Using IOR to Analyze the I/O Performance

Hongzhang Shan, John Shalf

NERSC

Motivation

- HPC community has started to build the petaflop platforms.
  - System:
    - Programming Interface
      - How to make programming of increasingly complex file system easily accessible to users
    - I/O scalability:
      - Handling exponentially increasing concurrency
      - scale proportionally to flops ?
  - Application:
    - Workload survey/characterization (what applications dominate our workload)
    - Understanding I/O requirements of key applications
    - Develop or adopt microbenchmarks that reflect those requirements
    - Set performance expectations (now) and targets (future)

Identify Application Requirements

- Identify users with demanding I/O requirements
  - Study NERSC allocations (ERCAP)
  - Study NERSC user surveys
- Approached sampling of top I/O users
  - Astrophysics (Cactus, FLASH, CMB/MadCAP)
  - Materials
  - AMR framework (Chombo), etc.

Survey Results

- **Access Pattern:**
  - Sequential I/O patterns dominate
  - Writes dominate (*exception: out-of-core CMB)*
- **Size of I/O Transaction**
  - Broad Range: *1KB - tens of MB*
- **Typical Strategies for I/O**
  - Run all I/O through one processor (serial)
  - One file per processor (multi-file parallel I/O)
  - MPI-IO to single file (single-file parallel I/O)
  - pHDF5 and parallelNetCDF (advanced self-describing, platform-neutral file formats)

Potential Problems

- Run all I/O through one processor
  - Potential performance bottleneck
  - Does not fit distributed memory
- One file per Processor
  - High overhead for metadata management
    - A recent FLASH run on BG/L generates 75 million files
  - Bad for archival storage (lots of small files)
  - Bad for metadata servers (lots of file creates)
  - Bad for data analysis

- Need to use shared files or new interface

Migration to Parallel I/O

- Parallel I/O to single file is slowly emerging
  - Used to imply MPI-IO for correctness, but concurrent Posix also works (now)
  - Motivated by need for fewer files
  - Simplifies data analysis, visualization
  - Simplifies archival storage
- Modest migration to high-level file formats pHDF5, parallelNetCDF
  - Motivated by portability & provenance concerns
  - Concerns about overhead of advanced file formats

Benchmark Requirements

- Need to develop or adopt benchmark that reflects application requirements
  - Access Pattern
  - File Type
  - Programming Interface
  - File Size
  - Transaction Size
  - Concurrency

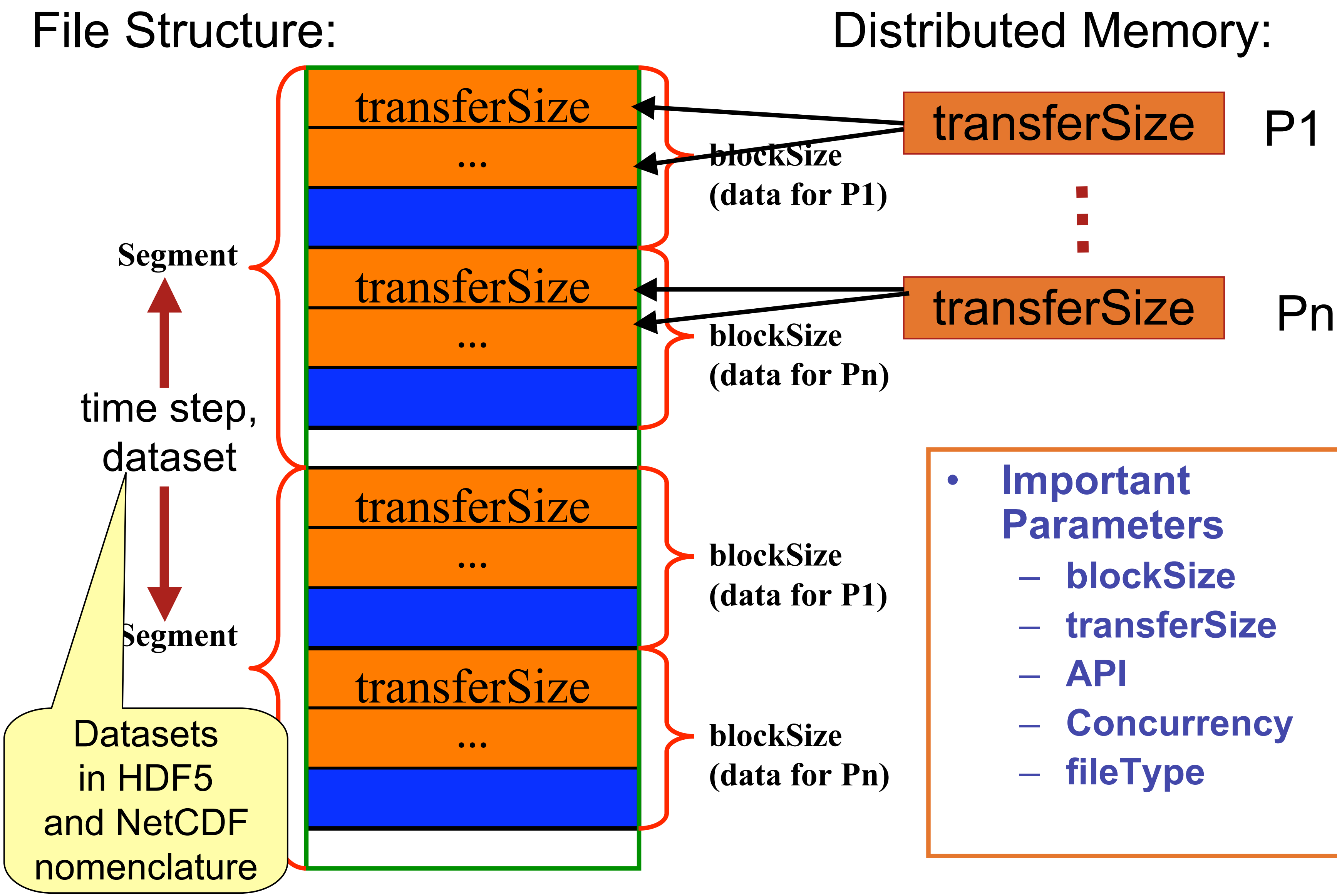
Synthetic Benchmarks

- Most synthetic benchmarks cannot be related to observed application IO patterns
  - Iozone, Bonnie, Self-Scaling benchmark, SDSC I/O benchmark, Effective I/O Bandwidth, IOR, etc
- Deficiencies
  - Access pattern not realistic for HPC
  - Limited programming interface
  - Serial only

LLNL IOR Benchmark

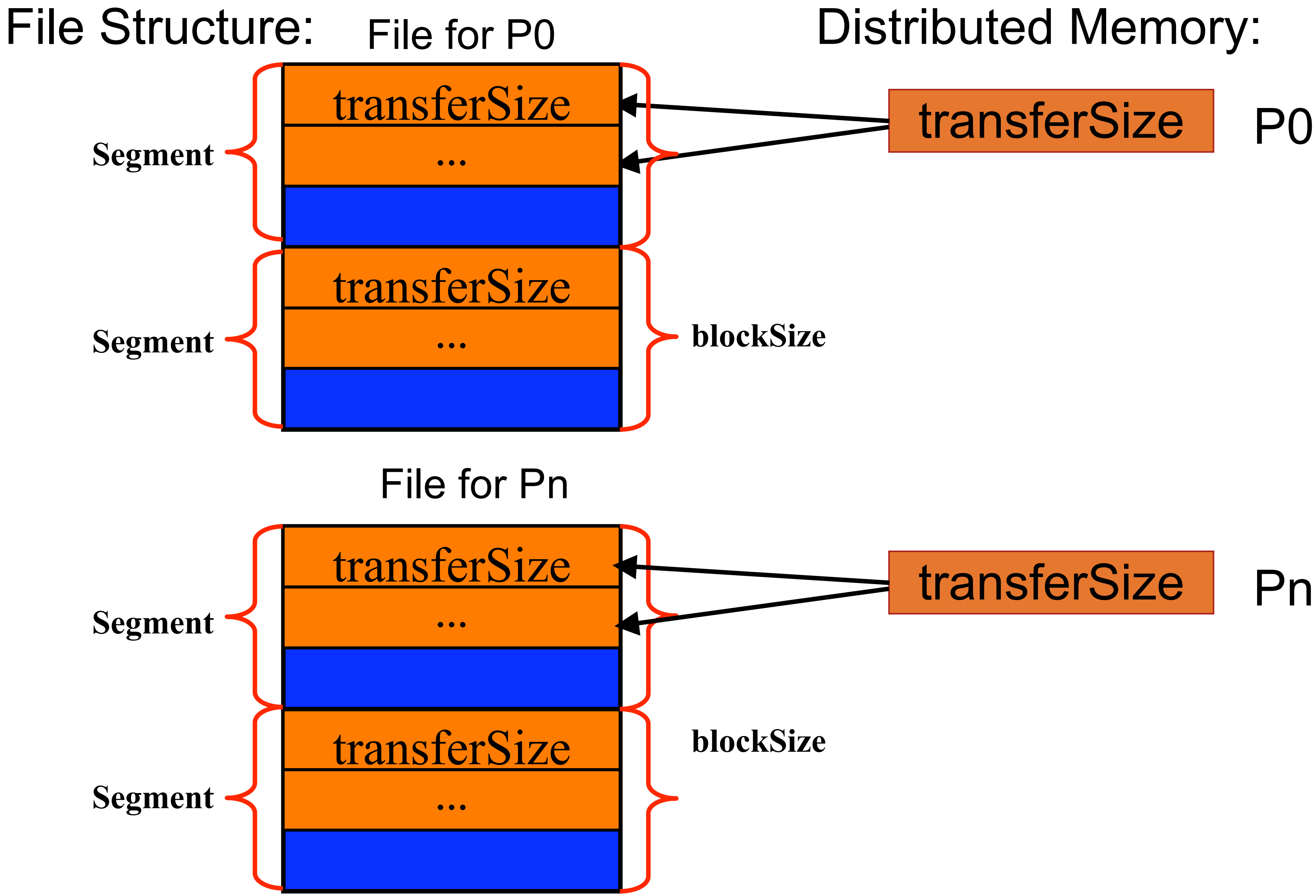
- Developed by LLNL, used for purple procurement
- Focuses on parallel/sequential read/write operations that are typical in scientific applications
- Can exercise one file per processor or shared file accesses for common set of testing parameters (differential study)
- Exercises array of modern file APIs such as MPI-IO, POSIX (shared or unshared), pHDF5, parallelNetCDF
- Parameterized parallel file access patterns to mimic different application situations

IOR Design (shared file)



- **Important Parameters**
  - blockSize
  - transferSize
  - API
  - Concurrency
  - fileType

IOR Design
(One file per processor)



Outline

- Why IOR ?
- **Using IOR to study system performance**
- Using IOR to predict I/O performance for application

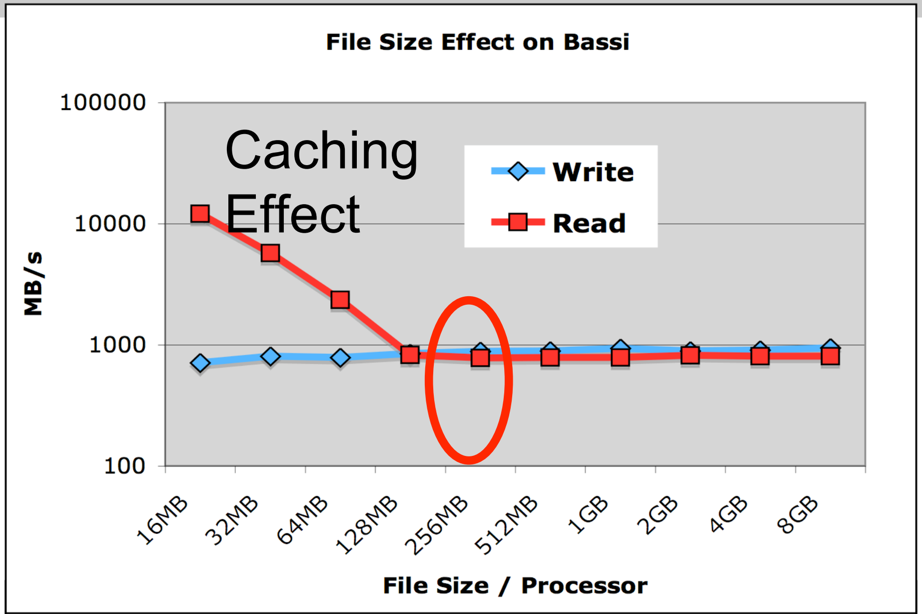
Platforms

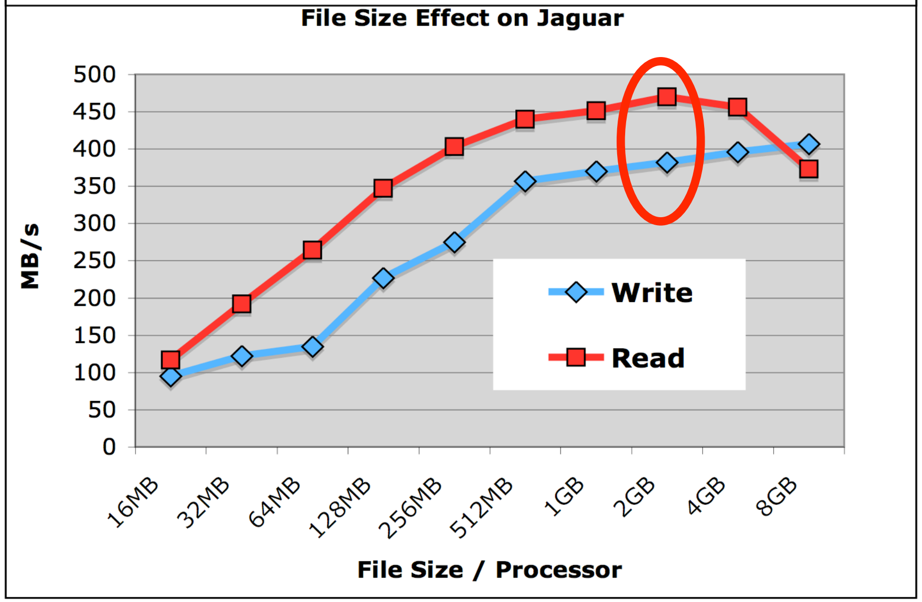
Machine Name	Parallel File System	Proc Arch	Inter-connect	Peak IO BW	Max Node BW to IO
Jaguar	Lustre	Opteron	SeaStar	18*2.3GB/s = 42GB	3.2GB/s (1.2GB/s)
Bassi	GPFS	Power5	Federation	6*1GB/s = ~6.0GB/s	4.0GB/s (1.6GB/s)

- 18 DDN 9550 couplets on Jaguar, each couplet delivers 2.3 - 3 GB/s
- Bassi has 6 VSDs with 8 non-redundant FC2 channels per VSD to achieve ~1GB/s per VSD. (2x redundancy of FC)

Effective unidirectional bandwidth in parenthesis

Caching Effects

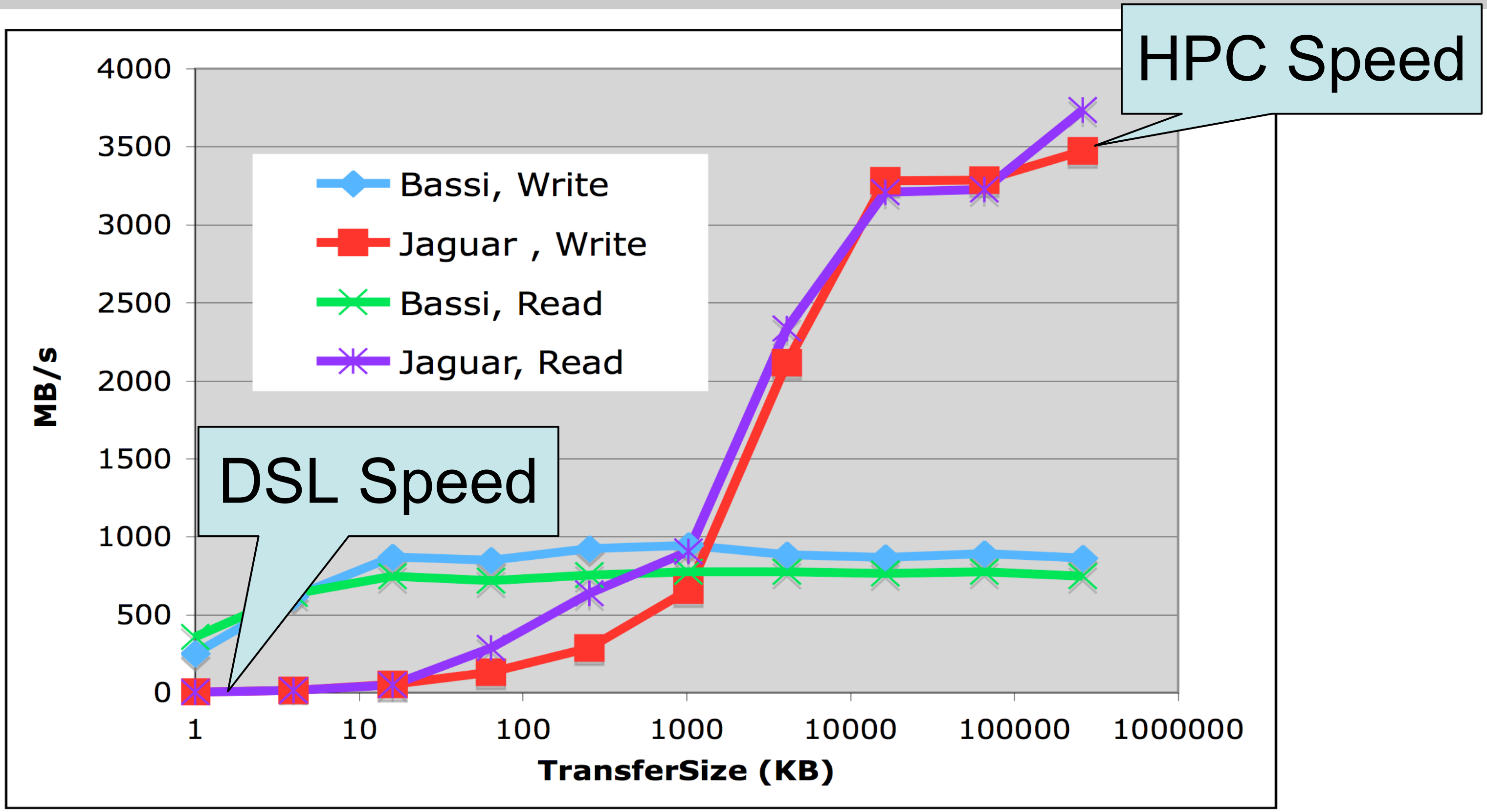




Machine Name	Mem Per Node	Node Size	Mem/ Proc
Jaguar	8GB	2	4GB
Bassi	32GB	8	4GB

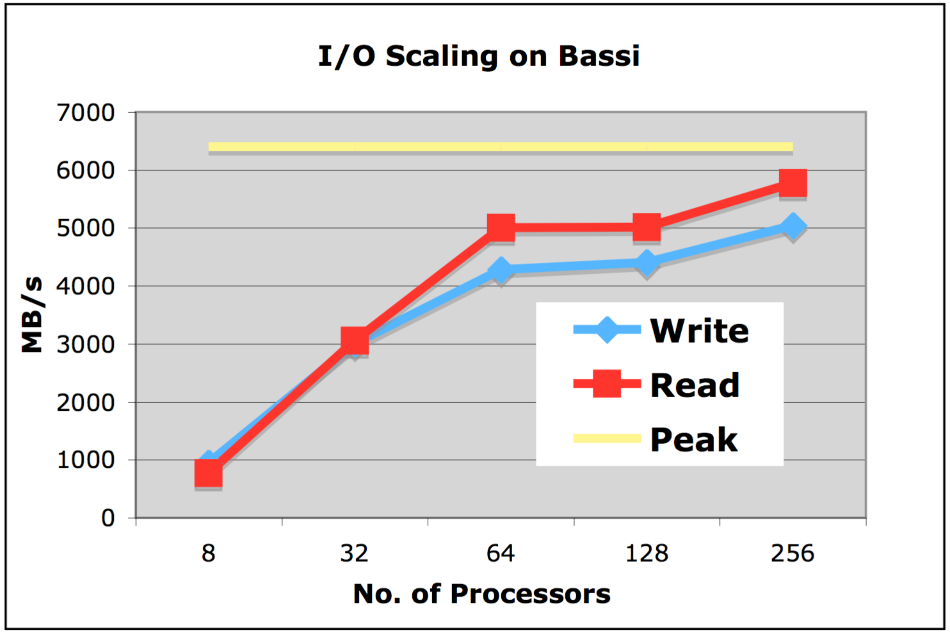
- On Bassi, file Size should be at least 256MB/ proc to avoid caching effect
- On Jaguar, we have not observed caching effect, 2GB/s for stable output

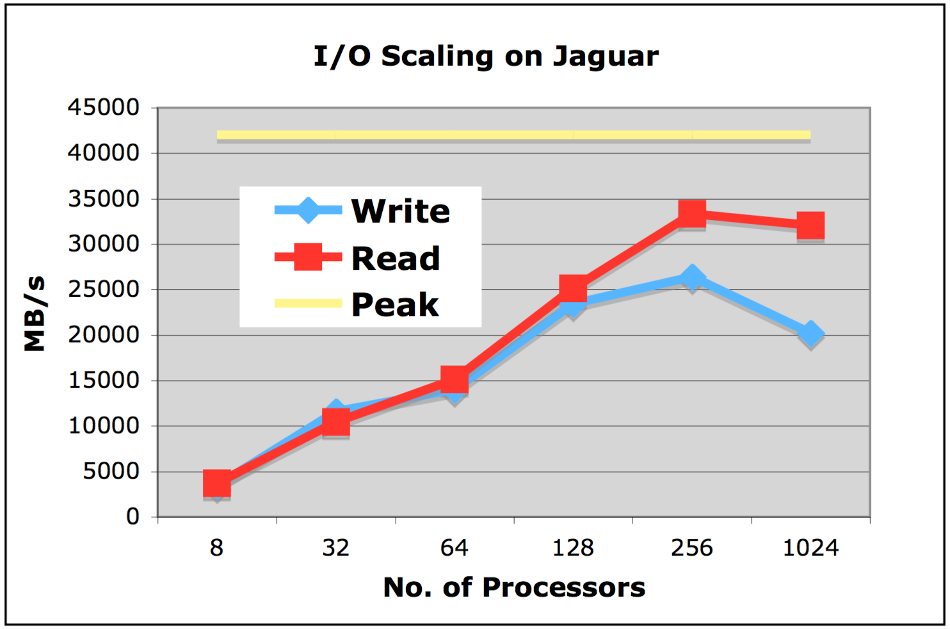
Transfer Size (P = 8)



- **Large transfer size is critical on Jaguar to achieve performance**
- **The effect on Bassi is not as significant**

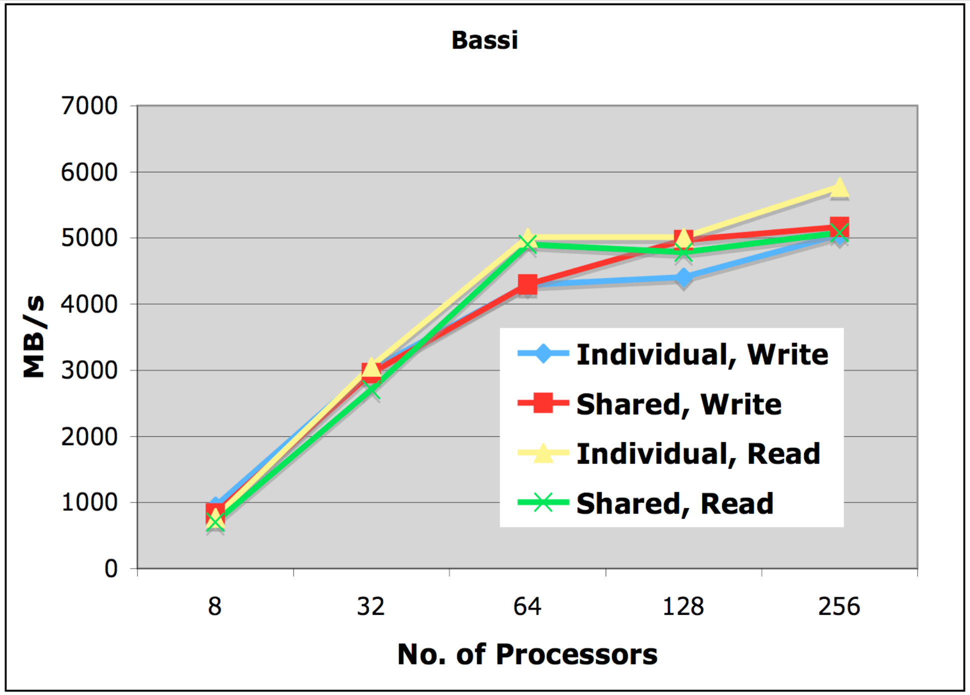
Scaling (No. of Processors)

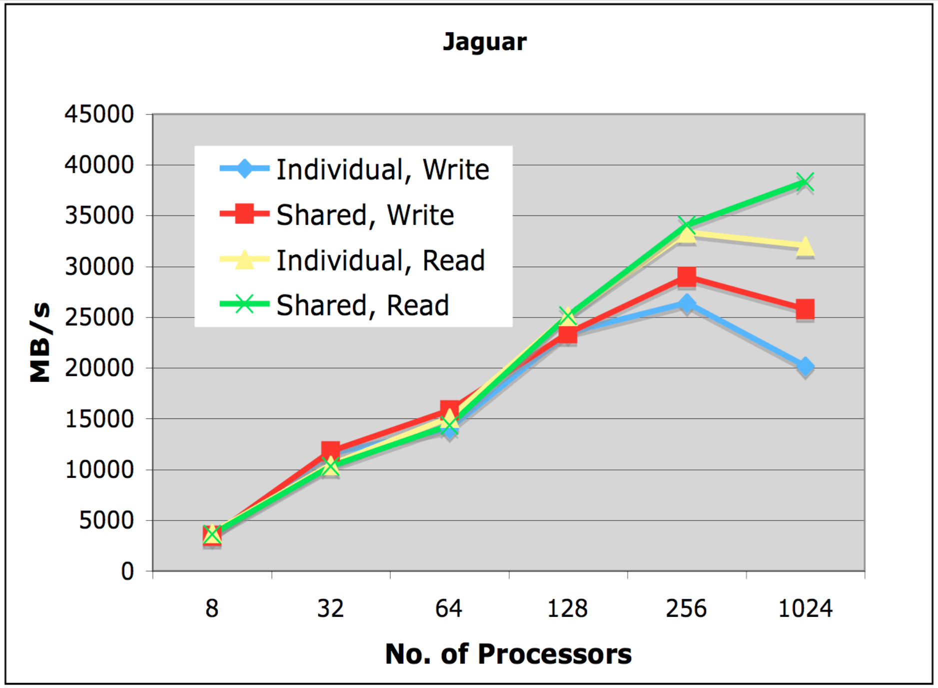




- The I/O performance peaks at:
  - P = 256 on Jaguar (lstripe=144),
  - Close to peaks at P = 64 on Bassi
- The peak of I/O performance can often be achieved at relatively low concurrency

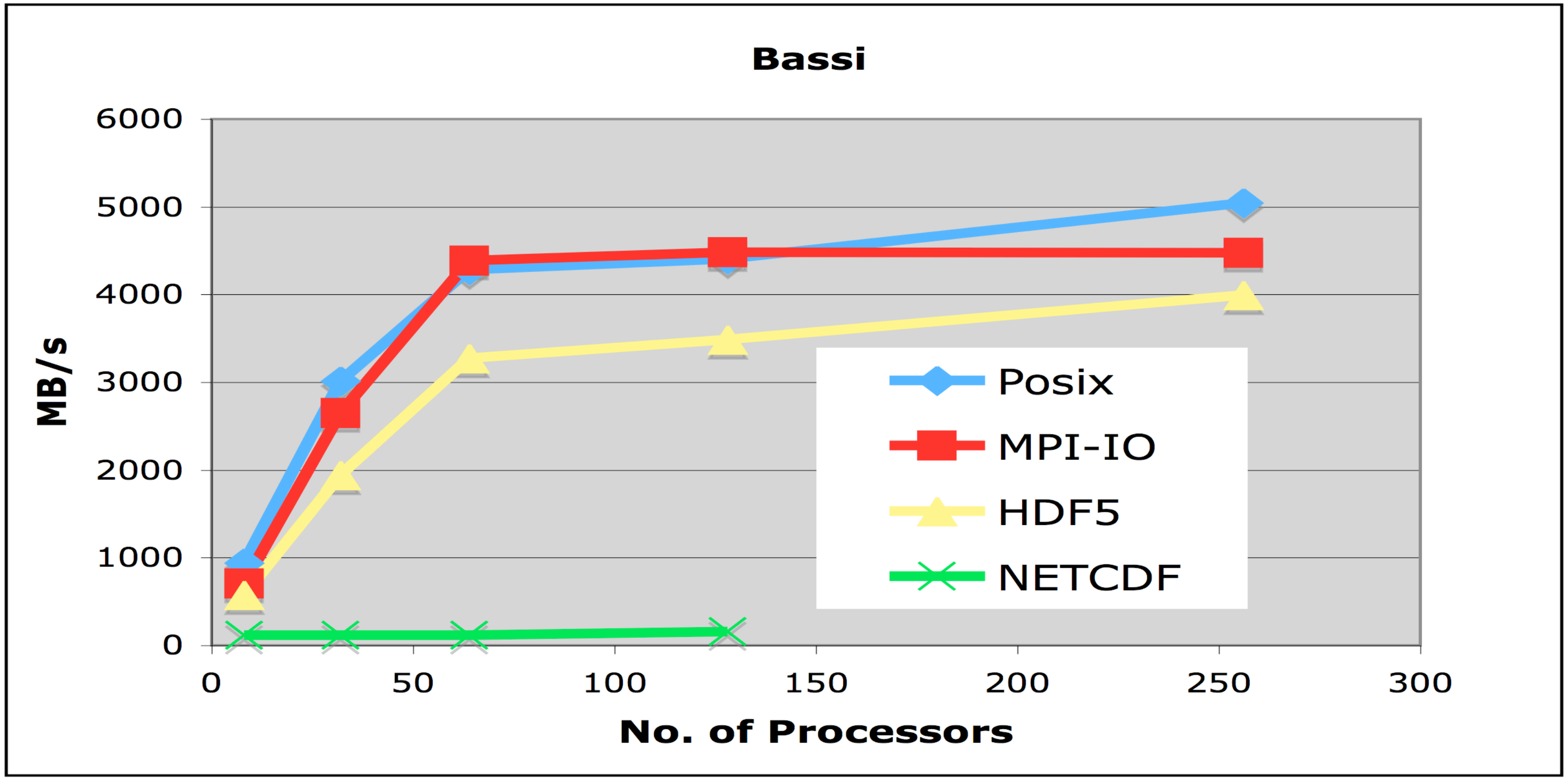
Shared vs. One file Per Proc





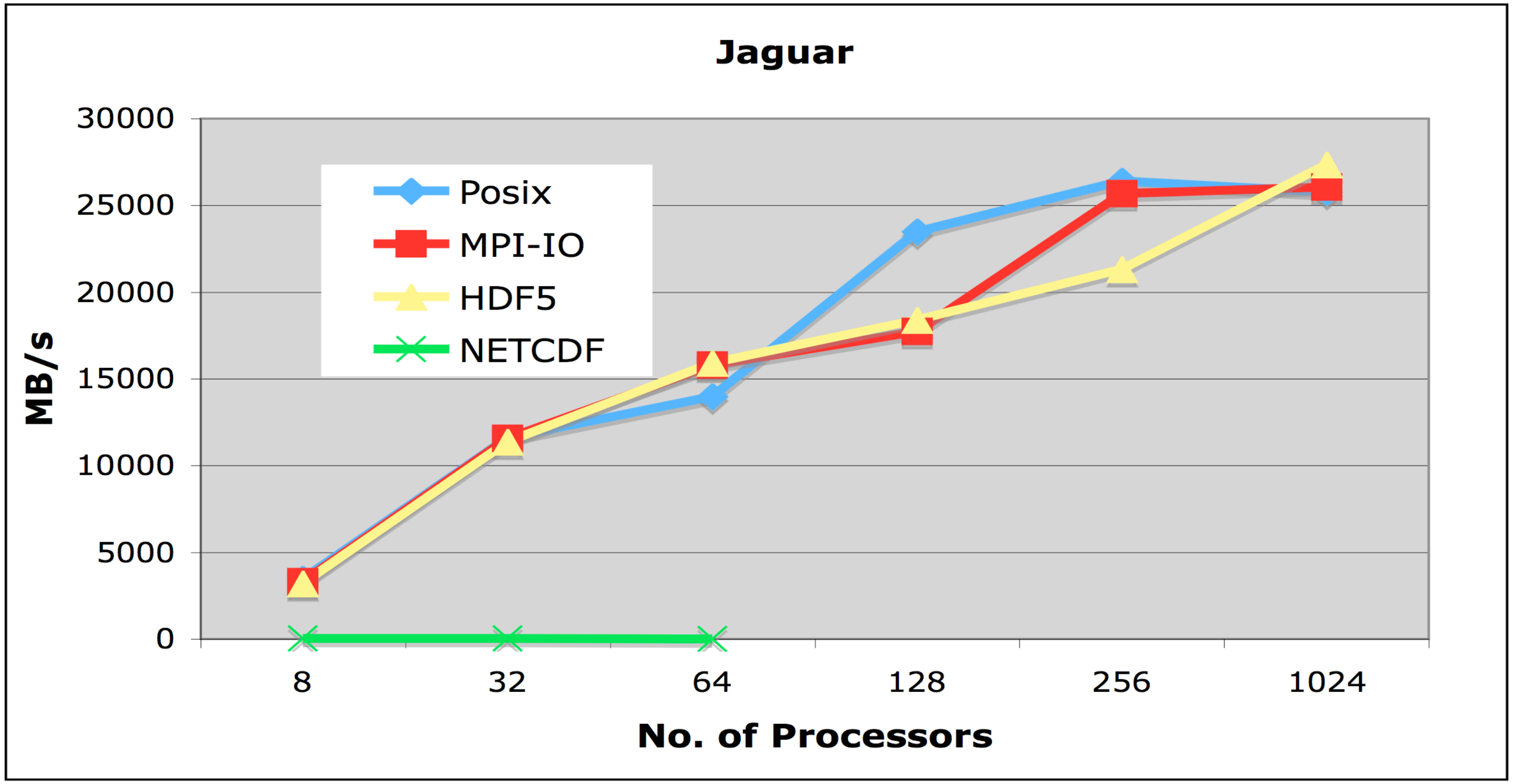
- The performance of using a shared file is very close to using one file per processor
- Using a shared file performs even better on Jaguar due to less metadata overhead

Programming Interface



- **MPI-IO is close to POSIX performance**
- **Concurrent POSIX access to single-file works correctly**
  - **MPI-IO used to be required for correctness, but no longer**
- **HDF5 (v1.6.5) falls a little behind, but tracks MPI-IO performance**
- **parallelNETCDF (v1.0.2pre) performs worst, and still has 4GB dataset size limitation *(due to limits on per-dimension sizes on latest version)***

Programming Interface



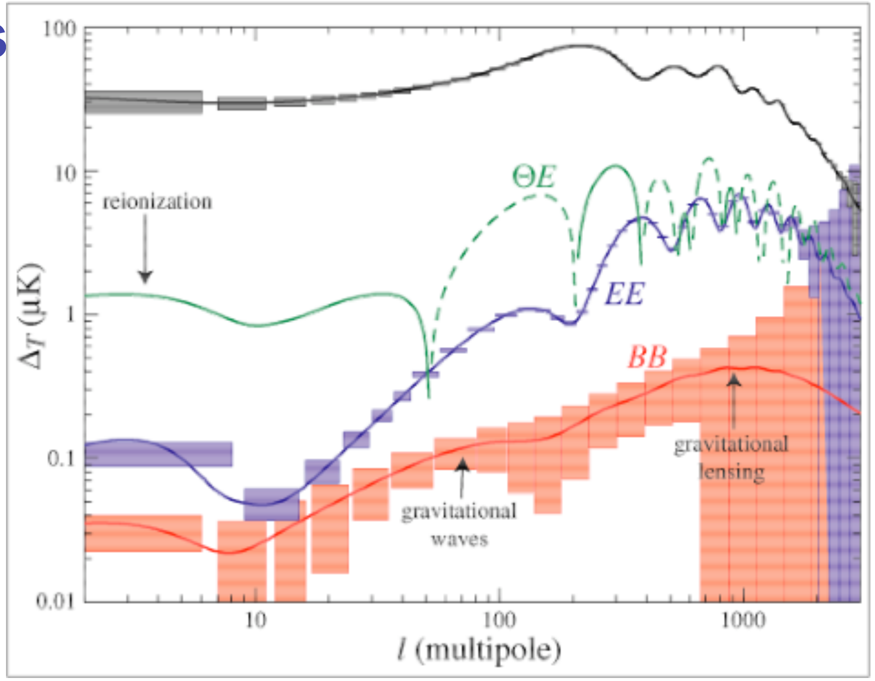
- **POSIX, MPI-IO, HDF5 (v1.6.5) offer very similar scalable performance**
- **parallelNetCDF (v1.0.2.pre): flat performance**

Outline

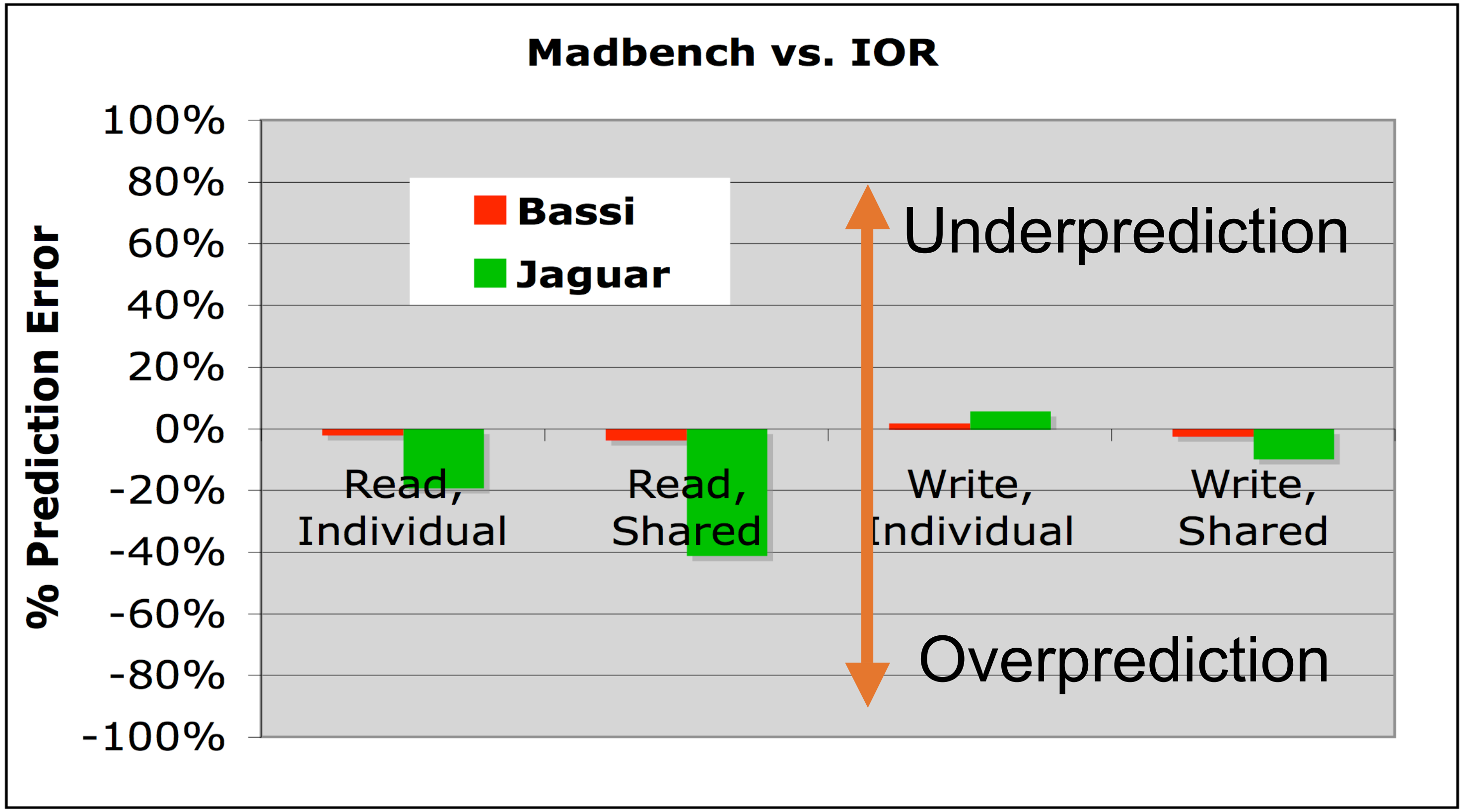
- Why IOR ?
- Using IOR to study system performance
- Using IOR to predict I/O performance for application

Madbench

- Astrophysics application, used to analyze the massive Cosmic Microwave Background datasets
- Important parameters related with IO:
  - Pixels: matrix size = pixels * pixels
  - Bins: number of matrices
- IO Behavior
  - Out-of-core app.
  - Matrix Write/Read
- Weak scaling problem
  - Pixels/Proc = 25K/16



I/O Performance Prediction for Madbench



- **IOR parameters: TransferSize=16MB, blockSize=64MB, segmentCount=1, P=64**

Summary

- Surveyed the I/O requirements of NERSC applications and selected IOR as the synthetic benchmark to study the I/O performance
- I/O Performance
  - Highly affected by file size, I/O transaction size, concurrency
  - Peaks at relatively low concurrency
  - The overhead of using HDF5 and MPI-IO is low, but pNETCDF is high
- IOR could be used effectively for I/O performance prediction for some applications