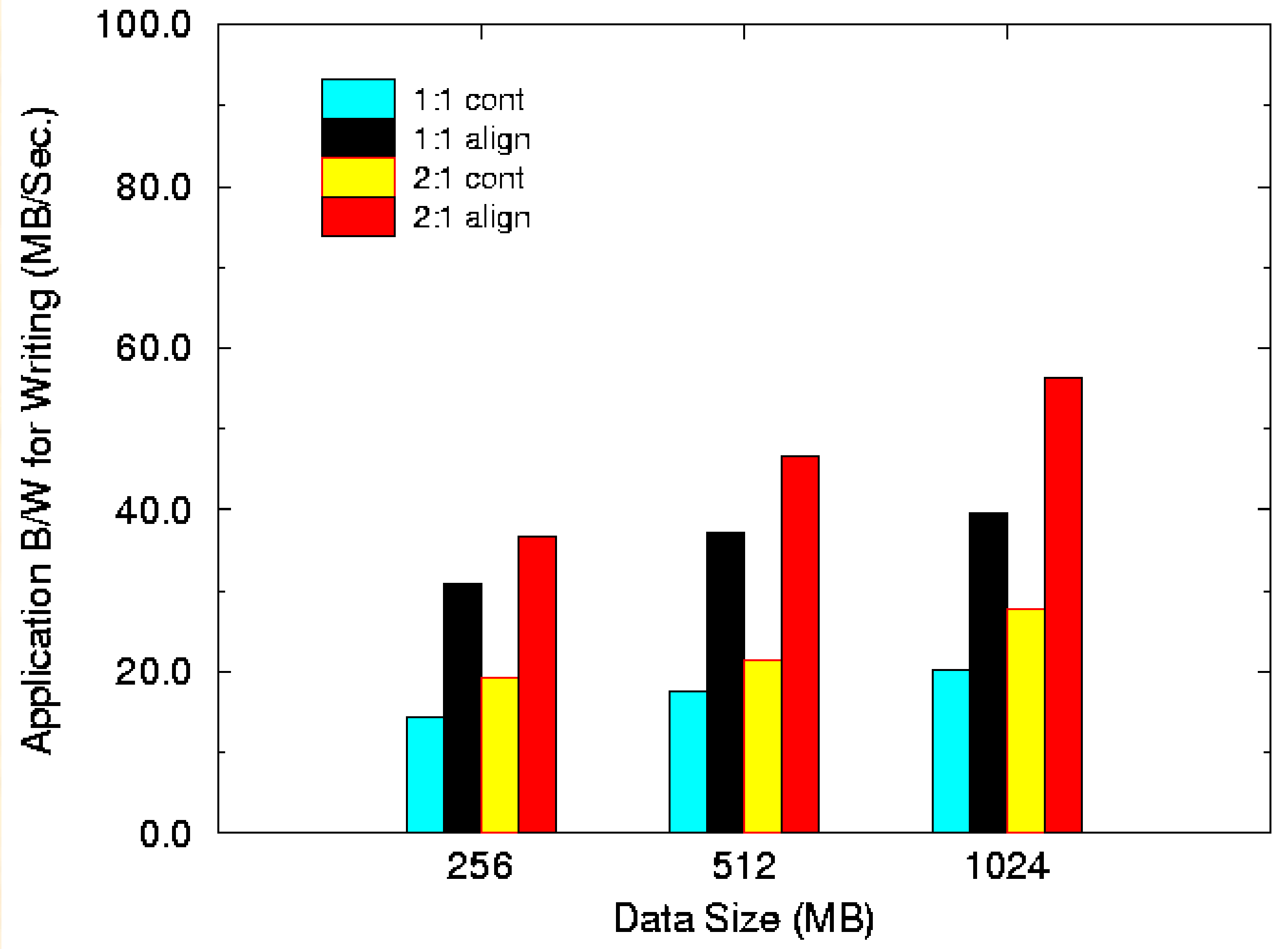
- Execution Time for S/W Caching Method compared with for Non-caching Method on TREX



- 16 I/O nodes partition
- The # of compute nodes is 32
- The total data size is 128MB
- Stripe unit is 64KB

I/O for Irregular Applications : Performance Evaluation

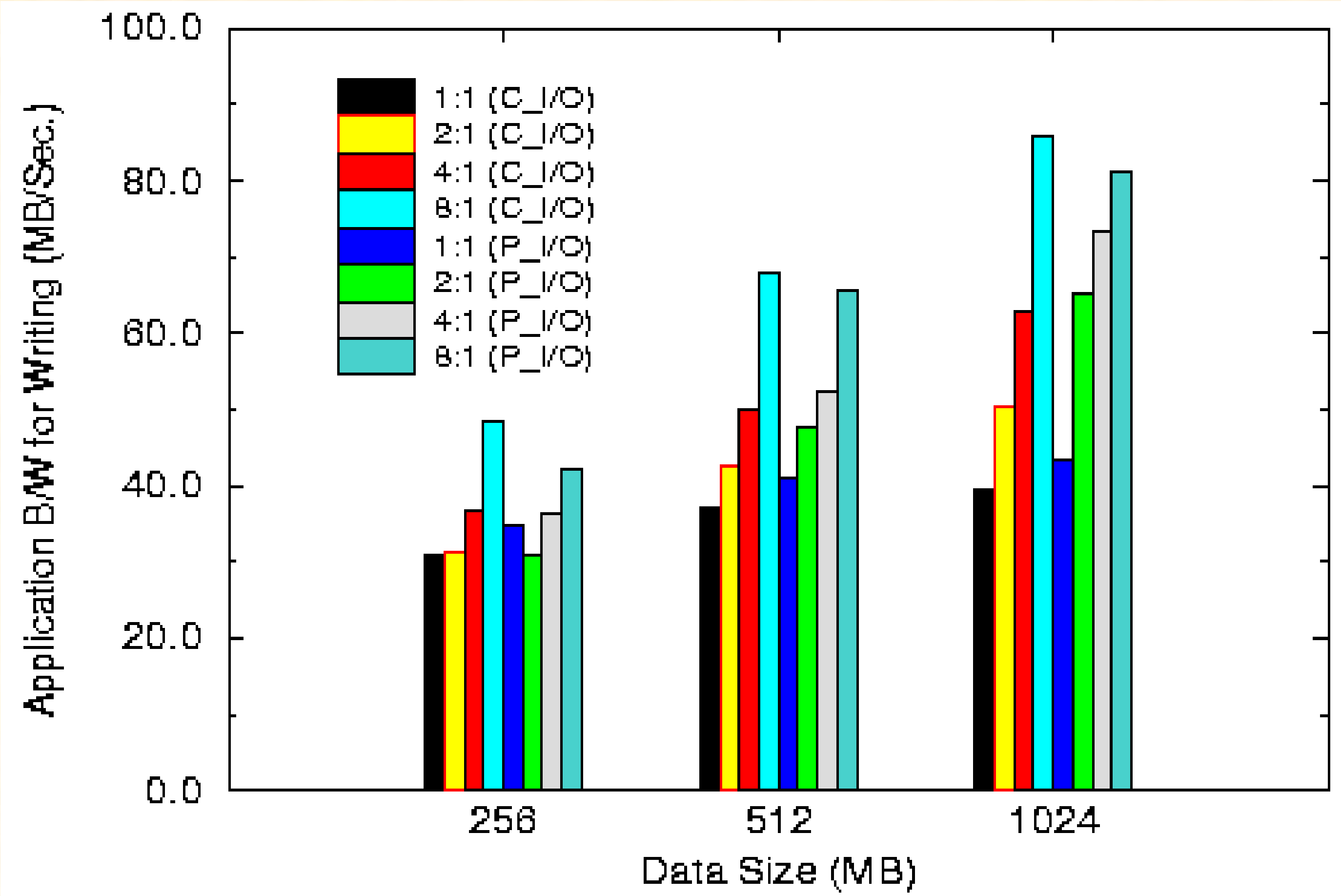
- Block Alignment Influence on Collective I/O on TREX



- 16 I/O nodes partition
- The # of compute nodes is 64(DS=256MB), 128(DS=512MB), and 256(1024MB)
- Stripe unit is 64KB

I/O for Irregular Applications : Performance Evaluation

- Application B/W for Collective and Pipelined Collective Write as a function of data size and compression ratio on TREX



- 16 I/O nodes partition
- The # of compute nodes is 64(DS=256MB), 128(DS=512MB), and 256(1024MB)
- Stripe unit is 64KB
- Compression ratios : 1:1, 2:1, 4:1, and 8:1

Future Trends of I/O in Software

- Algorithms
  - Better models for validating parallel I/O accesses
  - Expanding the catalog of out-of-core algorithms
  - More High level implementations
- OS and File Systems
  - Standard for parallel file system interfaces (e.g., POSIX)
  - Improved optimizations (caching, prefetching, replication)
  - Unified file systems for hierarchical storage
  - Unified file systems for emerging global computer networks
- Compiler and Runtime Systems
  - Improved optimizations for out-of-core problems
  - Better interactions with the OS/file systems

Further Information

- Refer the Parallel I/O bibliography

  http://www.cs.dartmouth.edu/pario/bib
- Technical Papers
  - IOPADS conference proceedings
  - IEEE Computer March 1994 Special Issue
  - *"I/O in Parallel…"*, Jain, Browne, Wirth, Editors.
  - Gibson et al., ACM Computing Surveys, December 1996
- URLs
  - Parallel I/O Archives
    - http://www.cs.dartmouth.edu/pario
  - Scalable I/O Initiative
    - http://www.cacr.caltech.edu/SIO