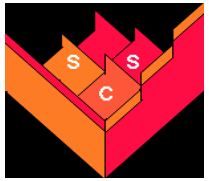


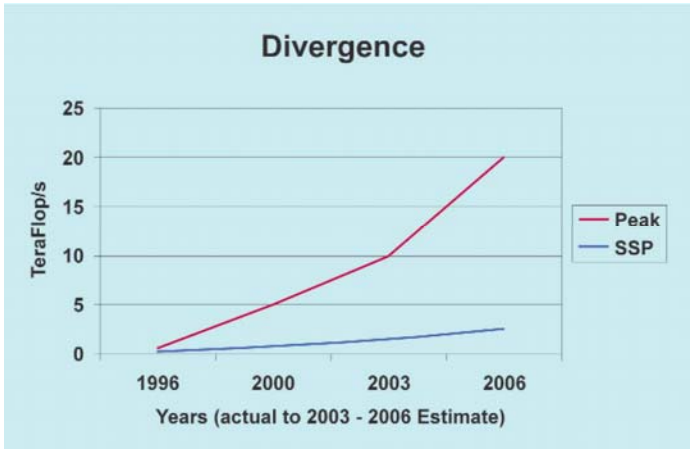
HECURA: *The Server-Push I/O Architecture for High-End Computing*

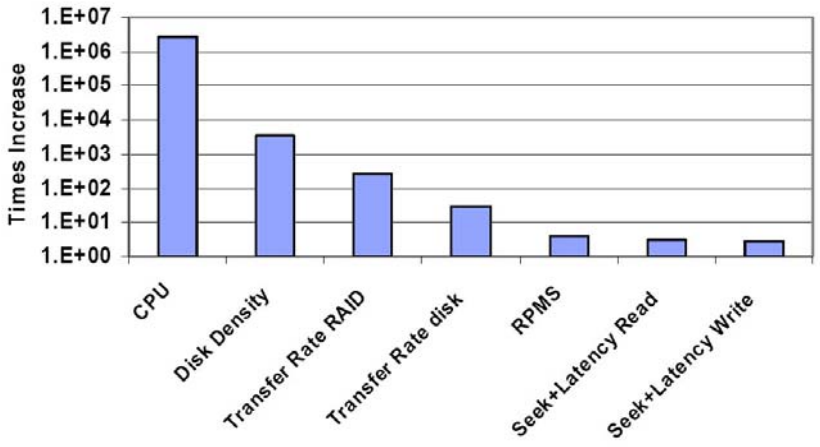
Xian-He Sun
William Gropp, Rajeev Thakur

Illinois Institute of Technology
Argonne National Laboratory
sun@iit.edu

Data Access: the last mile

- Parallel I/O does not work well for complex non-contiguous access
- Improving the performance of small I/O requests is a necessity
- Cache and Prefetching- fetch the data before the demands for it
- Limitation of Existing Prefetching
  - Conservative and limited to static prediction strategies
  - No prediction strategy on when to prefetch
  - Only works for simple access patterns with locality

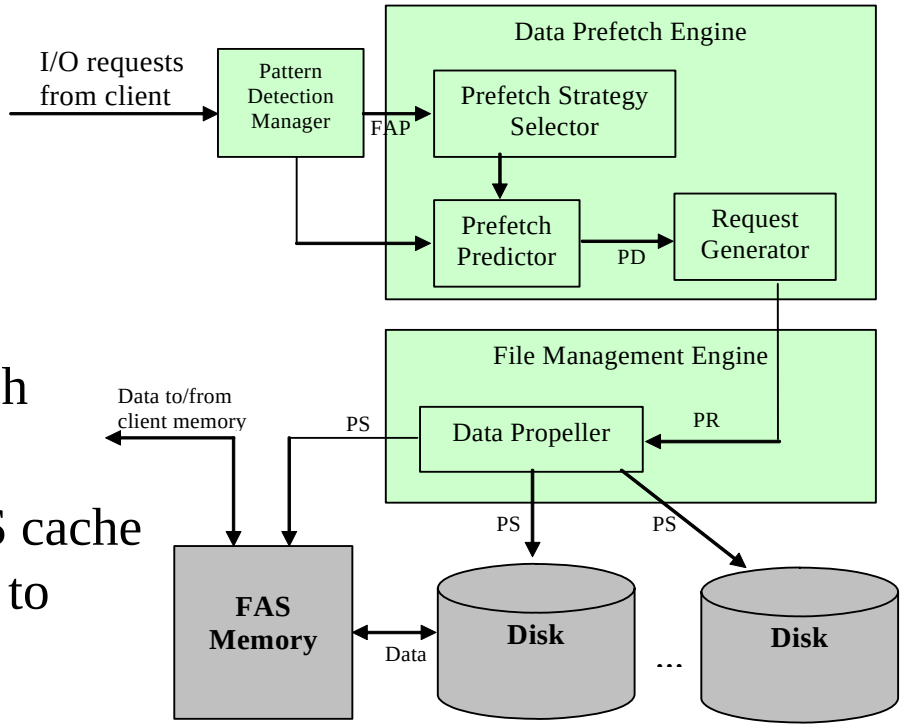




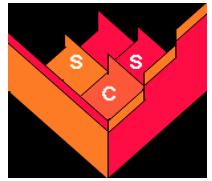
Our Solution: File Access Server (FAS)

- The server pro-actively "pushes" data in time
  - Push: data is sent before the client's I/O request
  - In time: data arrives the destination within a window of time
- Use of adaptive and advanced prediction algorithms

- Prefetch Engine (PFE)
  - Prefetch predictor (*What*)
  - Request generator (*When*)
- File Management Engine
  - Data Propeller: Issues the prefetch instructions
  - Pushes the data from disk to FAS cache and pushes data from FAS cache to client-side disk cache.

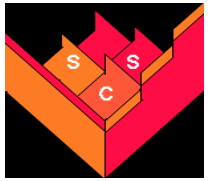


FAS: Functionality



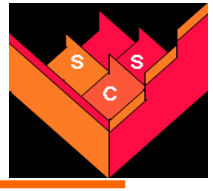
- Pattern Detection Manager: collects the history of past I/O access patterns
- Prefetch Strategy Selector: adaptively selects an appropriate method to predict future accesses
- Prefetch predictor: decides *what* data to fetch
- Request generator: decides *when* to fetch the data
- Data propeller: carries the prefetching and pushes the data into the appropriate disk cache.
- If the prefetching fails, the file memory engine handles the page fault as traditional file servers.

Challenge: What data to push?



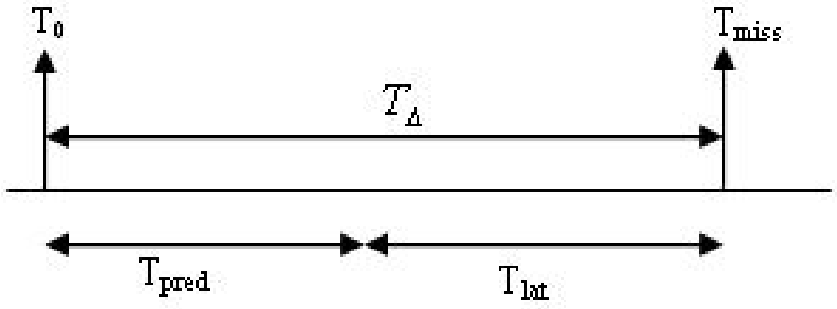
- Multi-dimension
  - location of data, the amount of data, the mode of accessing data, and strides
  - Time between any two accesses, between successive accesses to a specific data block
- Aggressive Prefetching
  - Overhead to predict the future accesses is no longer a issue
  - New aggressive methods to predict irregular data accesses
- Adapt a prefetch strategy based on the data access pattern
- Reduce prediction time by using hints provided by compiler and application/user

Challenge: When to Fetch

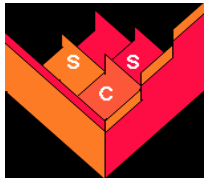


Three factors

- Time to predict the future accesses
  - Based on the chosen prefetching method
- Data transfer latency
  - I/O access delay model
- Time till next I/O cache miss
  - Data access model
- Overlapping the network latency by increasing the prefetch distance
- Adapting the prefetch distance based on the network latency variation



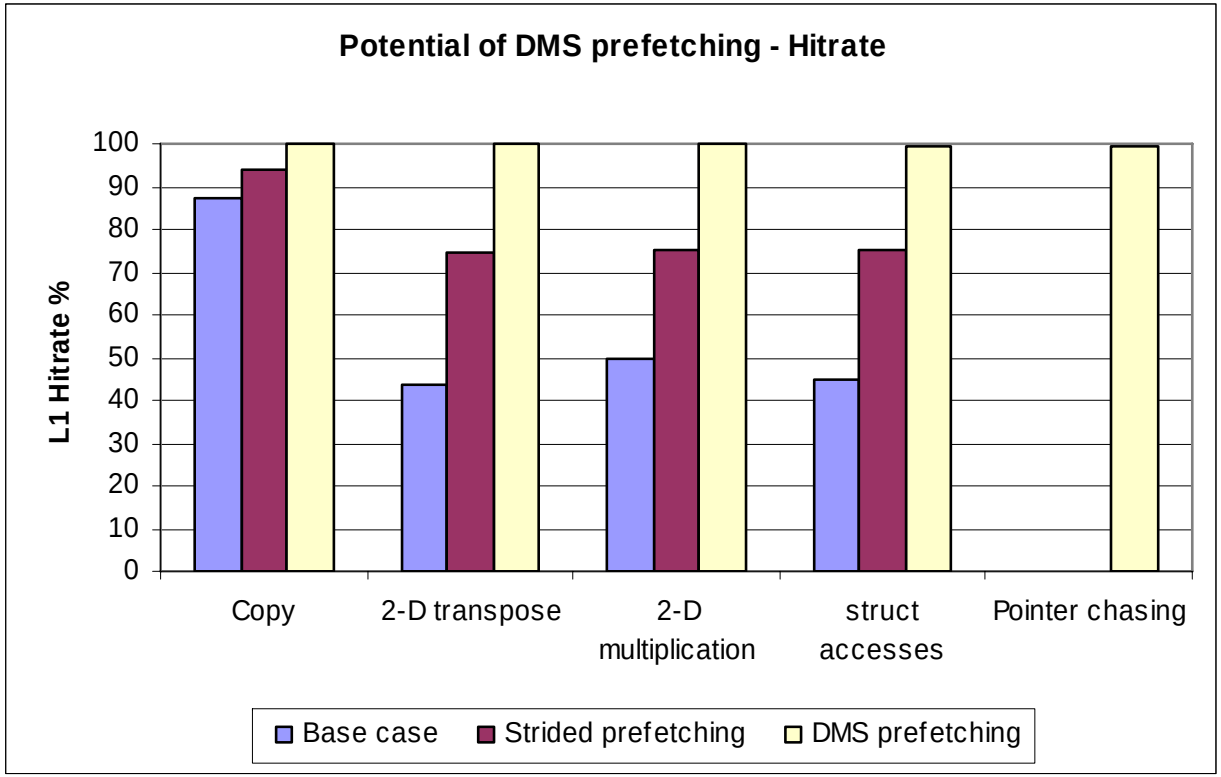
Challenge: Replacement Policies



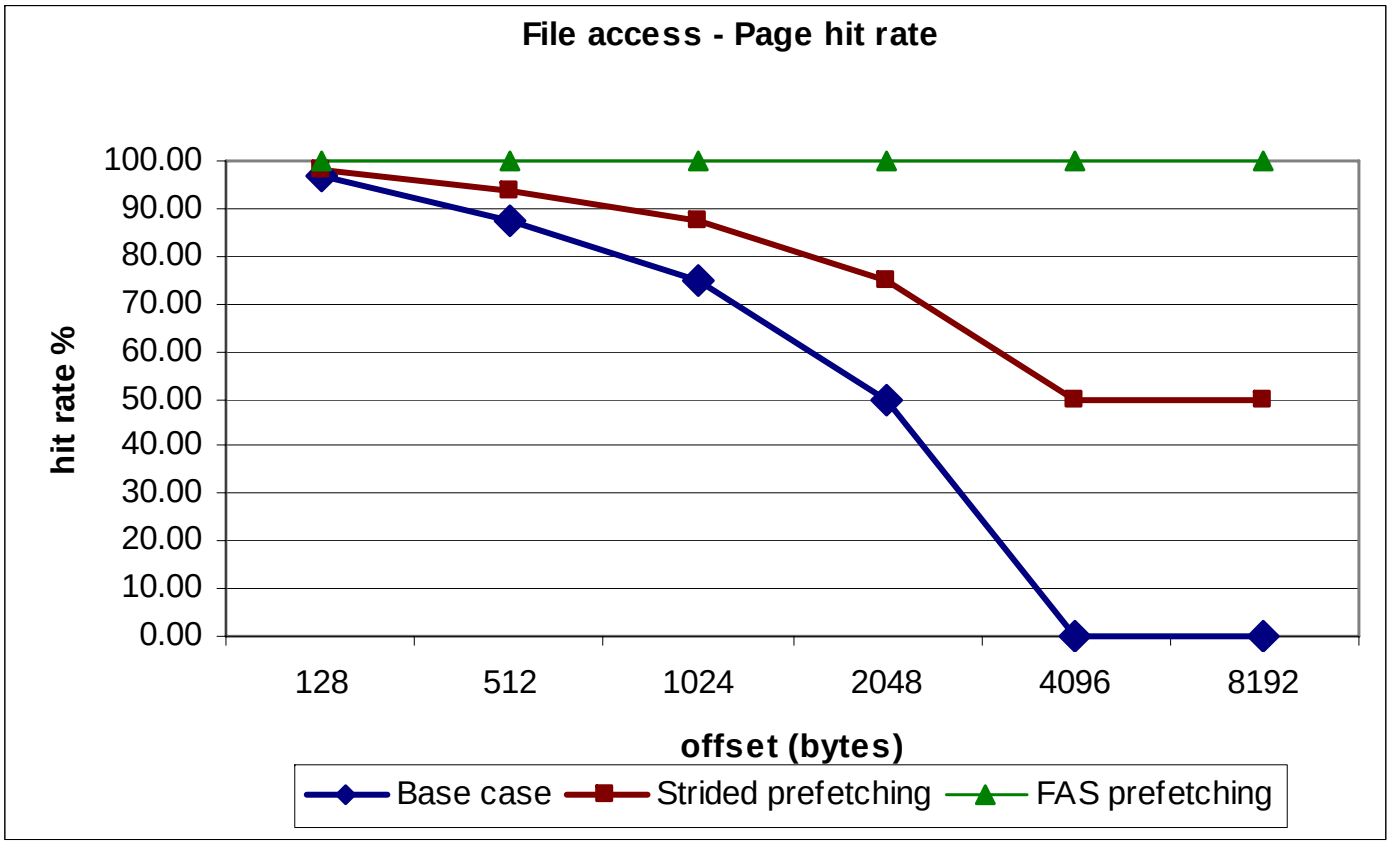
- Existing methods
  - LRU (“recency”)
  - ARC (Adaptive Replacement Cache) (“recency” and “frequency”)
- Prediction-based Adaptive Replacement (PAR)
  - A victim data block is selected by ARC policy
  - Verify if the selected victim is in the predicted list
  - If yes, select another one via ARC
- Optimize collective I/O:
  - Detection and prediction algorithms
  - User hint

Simulation Results: Memory Caching

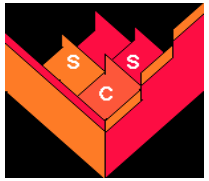
Modified Simplescalar Simulator



Simulation Results:– File accesses



Research Plan



- Develop File Access Server
  - Design and develop the *Pattern Detection Manager*, *Pattern Detection Manager*, *Prefetch Predictor and Request Generator*, *novel prediction based replacement policies*, *Data Propeller*
  - Testing of correctness and improving efficiency
- Validate the benefits of FAS at parallel file system level
  - Integrate FAS into parallel file systems such as PVFS
  - Collective I/O of the MPI specification.
  - I/O intensive applications
- Education plan
  - Mentor, teaching, cultivate collaborative spirit

Conclusion

- The Push-based FAS is new and has many advantages (over client-directed prefetching)
  - Dedicated computer power, fully supported OS, collective info, not lost in the I/O layers, full control, in time
- A strong team
  - Experience and long collaboration history
- A challenge task with a great potential

Grateful

to the NSF/DOE HECURA support