Parallel I/O

SciNet
www.scinet.utoronto.ca
University of Toronto
Toronto, Canada

February 27, 2013

Outline / Schedule

Disk I/O

Common Uses

- Checkpoint/Restart Files
- Data Analysis
- Data Organization
- Time accurate and/or Optimization Runs
- Batch and Data processing
- Database

Disk I/O

Common Bottlenecks

- Mechanical disks are slow!
- System call overhead (open, close, read, write)
- Shared file system (nfs, lustre, gpfs, etc)
- HPC systems typically designed for high bandwidth (GB/s) not IOPs
- Uncoordinated independent accesses

Disk Access Rates over Time

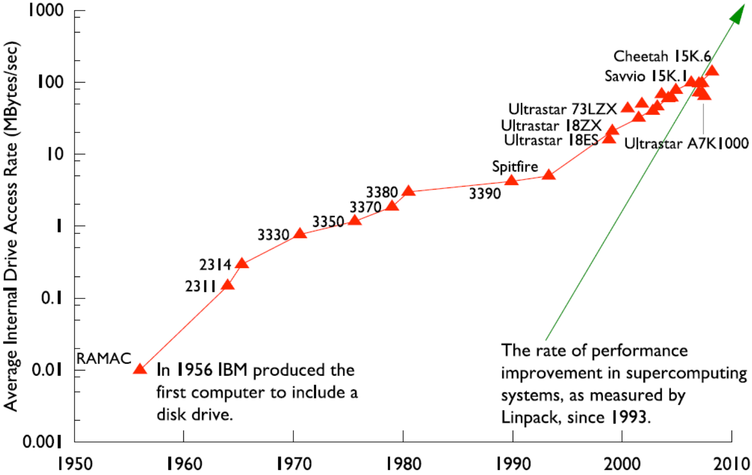


Figure by R. Ross, Argonne National Laboratory, CScADS09

Memory/Storage Latency

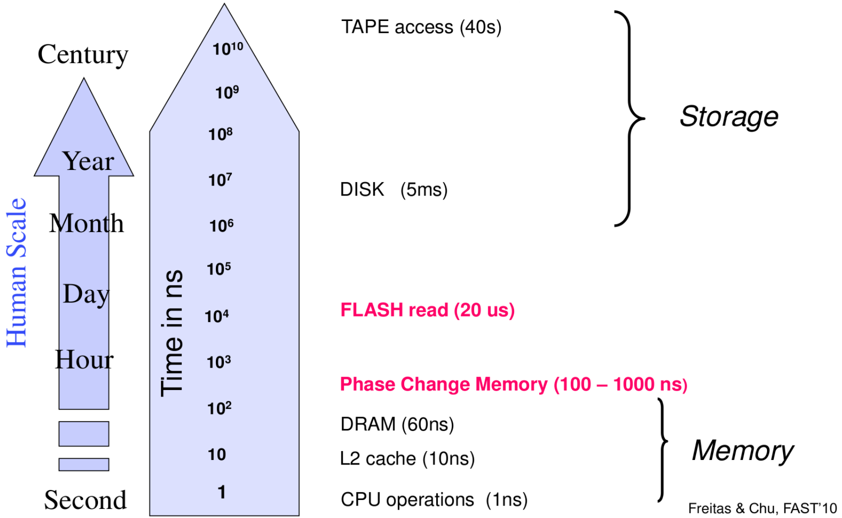


Figure by R. Freitas and L Chiu, IBM Almaden Labs, FAST'10

SciNet

Definitions

IOPs

Input/Output Operations Per Second (read,write,open,close,seek)

I/O Bandwidth

Quantity you read/write (think network bandwidth)

Comparisons

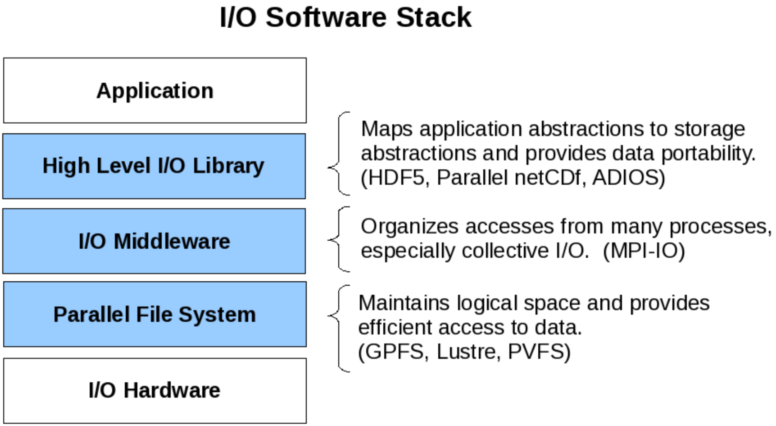
Device	Bandwidth (MB/s)	per-node	IOPs	per-node
SATA HDD	100	100	100	100
SSD HDD	250	250	4000	4000
SciNet	5000	1.25	30000	7.5

SciNet Filesystem

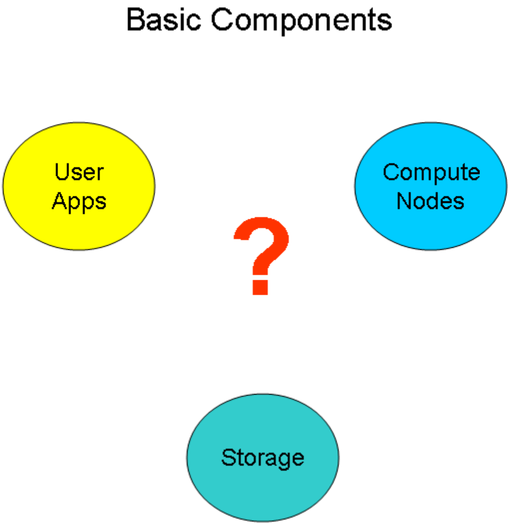
File System

- 1,790 1TB SATA disk drives, for a total of 1.4PB
- Two DCS9900 couplets, each delivering:
  - 4-5 GB/s read/write access (bandwidth)
  - 30,000 IOPs max (open, close, seek, . . .)
- Single *GPFS* file system on TCS and GPC
- I/O goes over Gb ethernet network on GPC (infiniband on TCS)
- File system is parallel!

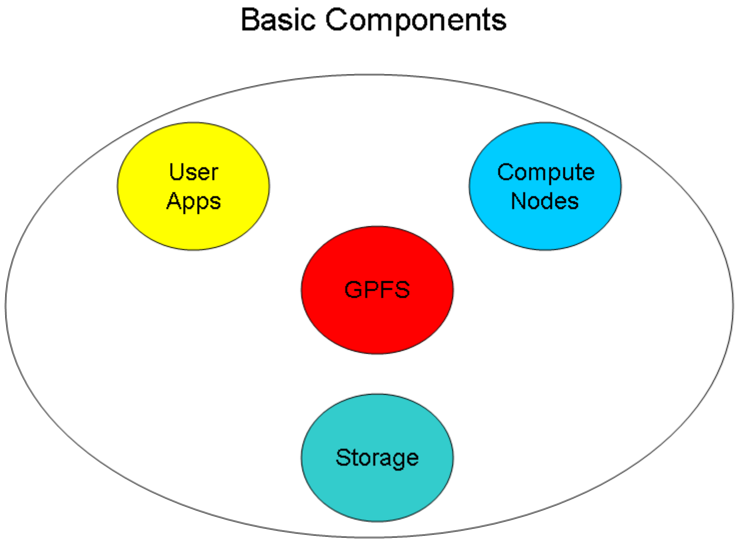
I/O Software Stack



Parallel File System

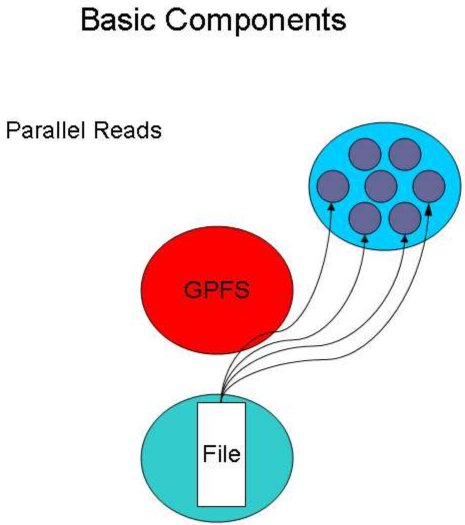


Parallel File System

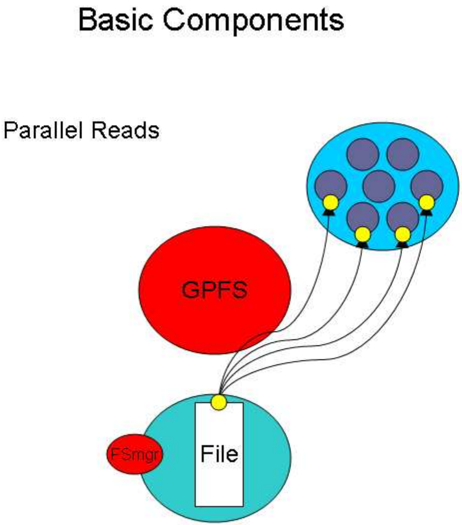


General Parallel File System

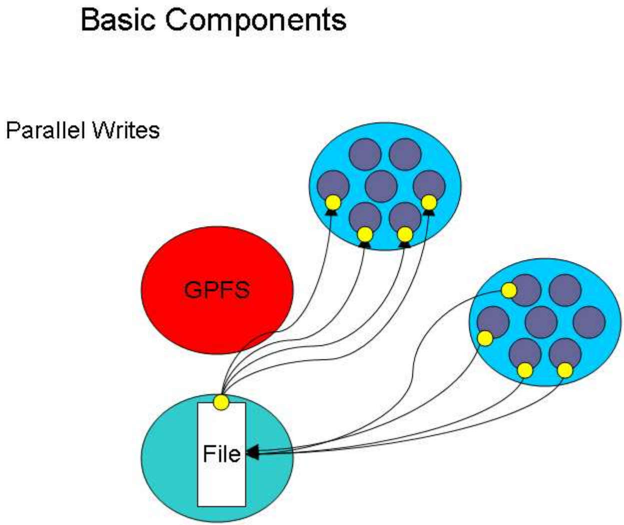
Parallel File System



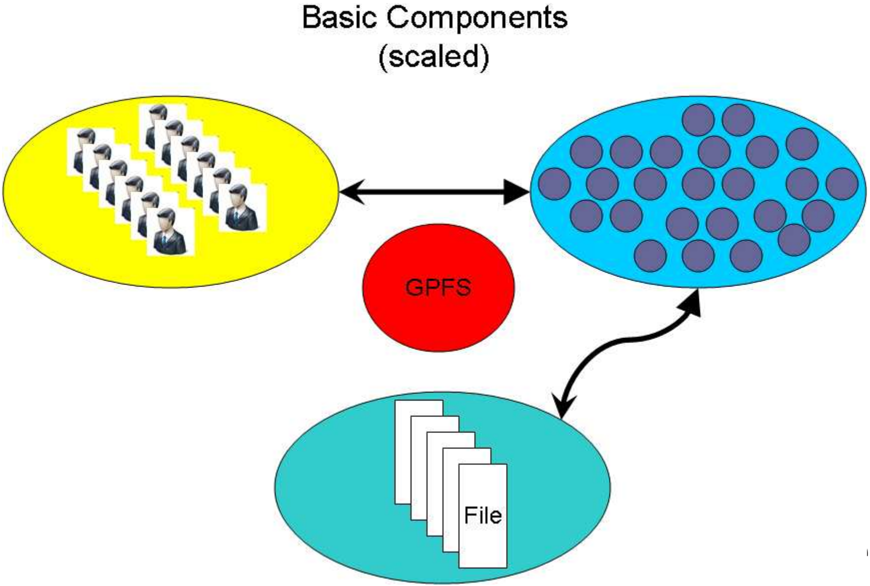
Parallel File System



Parallel File System

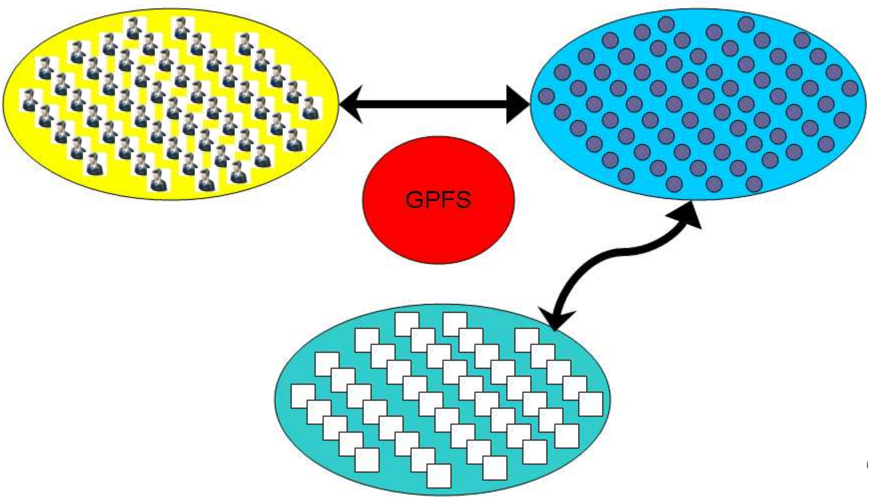


Parallel File System



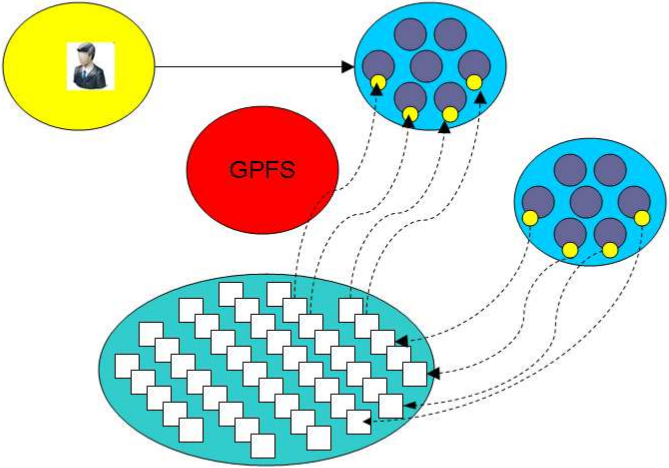
Parallel File System

How can we push the limit?



Parallel File System

How can we BREAK the limit?



Parallel File System

File Locks

Most parallel file systems use locks to manage concurrent file access

- Files are broken up into lock units
- Clients obtain locks on units that they will access before I/O occurs
- Enables caching on clients as well (as long as client has a lock, it knows its cached data is valid)
- Locks are reclaimed from clients when others desire access

Parallel File System

- Optimal for large shared files.
- Behaves poorly under many small reads and writes, high IOPs
- Your use of it affects everybody!
  (Different from case with CPU and RAM which are not shared.)
- How you read and write, your file format, the number of files in a directory, and how often you `ls`, affects every user!
- The file system is shared over the ethernet network on GPC: Hammering the file system can hurt process communications.
- File systems are not infinite!
  Bandwidth, metadata, IOPs, number of files, space, ...

Parallel File System

- 2 jobs doing simultaneous I/O can take much longer than twice a single job duration due to disk contention and directory locking.
- SciNet: 500+ users doing I/O from 4000 nodes. That's a lot of sharing and contention!

I/O Best Practices

Make a plan

- Make a plan for your data needs:
  - How much will you generate,
  - How much do you need to save,
  - And where will you keep it?
- Note that /scratch is temporary storage for 3 months or less.

Options?

1. Save on your departmental/local server/workstation (it is possible to transfer TBs per day on a gigabit link);
2. Apply for a project space allocation at next RAC call (but space is very limited);
3. Archive data using HPSS (tape)
4. Change storage format.

I/O Best Practices

Monitor and control usage

- Minimize use of filesystem commands like `ls` and `du`.
- Regularly check your disk usage using /scinet/gpc/bin/diskUsage.
- Warning signs which should prompt careful consideration:
  - More than 100,000 files in your space
  - Average file size less than 100 MB
- Monitor disk actions with `top` and `strace`

- RAM is always faster than disk; think about using ramdisk.
- Use `gzip` and `tar` to compress files to bundle many files into one
- Try gziping your *data* files. 30% not atypical!
- Delete files that are no longer needed
- Do "housekeeping" (`gzip`, `tar`, delete) regularly.

et

I/O Best Practices

Do's

- Write binary format files
  Faster I/O and less space than ASCII files.
- Use parallel I/O if writing from many nodes
- Maximize size of files. Large block I/O optimal!
- Minimize number of files. Makes filesystem more responsive!

Don'ts

- Don't write lots of ASCII files. Lazy, slow, and wastes space!
- Don't write many hundreds of files in a 1 directory. (File Locks)
- Don't write many small files ($<$ 10MB).
  System is optimized for large-block I/O.

Data Management

Formats

- ASCII
- Binary
- MetaData (XML)
- Databases
- Standard Library's (HDF5,NetCDF)

ASCII

American Standard Code for Information Interchange

Pros

- Human Readable
- Portable (architecture independent)

Cons

- Inefficient Storage
- Expensive for Read/Write (conversions)

Native Binary

100100100

Pros

- Efficient Storage (256 x floats @4bytes takes 1024 bytes)
- Efficient Read/Write (native)

Cons

- Have to know the format to read
- Portability (Endianness)

ASCII vs. binary

Writing 128M doubles

Format	/scratch (GPCS)	/dev/shm (RAM)	/tmp (disk)
ASCII	173s	174s	260s
Binary	6s	1s	20s

Syntax

Format	C	FORTRAN
ASCII	`fprintf()`	open(6,file='test',form='formatted') write(6,*)
Binary	`fwrite()`	open(6,file='test',form='unformatted') write(6)

SciNet

Metadata

What is Metadata?

Data about Data

- File System: size, location, date, owner, etc.
- App Data: File format, version, iteration, etc.

Example: XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<slice_data>
  <format>UTF1000</format>
  <verstion>6.8</version>
  <img src="slice1_2010.img" alt='Slice 1 of Data'/>
  <date> January 15th, 2010 </date>
  <loc> 47 23.516 -122 02.625 </loc>
</slice_data>
```

Databases

Beyond flat files

- Very powerful and flexible storage approach
- Data organization and analysis can be greatly simplified
- Enhanced performance over seek/sort depending on usage
- Open Source Software
  - SQLite (serverless)
  - PostgreSQL
  - mySQL

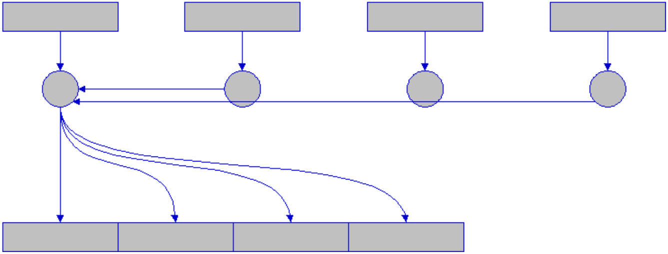
“Standard” Formats

- CGNS (CFD General Notation System)
- IGES/STEP (CAD Geometry)
- HDF5 (Hierarchical Data Format)
- NetCDF (Network Common Data Format)
- disciplineX version

Common Ways of Doing Parallel I/O

Sequential I/O (only proc 0 Writes/Reads)

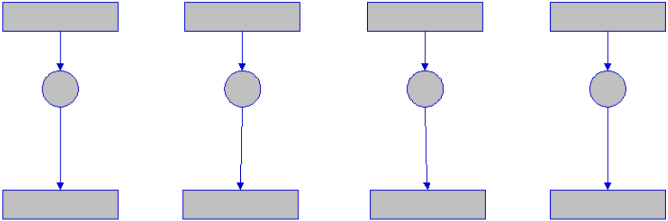
- Pro
  - Trivially simple for small I/O
  - Some I/O libraries not parallel
- Con
  - Bandwidth limited by rate one client can sustain
  - May not have enough memory on node to hold all data
  - Won't scale (built in bottleneck)



Common Ways of Doing Parallel I/O

N files for N Processes

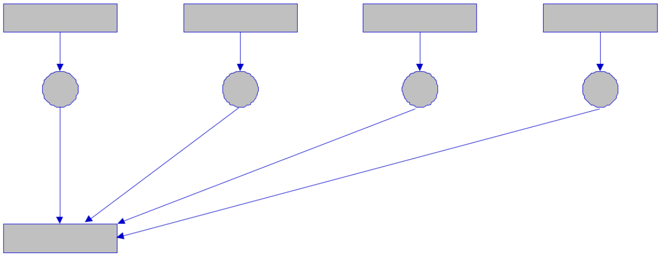
- Pro
  - No interprocess communication or coordination necessary
  - Possibly better scaling than single sequential I/O
- Con
  - As process counts increase, lots of (small) files, won't scale
  - Data often must be post-processed into one file
  - Uncoordinated I/O may swamp file system (File LOCKS!)



Common Ways of Doing Parallel I/O

All Processes Access One File

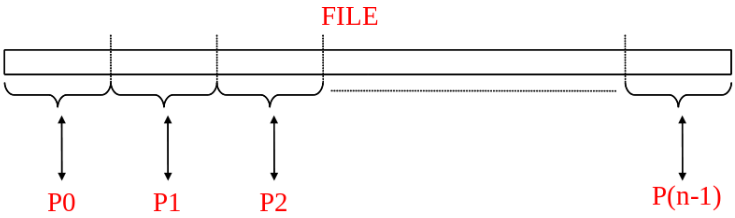
- Pro
  - Only one file
  - Data can be stored canonically, avoiding post-processing
  - Will scale if done correctly
- Con
  - Uncoordinated I/O WILL swamp file system (File LOCKS!)
  - Requires more design and thought



Parallel I/O

What is Parallel I/O?

Multiple processes of a parallel program accessing data (reading or writing) from a common file.



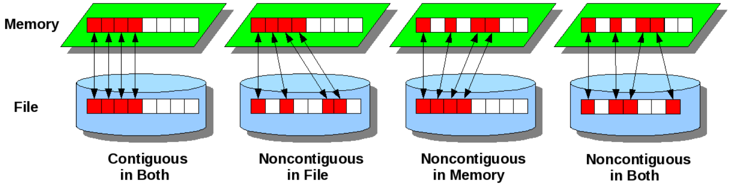
Parallel I/O

Why Parallel I/O?

- Non-parallel I/O is simple but:
  - Poor performance (single process writes to one file)
  - Awkward and not interoperable with other tools (each process writes a separate file)
- Parallel I/O
  - Higher performance through collective and contiguous I/O
  - Single file (visualization, data management, storage, etc)
  - Works with file system not against it

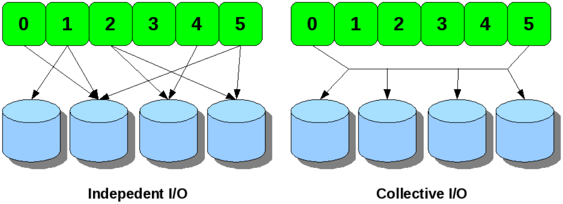
SciNet

Contiguous and Noncontiguous I/O



- Contiguous I/O move from a single memory block into a single file block
- Noncontiguous I/O has three forms:
  - Noncontiguous in memory, in file, or in both
- Structured data leads naturally to noncontiguous I/O (e.g. block decomposition)
- Describing noncontiguous accesses with a single operation passes more knowledge to I/O system

Independent and Collective I/O



- Independent I/O operations specify only what a single process will do
  - calls obscure relationships between I/O on other processes
- Many applications have phases of computation and I/O
  - During I/O phases, all processes read/write data
  - We can say they are collectively accessing storage
- Collective I/O is coordinated access to storage by a group of processes
  - functions are called by all processes participating in I/O
  - Allows file system to know more about access as a whole, more optimization in lower software layers, better performance

et

Parallel I/O

Available Approaches

- MPI-IO: MPI-2 Language Standard
- HDF (Hierarchical Data Format)
- NetCDF (Network Common Data Format)
- Adaptable IO System (ADIOS)
  - Actively developed (OLCF,SandiaNL,GeorgiaTech) and used on largest HPC systems (Jaguar,Blue Gene/P)
  - External to the code XML file describing the various elements
  - Uses MPI-IO, can work with HDF/NetCDF

MPI-IO

HDF5/NETCDF