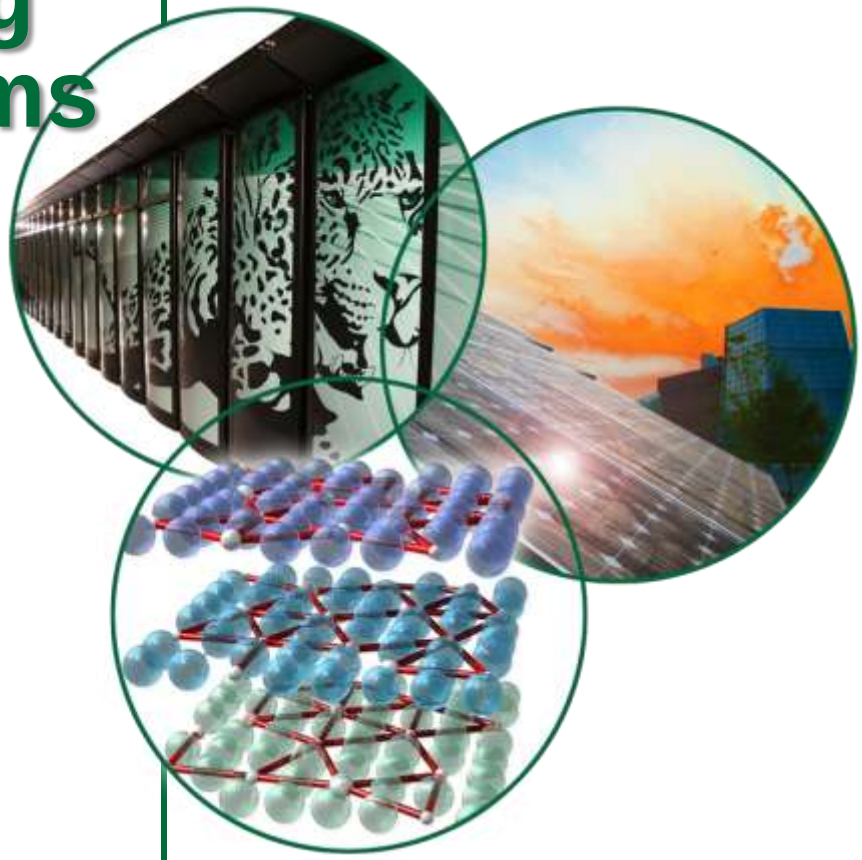
Collective Prefetching for Parallel I/O Systems

Yong Chen and Philip C. Roth

Oak Ridge National Laboratory



Future Technologies Group ornl

U.S. DEPARTMENT OF ENERGY

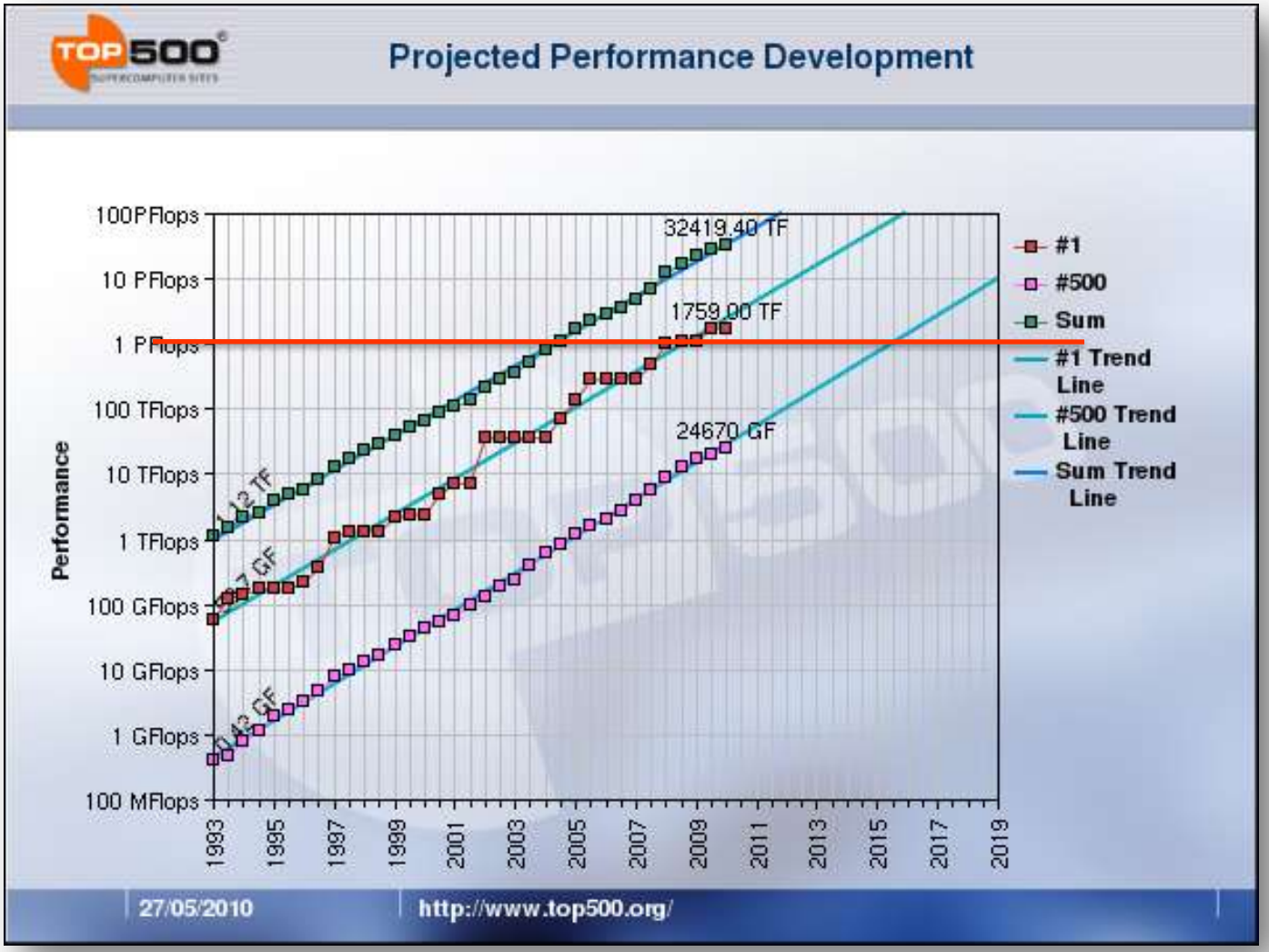
OAK RIDGE NATIONAL LABORATORY

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Outline

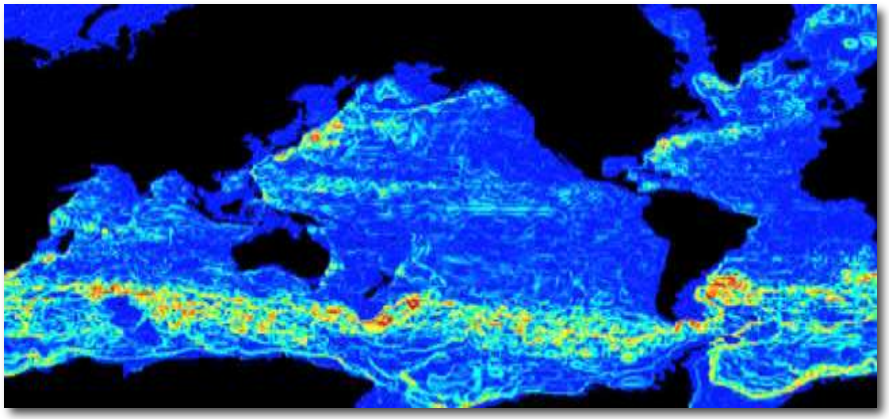
- I/O gap in high-performance computing
- I/O prefetching and limitation
- Collective prefetching design and implementation
- Preliminary experimental evaluation
- Conclusion and future work

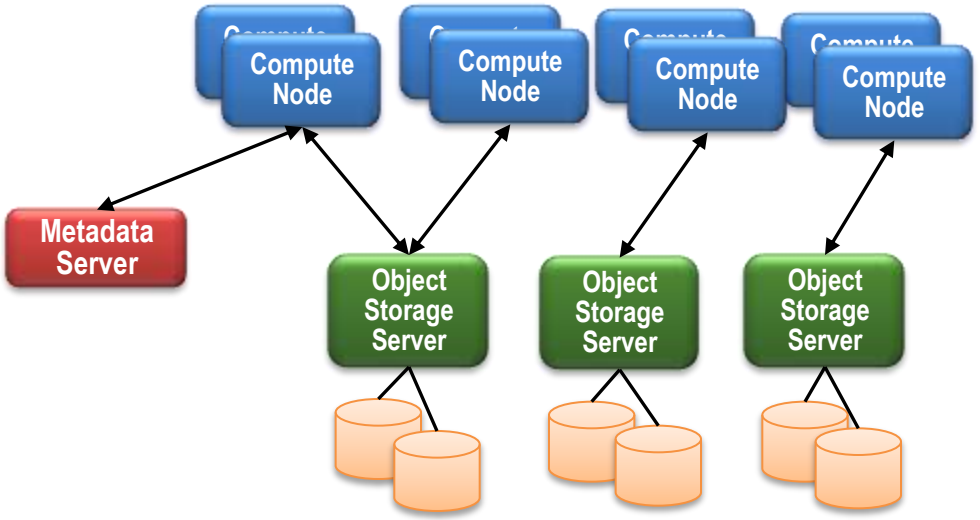
High-Performance Computing Trend

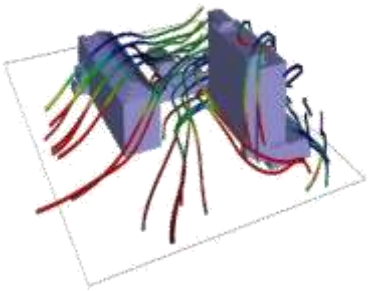


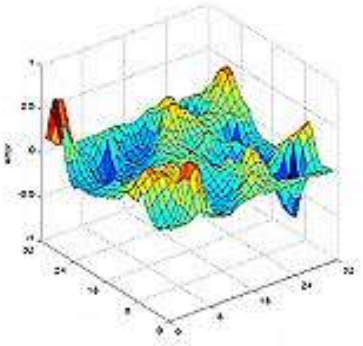
I/O for Large-scale Scientific Computing

- Reading input and restart files
- Reading and processing large amount of data
- Writing checkpoint files
- Writing movie, history files
- Applications tend to be data intensive



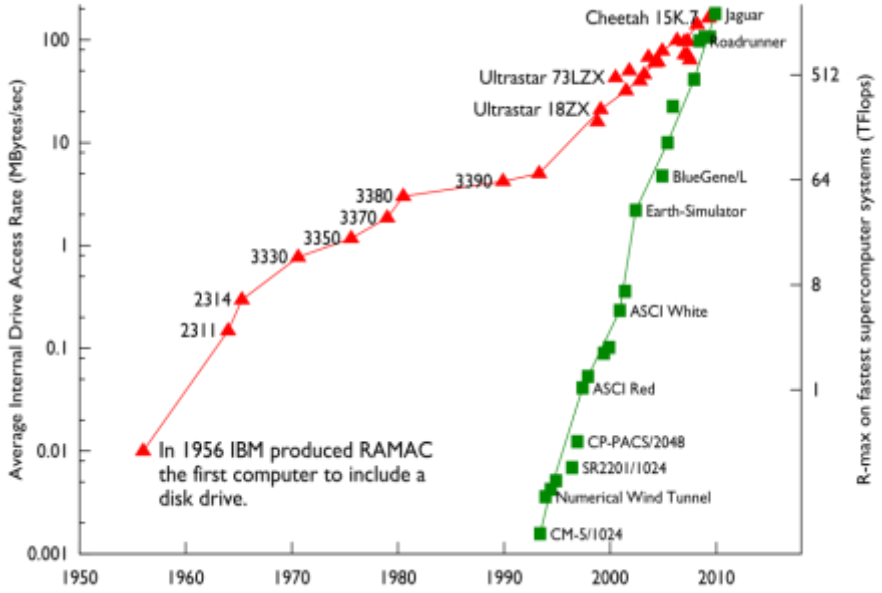




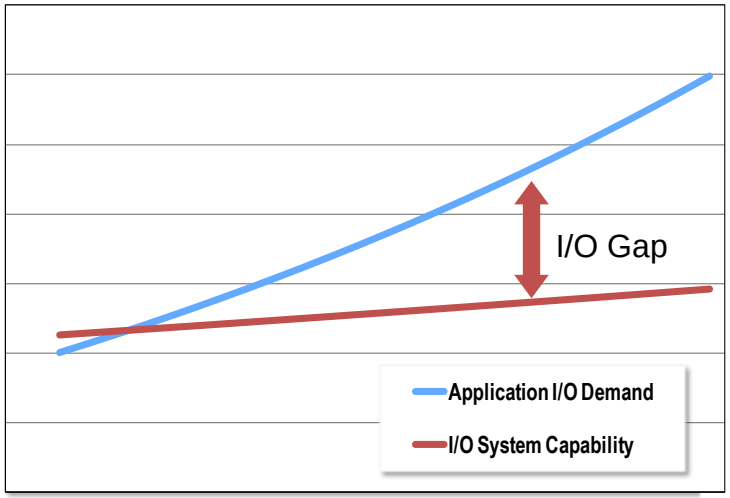


The I/O Gap

- Widening gap between computing and I/O
- Widening gap between demands and I/O capability
- Long I/O access latency leads to severe overall performance degradation
- Limited I/O capability attributed as the cause of low sustained performance



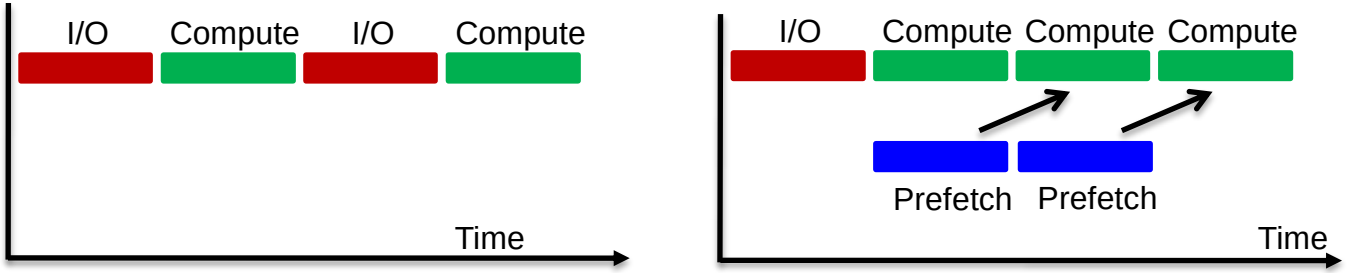
FLOPS v.s. Disk Bandwidth



System size

Bridging Gap: Prefetching

- Move data in advance and closer
- Improve I/O system capability



- Representative existing works
  - Patterson and Gibson, TIP, SOSP'95
  - Tran and Reed, time series model based, TPDS'04
  - Yang et. al, speculative execution, USENIX'02
  - Byna et. al, signature based, SC'08
  - Blas et. al, multi-level caching and prefetching for BGP, PVM/MPI'09

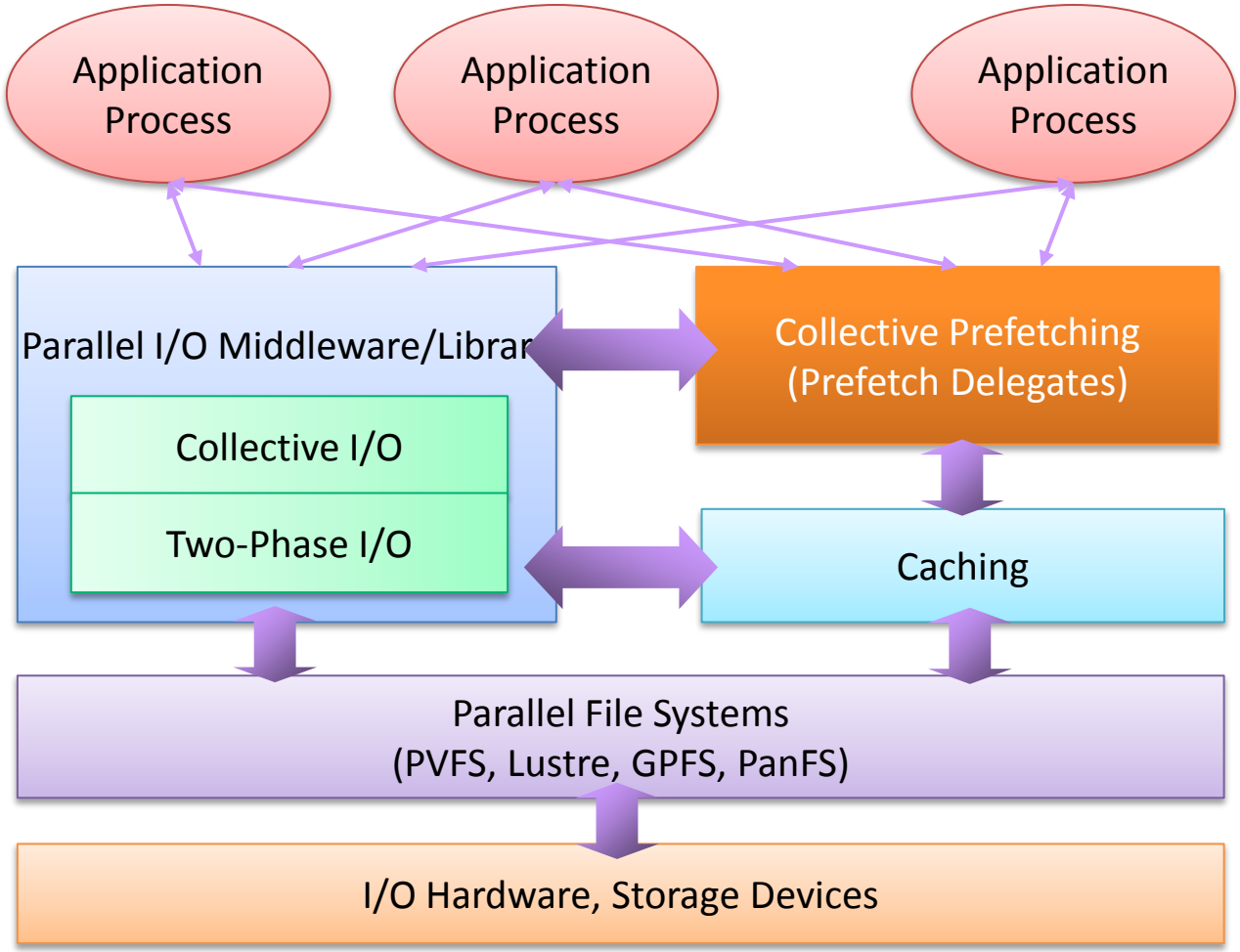
Limitation of Existing Strategies

- The effectiveness of I/O prefetching depends on carrying out prefetches efficiently and moving data swiftly
- Existing studies take an independent approach, without considering the correlation of accesses among processes
  - *Independent prefetching*
- Multiple processes of parallel applications have strong correlation with each other with respect to I/O accesses
  - Foundation of collective I/O, data sieving, etc.
- We propose to take advantage of this correlation
  - Parallel I/O prefetching should be done in a collective way rather an ad hoc individual and independent way

Collective Prefetching Idea

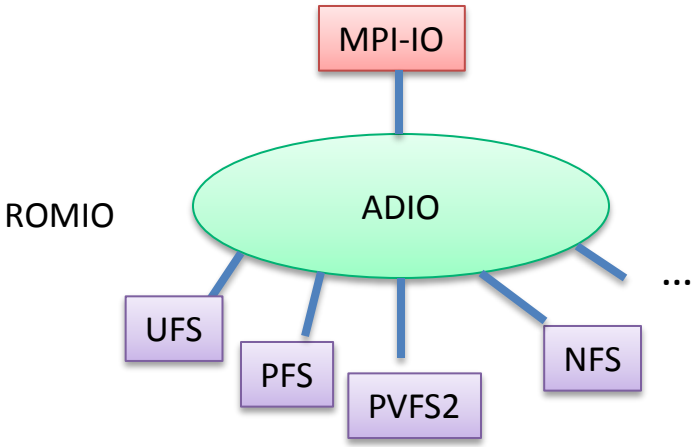
- Take advantage of the correlation among I/O accesses of multiple processes to optimize prefetching
- Benefits/features
  - Filter overlapping and redundant prefetch requests
  - Combine prefetch requests from multiple processes
  - Combine demand requests with prefetch requests
  - Form large and contiguous requests
  - Reduces system calls
- Similar mechanism exploited in optimizations like collective I/O, data sieving, but no study for prefetching yet

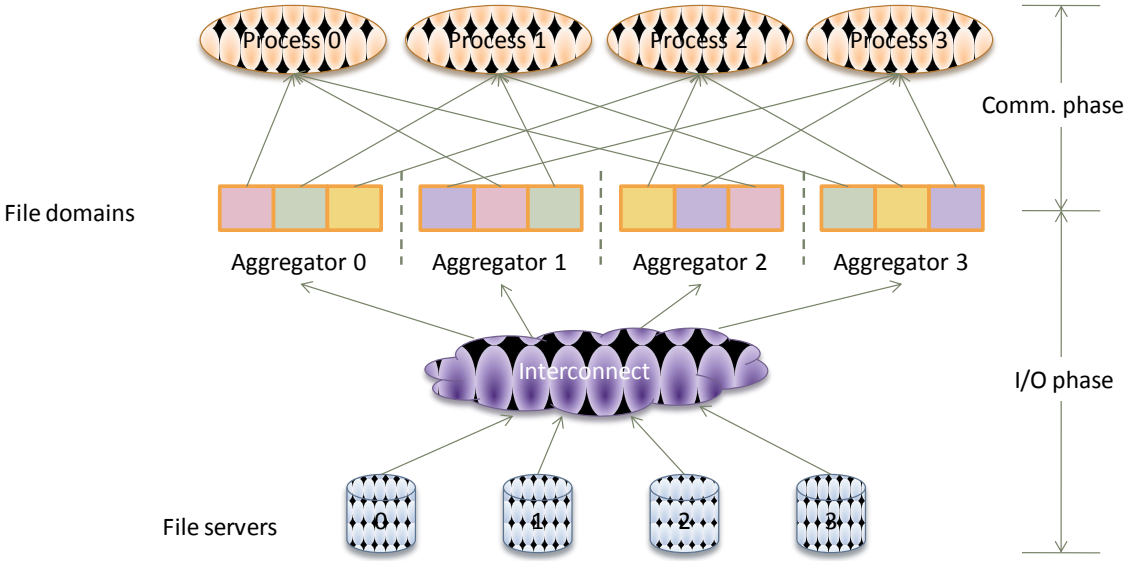
Collective Prefetching Framework



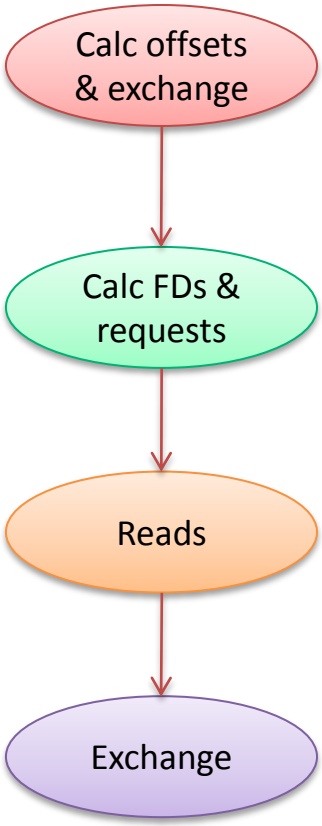
MPI-IO with Collective Prefetching

- MPI-IO and ROMIO
- Collective I/O and Two-phase Implementation



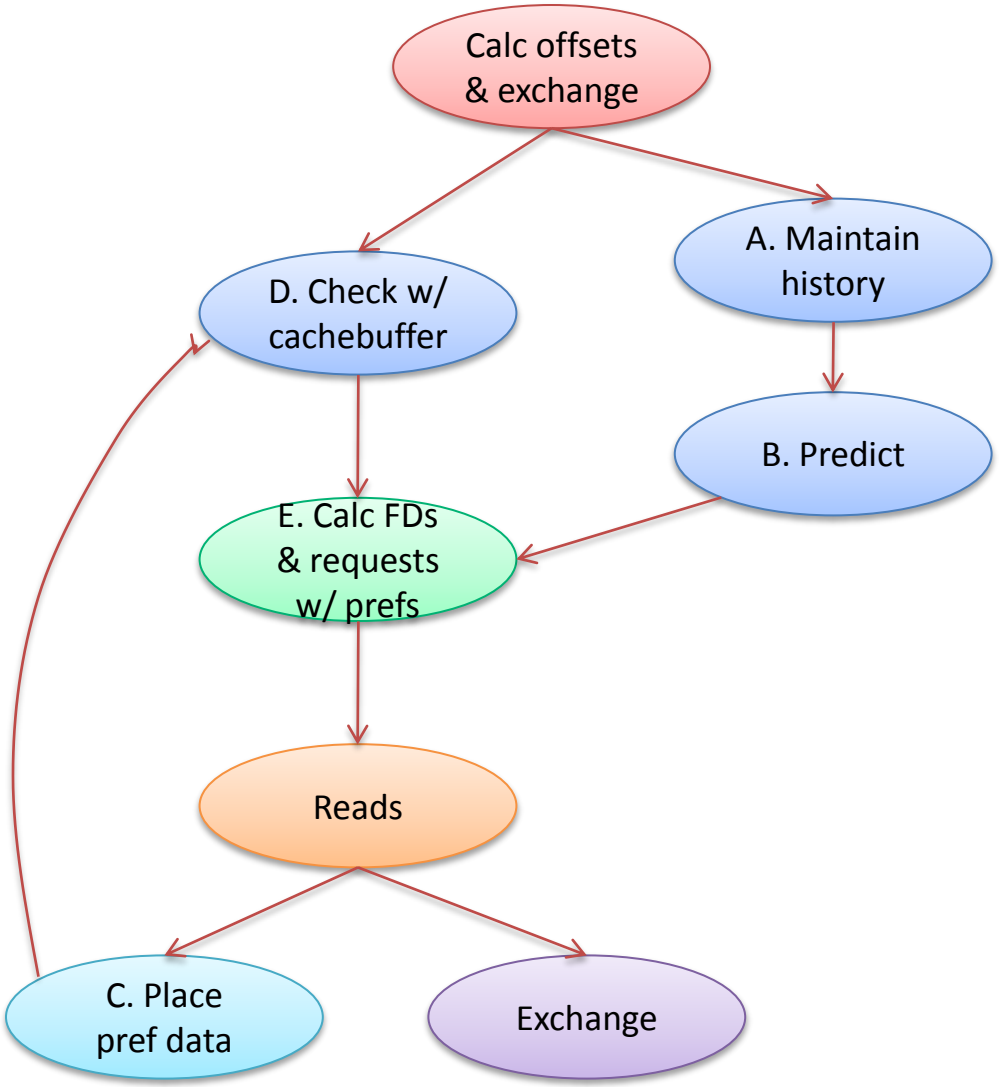


Two-Phase Read Protocol in ROMIO



- Each aggregator calculates the I/O requests span and exchange
- Partitions the aggregated span into file domains
- Each aggregator carries out I/O requests for its own file domain
- All aggregators send data to the requesting processes, and each process receives its required data

Extended Protocol with Collective Prefetching



Collective Prefetching Algorithm

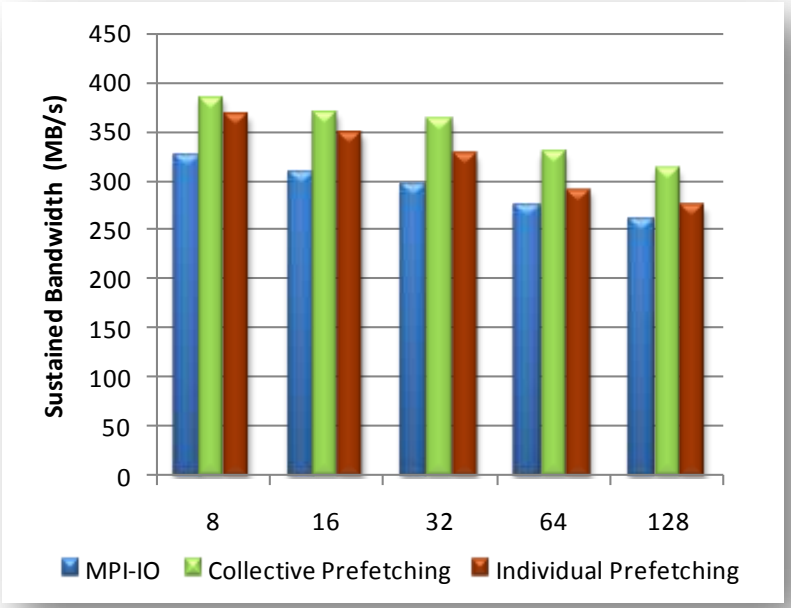
Algorithm *cpf /* Collective Prefetching at MPI-IO */*
Input: *I/O request offset list, I/O request length list*
Output: *none*
Begin

1. Each aggregator maintains recent access history of window size *w*
2. Aggregators/prefetch delegates run prediction or mining algorithms on all tracked global access history
    1. Algorithms can be as streaming, strided, Markov, or advanced mining algorithms such as PCA/ANN
3. Generate prefetch requests and enqueue them in PFQ
4. Process requests in PFQs together with demand accesses
5. Filter out overlapping and redundant requests
6. Perform extended two-phase I/O protocol with prefetch requests
    1. Prefetched data are kept in cache buffer to satisfy future requests
    2. Exchange data to satisfy demand requests (move data to user buffer)

End

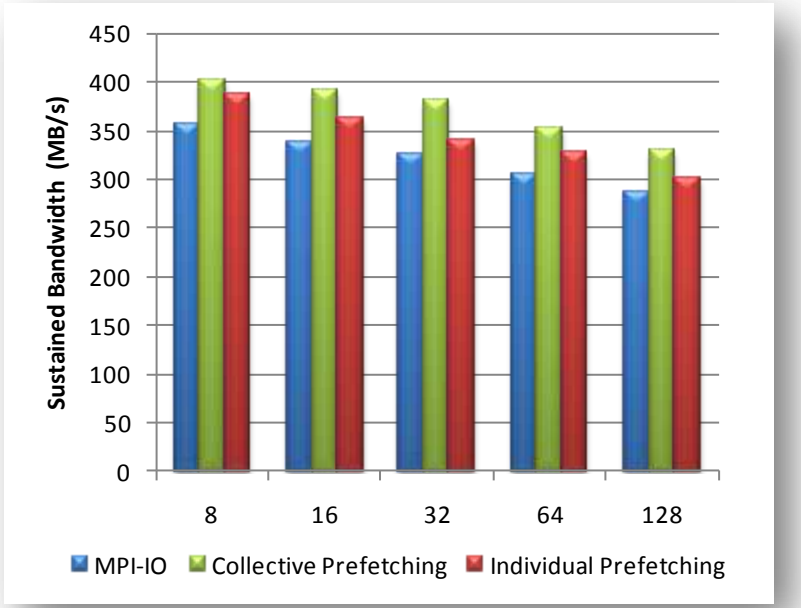
Preliminary Results

- **Strided access pattern, with 1MB and 4MB strides**



With 1MB stride

Collective prefetching: up to 22%, 19% on average
Individual prefetching: up to 12%, 8% on average

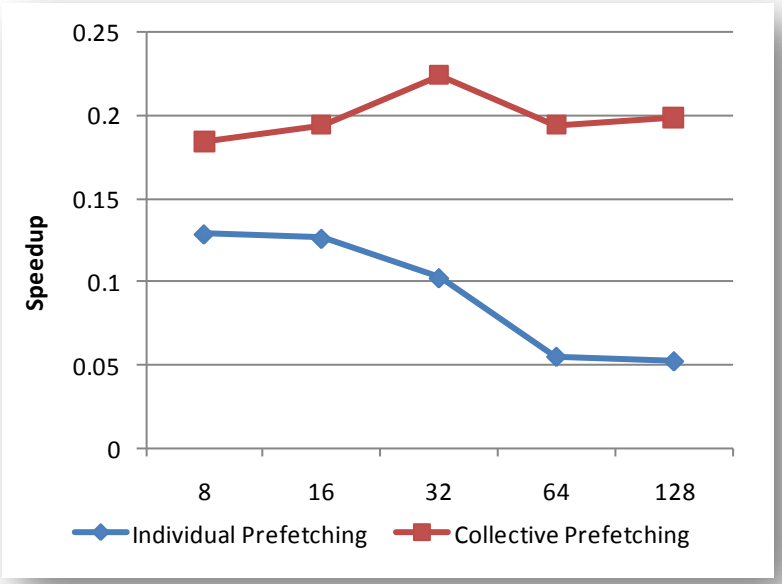


With 4MB stride

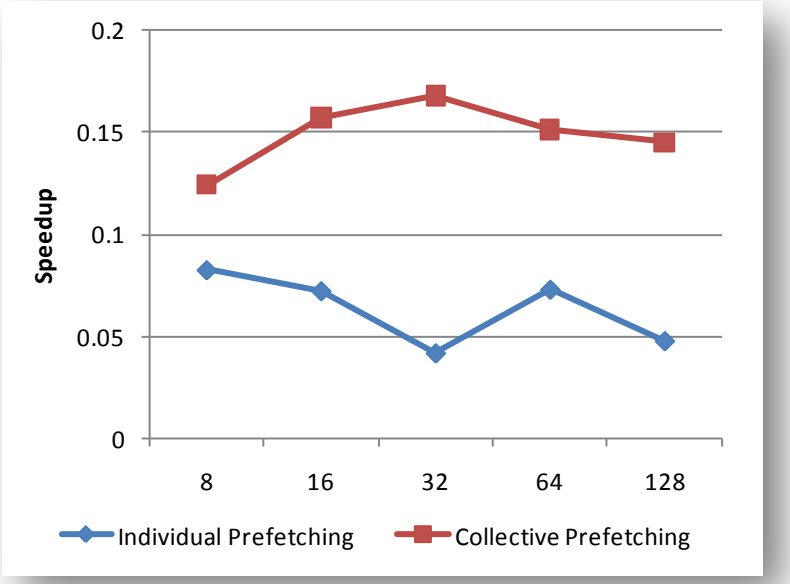
Collective prefetching: up to 17%, 15% on average
Individual prefetching: up to 8%, 6% on average

Preliminary Results

- **Strided access pattern, with 1MB and 4MB strides**



With 1MB stride

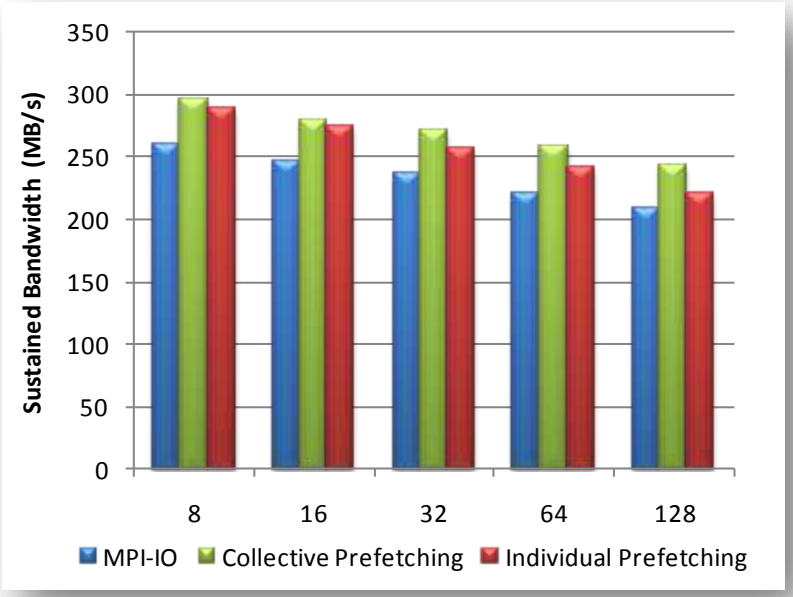


With 4MB stride

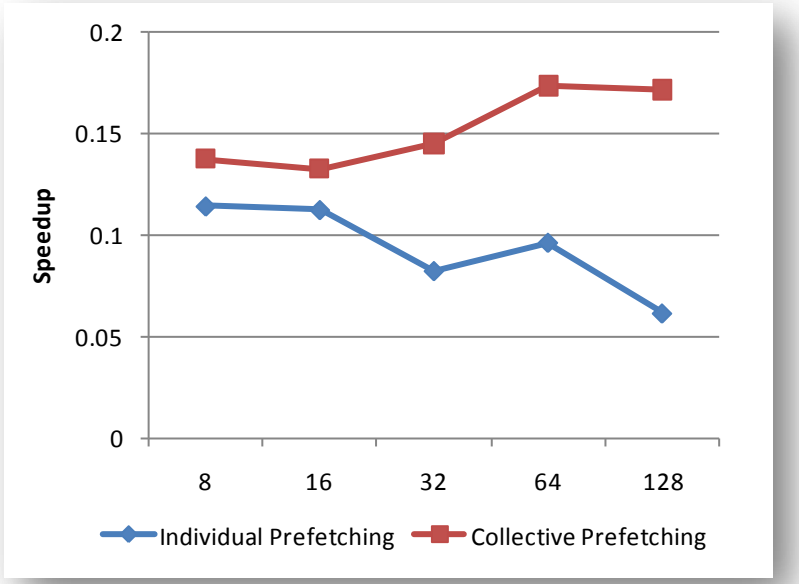
- **Collective prefetching outperformed by over one fold**
- **Collective prefetching had a more stable performance trend**

Preliminary Results

- **Nested strided access pattern, with (1MB, 3MB) stride**



Bandwidth



Speedup

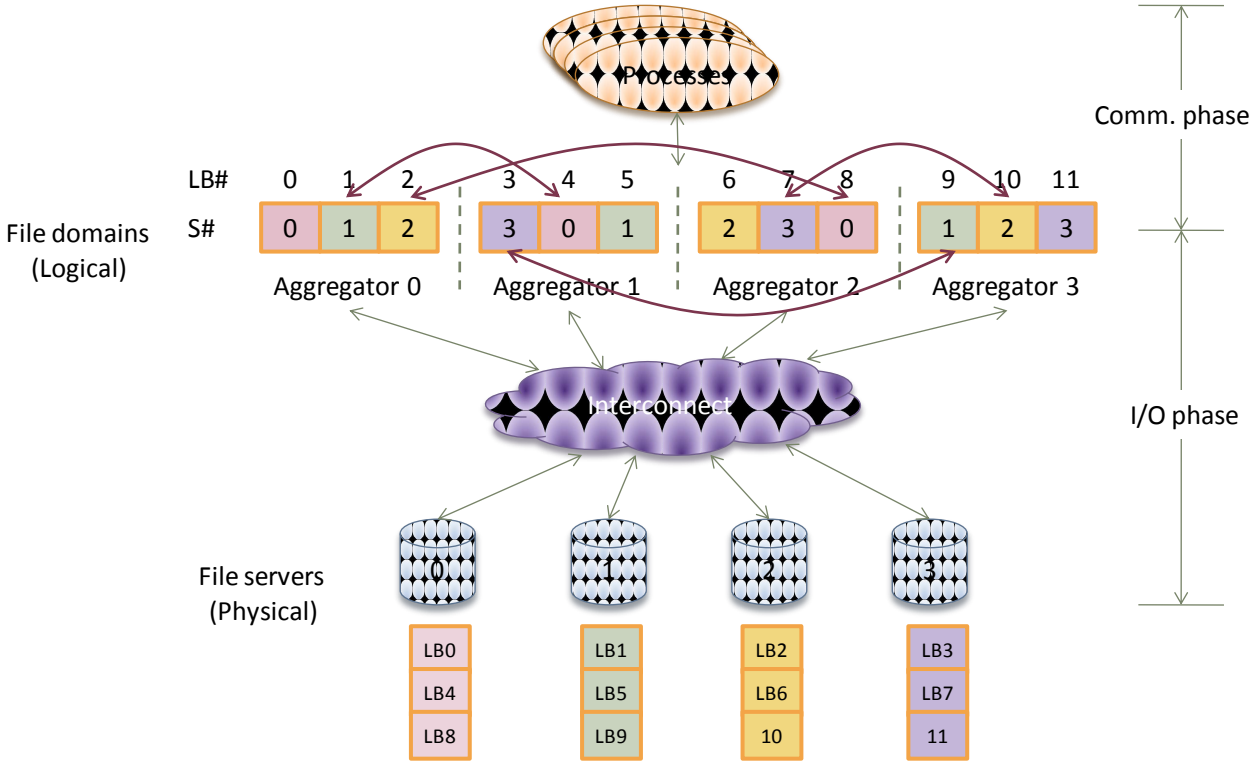
- **Collective prefetching outperformed by over 66%**
- **Collective prefetching had a similar stable performance trend**

Conclusion

- I/O has been widely recognized as the performance bottleneck for many HEC/HPC applications
- Correlation of I/O accesses exploited in data sieving and collective I/O, but no study exploit for prefetching yet
- We propose a new form of collective prefetching for parallel I/O systems
- Preliminary results have demonstrated the potential
- A general idea that can be applied at many levels, such as the storage device level or server level

Ongoing and Future Work

- Exploit the potential at the server level
- LACIO: A New Collective I/O Strategy, and I/O customization



Any Questions?

Thank you.

- **Acknowledgement: Prof. Xian-He Sun of Illinois Institute of Technology, Dr. Rajeev Thakur of Argonne National Lab, Prof. Wei-Keng Liao and Prof. Alok Choudary of Northwestern University.**