View I/O: improving the performance of non-contiguous I/O

Florin Isaila

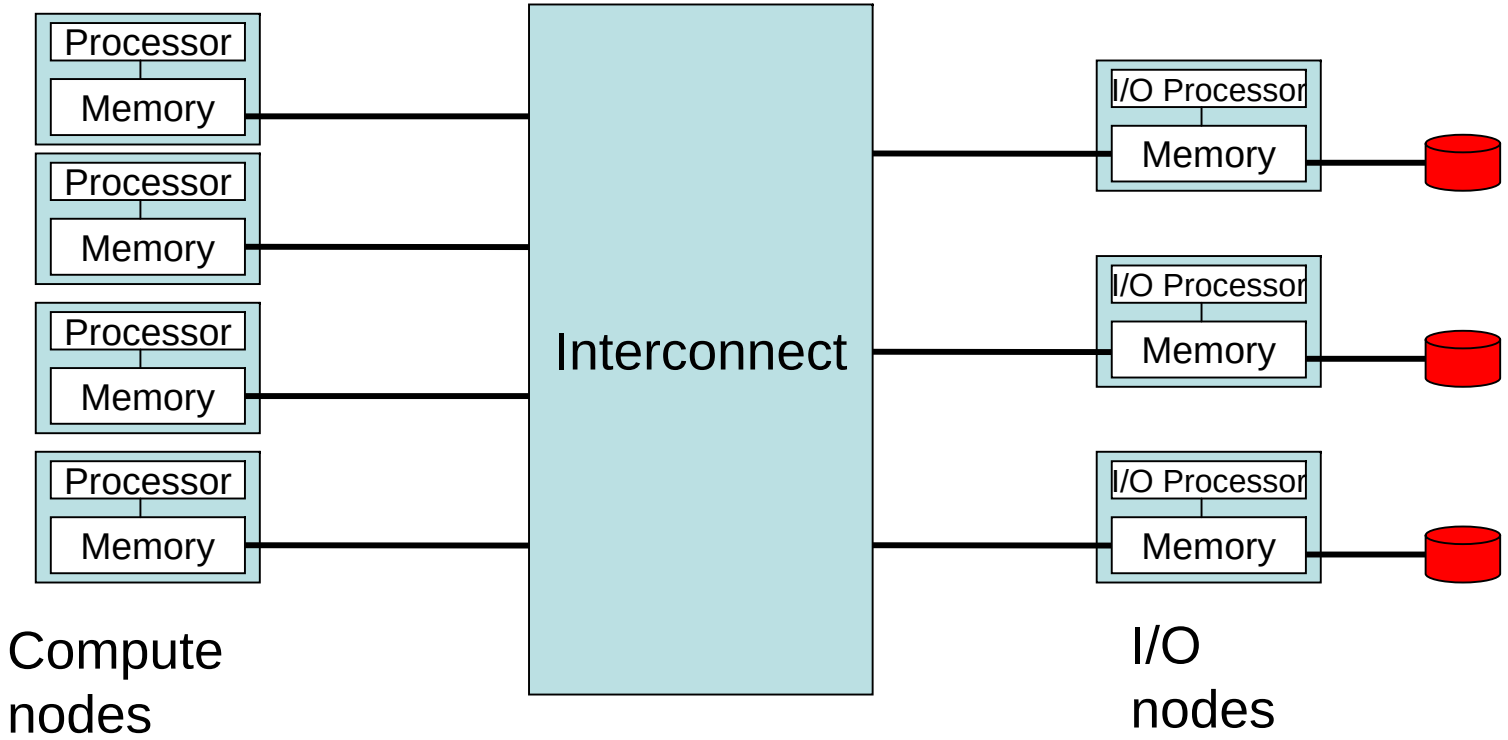
Walter F. Tichy

Overview

- Motivation
- Non-contiguous I/O methods
- View I/O
- Experimental results
- Conclusion

Parallel File System Model

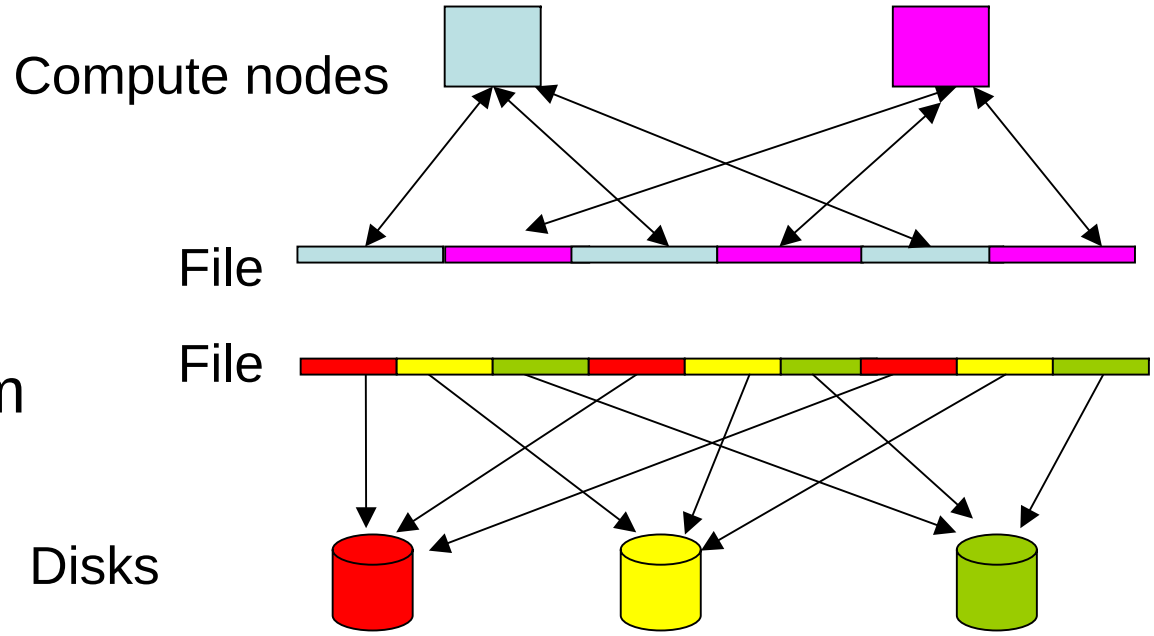
Used parallel file systems: PVFS and Clusterfile



Motivation

- Parallel I/O workload characterization studies: non-contiguous access frequent
- One main reason: different types of parallelism do not match

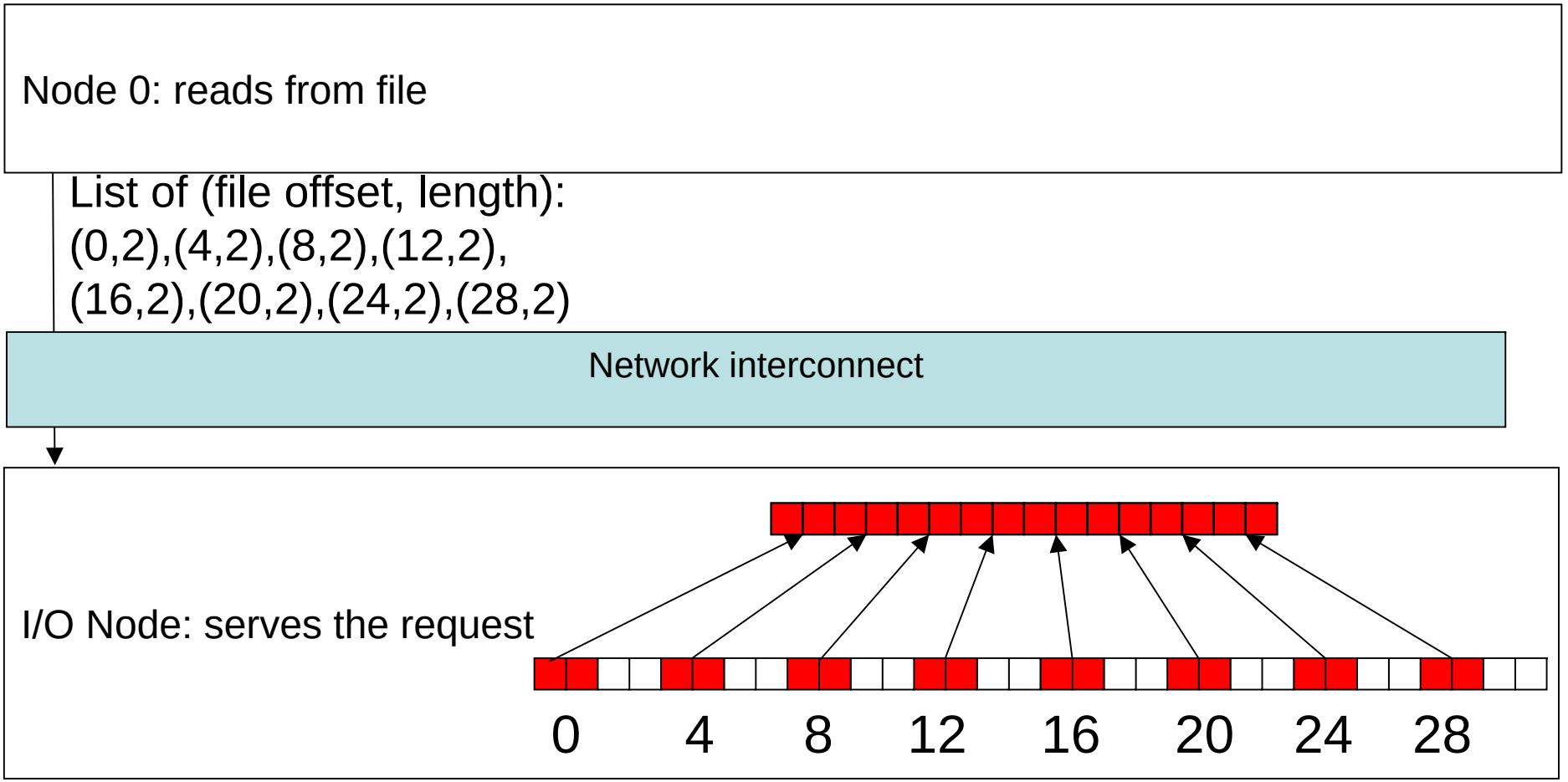
- Logical parallelism
- Physical parallelism



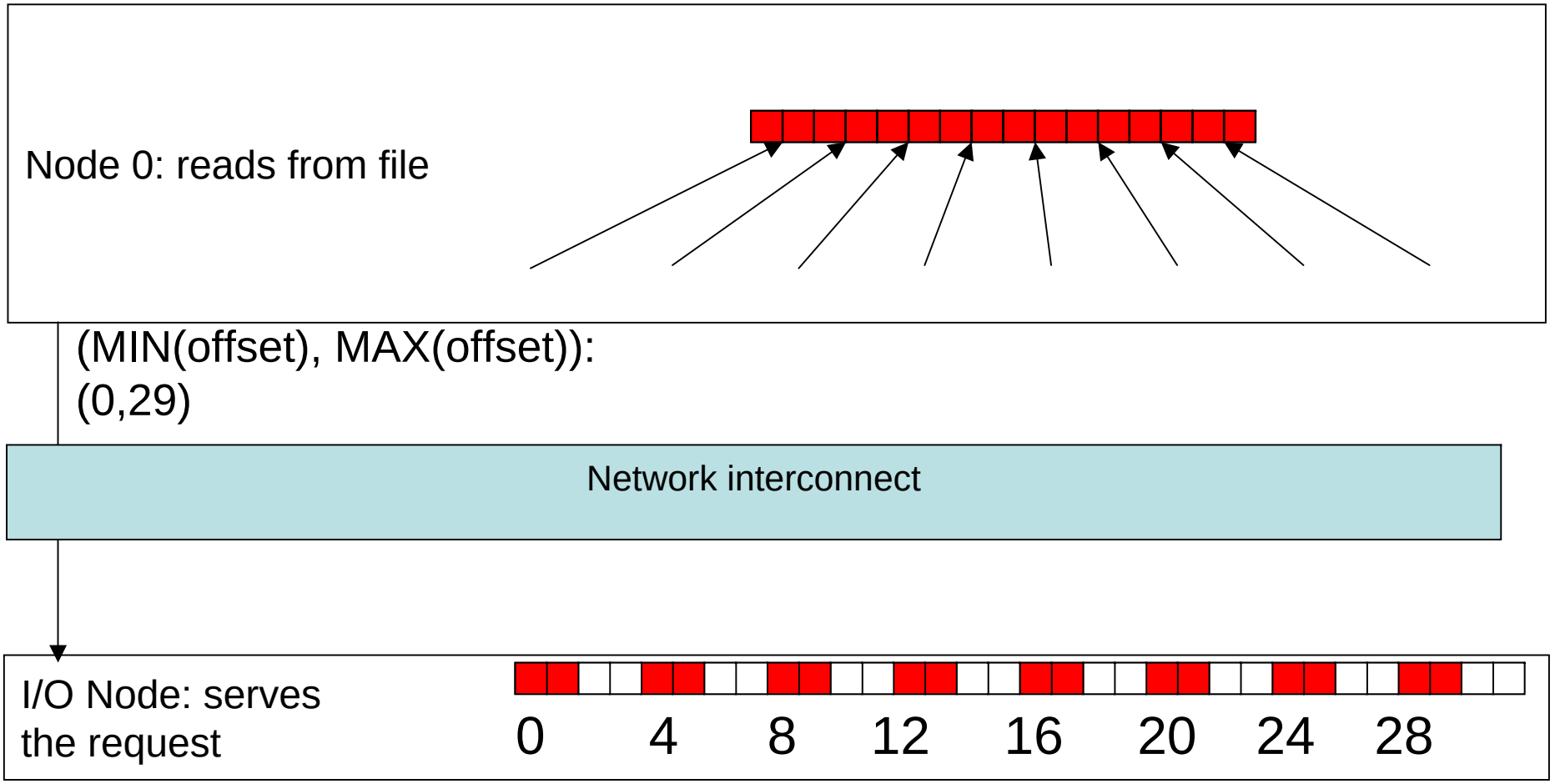
Motivation

- Efficient non-contiguous I/O
- How does the file distribution influence performance?
- Is the linear file model always suitable?

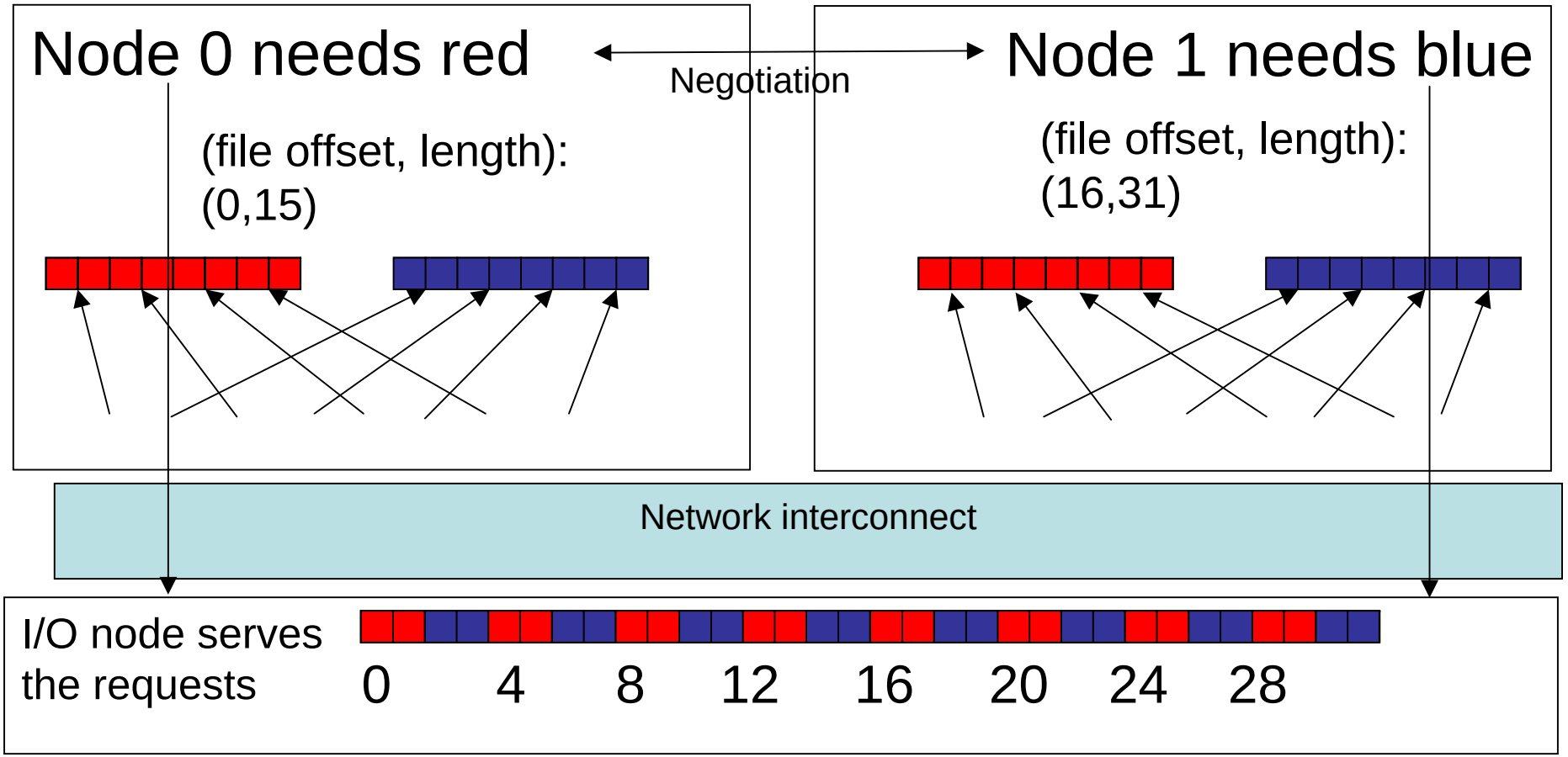
Method1 : List I/O(PVFS): read ex.



Method 2: Data sieving: read ex.

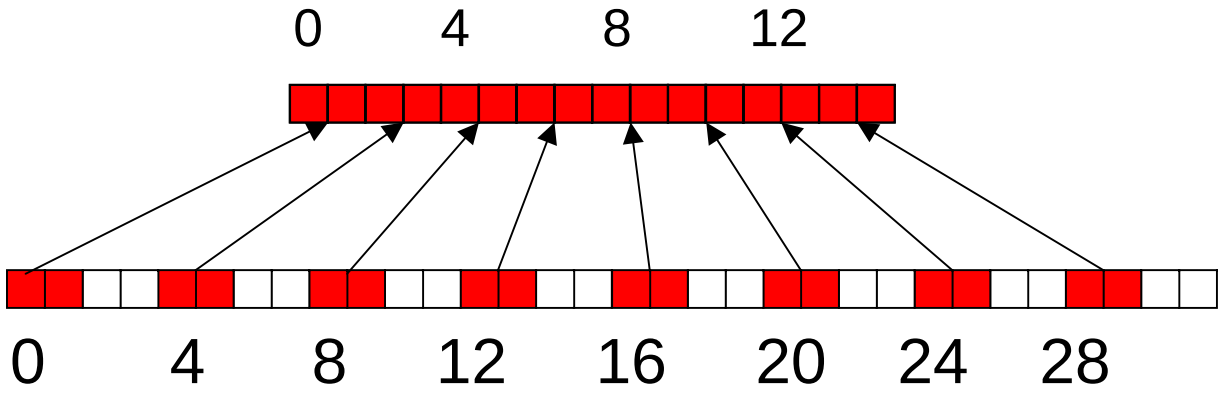


Method 3: Two-phase I/O: read ex.



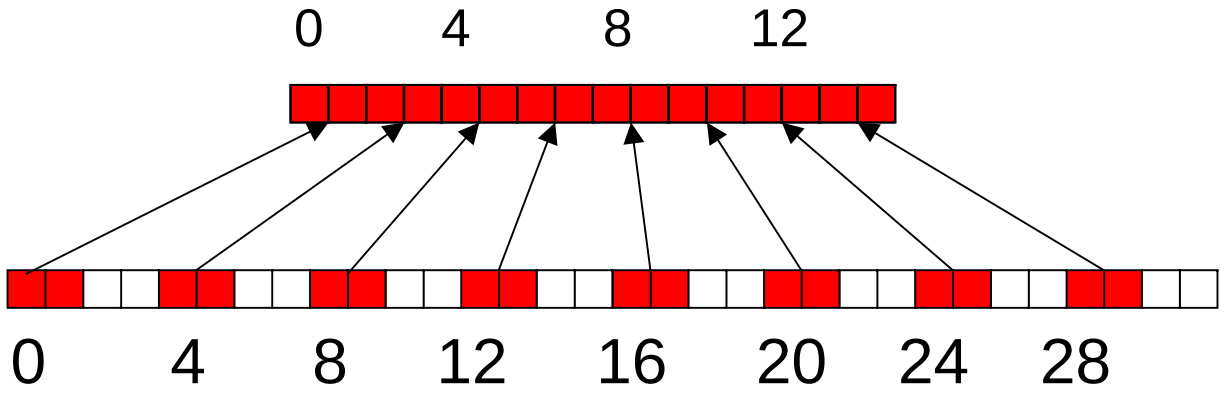
Our method: View I/O

- View : logical windows to subsets of a file
  - Advantages: accessing non-contiguous regions of a file with a single call, simplified offset computation
- Mapping view-file :
  - MPI-IO: kept at compute node
  - Clusterfile: kept at I/O node

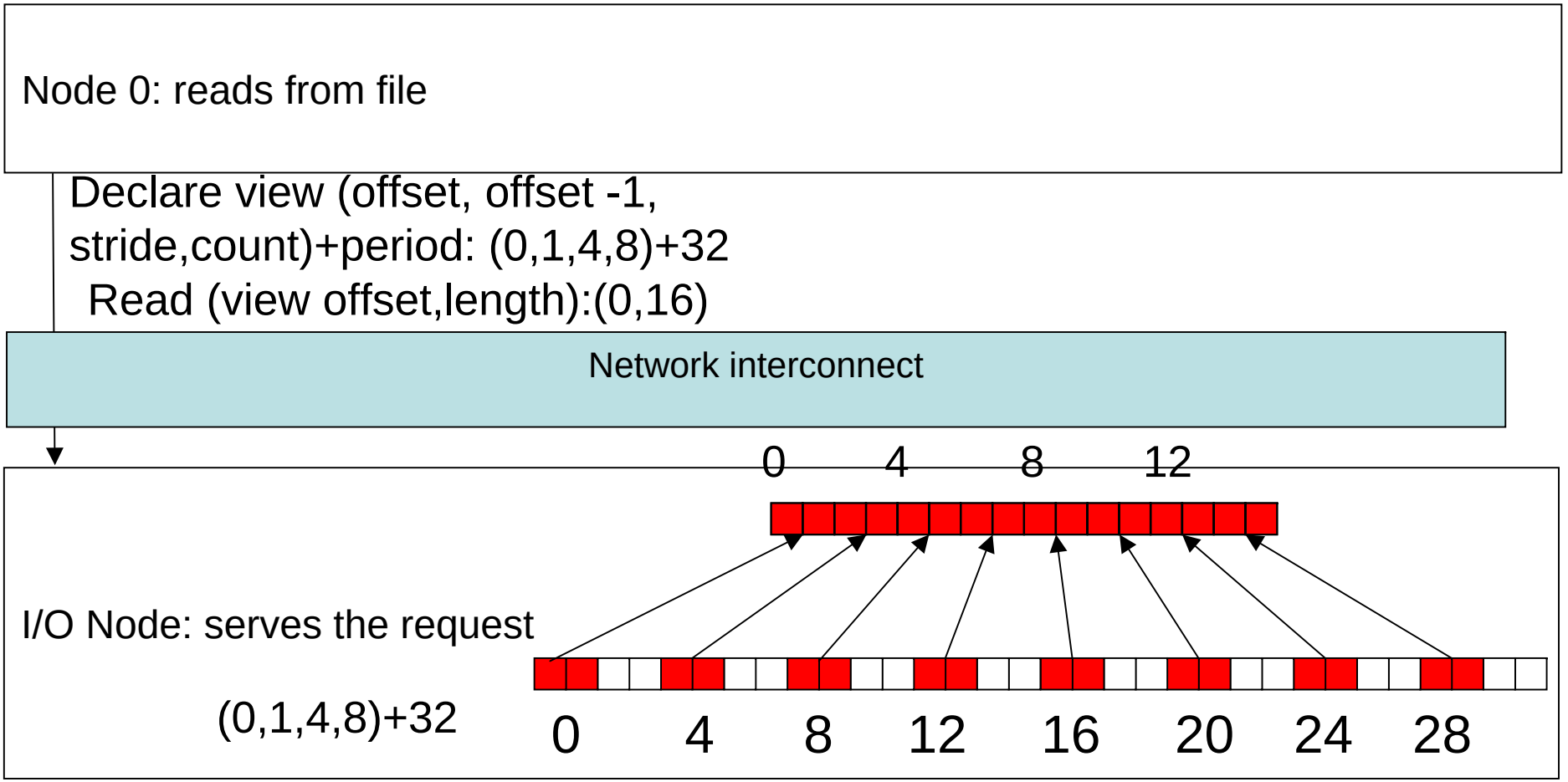


View I/O: access indices

List I/O	View I/O
List of (file offset,length): (0,2),(4,2),(8,2),(12,2),(16,2), (20,2), (24,2),(28,2)	Compact regular patterns: (offset, offset+length-1,stride,times)= (0,1,4,8)
	Use repetitive patterns: period=32: the compacted pattern is repeated
Access indices sent to I/O node at each access	Access indices sent to I/O node at view declaration



View I/O: read example



Comparison

	Data sieving	Two phase	List I/O	View I/O

Comparison

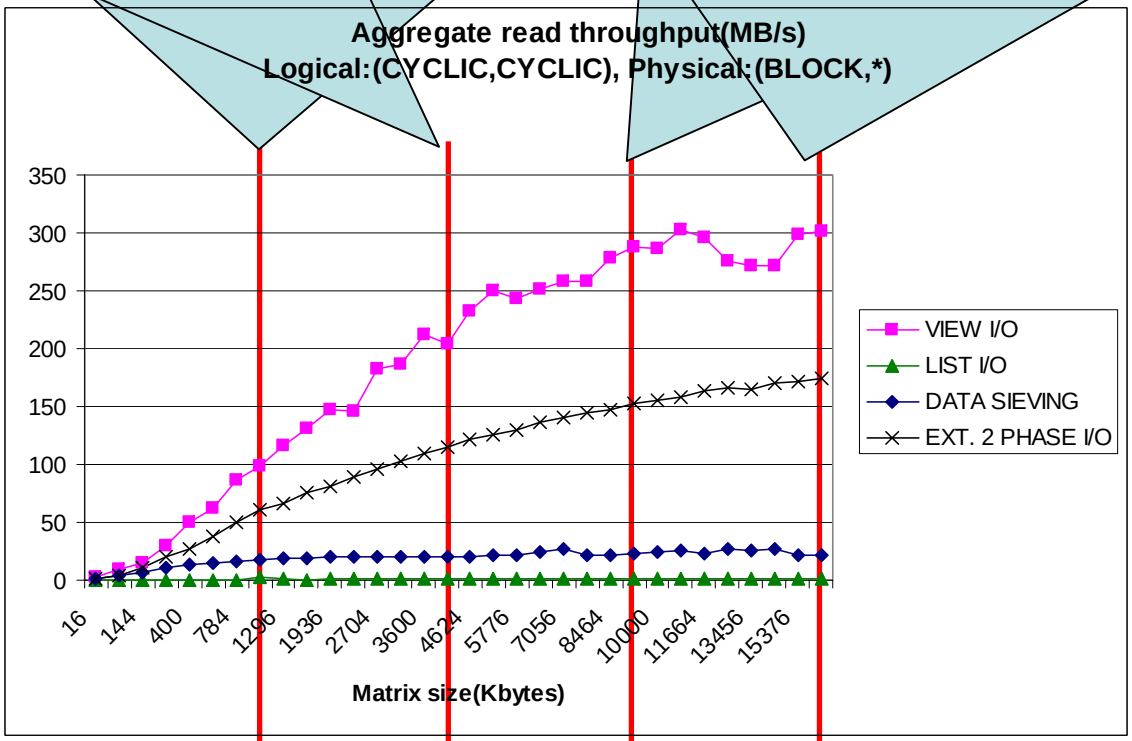
	Data sieving	Two phase	List I/O	View I/O

Experimental setup

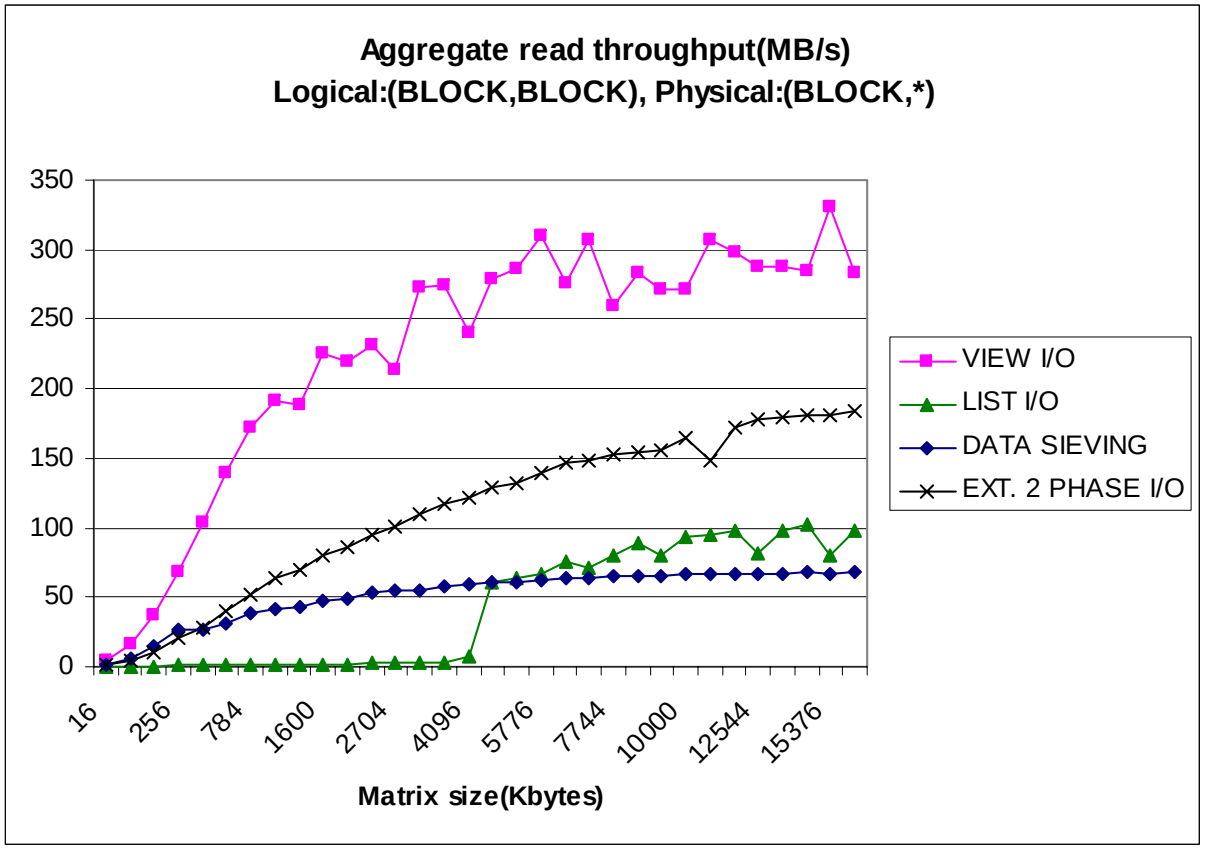
- 16 Pentium III 800MHz, 512 MB RAM
- IDE disks
- Myrinet LANai 9
- 3 matrix distributions: (BLOCK,BLOCK), (*,BLOCK), (CYCLIC(k),CYCLIC(k)

- Each processor
  - declares a view
  - writes/reads its corresponding matrix part
- 16 compute nodes + 16 I/O nodes

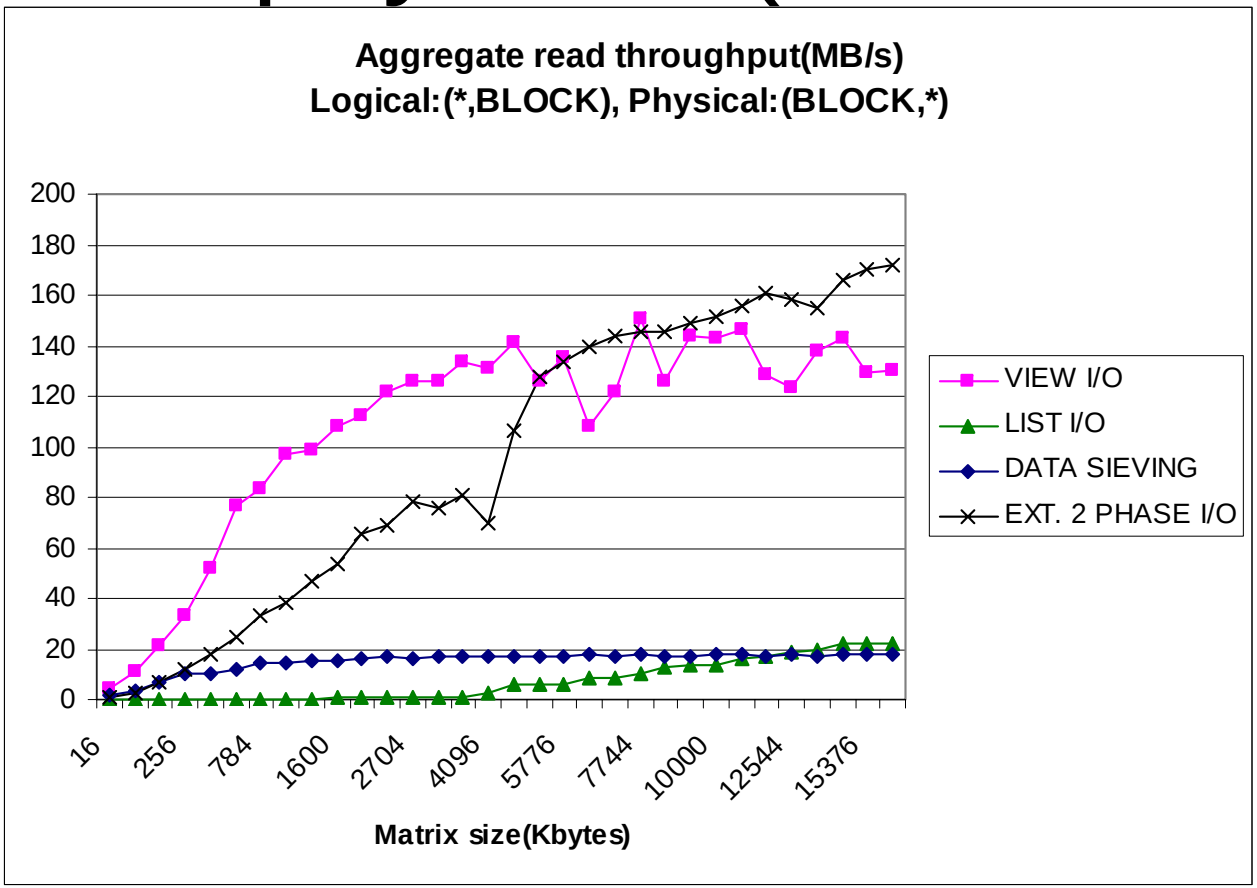
	Data sieving	Extended two-phase	List I/O	View I/O
FS calls	16	16	5968	16
Offset (B)	128	128	786432	128
More data (MB)	144	0	0	0



Views: (BLOCK,BLOCK) File physical :(BLOCK,*)

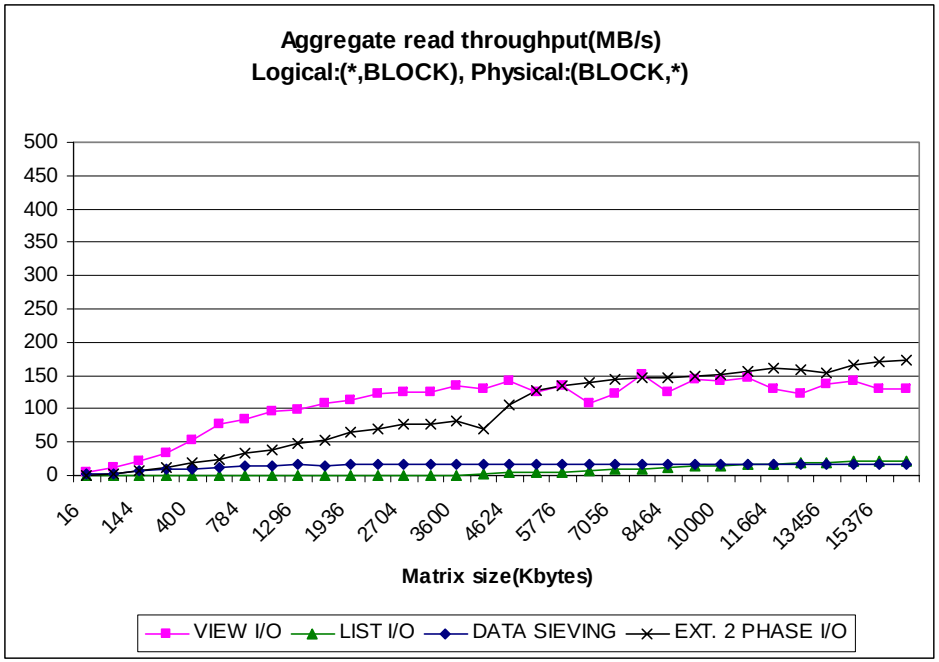


Views: (*,BLOCK)
File physical :(BLOCK,*)

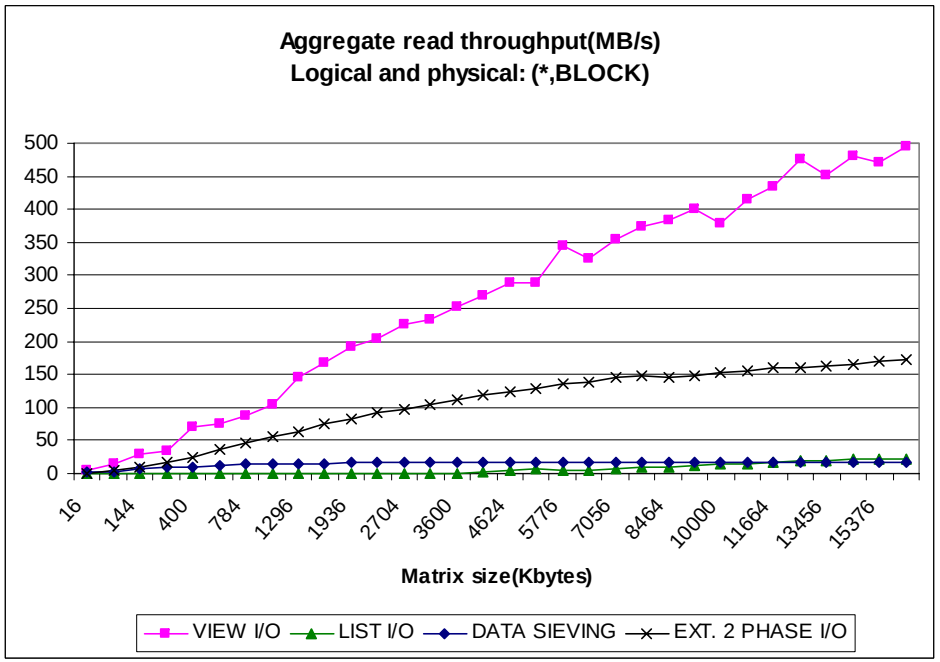


READ: Views: (*,BLOCK)

File Physical: (BLOCK,*)

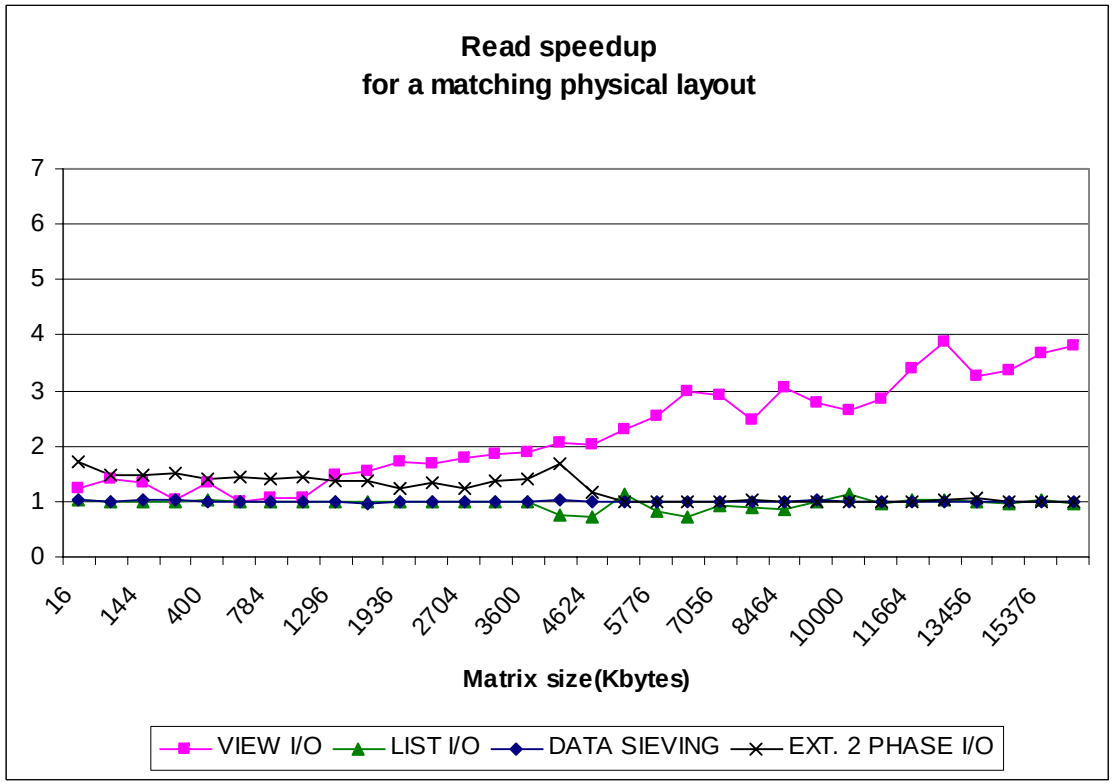


File Physical: (*,BLOCK)



Read Speedup

$$\text{Speedup} = \frac{\text{Aggregate throughput for (*,BLOCK) view (*,BLOCK) file}}{\text{Aggregate throughput for (*,BLOCK) view (BLOCK,*) file}}$$



Conclusions

- Linear file model may be unsuitable for non-contiguous I/O
- View I/O: parallel file model for both views and physical data placement
- View overhead amortization over several accesses
- Compact access indices
- Simple access syntax