

HPC Best Practices

Ontario Summer School on High Performance Computing

Scott Northrup
SciNet HPC Consortium
Compute Canada

May 10th, 2013

- 1 Work-flow
- 2 Batch Computing
- 3 Data Management
 - File Systems and I/O
 - Data Management
 - Parallel I/O
- 4 Development Overview
 - Compilers
 - Libraries
- 5 Performance Analysis

Contributing Material

- HPC Best Practices - G. Baolai, SHARCNET
- The Parallel File System and I/O - R. van Zon, SciNet
- Monitoring Job Efficiently - R. van Zon, SciNet
- Profiling and Tuning - L. J. Dursi, SciNet
- Tuning MPI - L. J. Dursi, SciNet

Typical Simulation/Analysis Work-flow

- pre-process (grid creation, partitioning)
- solve/analysis
- postprocess (data-mining, generate plots)

Typical Simulation/Analysis Work-flow

- pre-process (grid creation, partitioning)
- solve/analysis
- postpones (data-mining, generate plots)

Automate

- learn and use script languages (bash, python)
- use scheduler efficiently (job size, dependencies)
- add data management into work-flow from beginning

Batch Computing

SciNet systems are batch compute clusters

- Computing by submitting **batch jobs** to the **scheduler**.
- When you submit a job, it gets placed in a **queue**.
- Job priority is based on **allocation** and **fairshare**.
- When sufficient nodes are free to execute a job, it starts the job on the appropriate compute nodes.
- Jobs remain **'idle'** until resources become available.
- Jobs can be temporarily **'blocked'** if you submit too much.

Components

Torque: Resource manager providing control over batch jobs and distributed compute nodes.

Moab: A policy-based job scheduler and event engine that enables utility-based computing for clusters.

Fairshare: Mechanism using past utilization for prioritization.

Preparation

- Compile
- Test on devel node
- Determine resources
- Write job script

`llsubmit`
`qsub`

Preparation

- Compile
- Test on devel node
- Determine resources
- Write job script

```
llsubmit  
qsub
```

Monitor

- Job queued?
- When will it run?
- What else is queued?
- Efficiency?

```
qstat -f  
checkjob  
showstart  
showbf  
showq
```

Preparation

- Compile
- Test on devel node
- Determine resources
- Write job script

lsubmit
qsub

Monitor

- Job queued?
- When will it run?
- What else is queued?
- Efficiency?

qstat -f
checkjob
showstart
showbf
showq

Control

- Cancel job
- Ssh to nodes
- Interactive jobs
- Debug queue

canceljob
top
qsub -I
qsub -q debug

Preparation

- Compile
- Test on devel node
- Determine resources
- Write job script

llsubmit
qsub

Monitor

- Job queued?
- When will it run?
- What else is queued?
- Efficiency?

qstat -f
checkjob
showstart
showbf
showq

Control

- Cancel job
- Ssh to nodes
- Interactive jobs
- Debug queue

canceljob
top
qsub -I
qsub -q debug

Reports

- Check .o/.e
jobname.{o,e}
 - usage stats on Scinet web portal
- showstats -u

Monitoring not-yet-running jobs

qstat and checkjob

- Show torque status right away on GPC: `qstat`
- Show moab status (better): `checkjob jobid`
- See more details of the job: `checkjob -v jobid`
(e.g., why is my job blocked?)

showq

- See all the jobs in the queue: `showq` (from gpc or tcs)
- See your jobs in the queue: `showq -u user`

showstart and showbf

- Estimate when a job may start: `showbf`
- Estimate when a queued job may start: `showstart jobid`
- **Estimates only!**

Monitoring running jobs

checkjob

- `checkjob jobid`

showq

- `showq -r -u $USER`

ssh

- `ssh node` (node name from checkjob)
- `top`: shows process state, memory and cpu usage

Job stdout/stderr files

- `{jobname}.o{jobid}`
- `{jobname}.e{jobid}`

Top example

```
gpc-f103n084-$ ssh gpc-f109n001
gpc-f109n001-$ top
```

```
top - 21:56:45 up 5:56, 1 user, load average: 5.55, 1.73, 0.88
Tasks: 234 total, 1 running, 233 sleeping, 0 stopped, 0 zombie
Cpu(s): 11.4%us, 36.2%sy, 0.0%ni, 52.2%id, 0.0%wa, 0.0%hi, 0.2%si, 0.0%st
Mem: 16410900k total, 1542768k used, 14868132k free, 0k buffers
Swap: 0k total, 0k used, 0k free, 294628k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	P	COMMAND
22479	ljdursi	18	0	108m	4816	3212	S	98.5	0.0	1:04.81	6	gameoflife
22480	ljdursi	18	0	108m	4856	3260	S	98.5	0.0	1:04.85	13	gameoflife
22482	ljdursi	18	0	108m	4868	3276	S	98.5	0.0	1:04.83	2	gameoflife
22483	ljdursi	18	0	108m	4868	3276	S	98.5	0.0	1:04.82	8	gameoflife
22484	ljdursi	18	0	108m	4832	3232	S	98.5	0.0	1:04.80	9	gameoflife
22481	ljdursi	18	0	108m	4856	3256	S	98.2	0.0	1:04.81	3	gameoflife
22485	ljdursi	18	0	108m	4808	3208	S	98.2	0.0	1:04.80	4	gameoflife
22478	ljdursi	18	0	117m	5724	3268	D	69.6	0.0	0:46.07	15	gameoflife
8042	root	0	-20	2235m	1.1g	16m	S	2.3	6.8	0:30.59	8	mmfsd
10735	root	15	0	3707	452	372	S	1.3	0.0	0:16.80	0	cat

Top example

```
gpc-f103n084-$ ssh gpc-f109n001
gpc-f109n001-$ top
```

```
top - 21:56:45 up 5:56, 1 user, load average: 5.55, 1.73, 0.88
Tasks: 234 total, 1 running, 233 sleeping, 0 stopped, 0 zombie
Cpu(s): 11.4%us, 36.2%sy, 0.0%ni, 52.2%id, 0.0%wa, 0.0%hi, 0.2%si, 0.0%st
Mem: 16410900k total, 1542768k used, 14868132k free, 0k buffers
Swap: 0k total, 0k used, 0k free, 294628k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	P	COMMAND
22479	ljdursi	18	0	108m	4816	3212	S	98.5	0.0	1:04.81	6	gameoflife
22480	ljdursi	18	0	108m	4856	3260	S	98.5	0.0	1:04.85	13	gameoflife
22482	ljdursi	18	0	108m	4868	3276	S	98.5	0.0	1:04.83	2	gameoflife
22483	ljdursi	18	0	108m	4868	3276	S	98.5	0.0	1:04.82	8	gameoflife
22484	ljdursi	18	0	108m	4832	3232	S	98.5	0.0	1:04.80	9	gameoflife
22481	ljdursi	18	0	108m	4856	3256	S	98.2	0.0	1:04.81	3	gameoflife
22485	ljdursi	18	0	108m	4808	3208	S	98.2	0.0	1:04.80	4	gameoflife
22478	ljdursi	18	0	117m	5724	3268	D	69.6	0.0	0:46.07	15	gameoflife
8042	root	0	-20	2235m	1.1g	16m	S	2.3	6.8	0:30.59	0	mmrtd
10735	root	15	0	3702	452	372	S	1.2	0.0	0:16.80	0	cat

canceljob

- If you spot a mistake: `canceljob jobid`

qsub for interactive and debug jobs

- `-I`:
 - Interactive
 - After qsub, waits for jobs to start.
 - Usually combined with:
- `-q debug`:
 - Debug queue has 10 nodes reserved for short jobs.
 - You can get 1 node for 2 hours, but also
 - 8 nodes, for half an hour.

Job output/error files (*.e / *.o)

```
-----  
Begin PBS Prologue Tue Sep 14 17:14:48 EDT 2010 1284498888  
Job ID:      3053514.gpc-sched  
Username:    ljdursi  
Group:      scinet  
Nodes:      gpc-f134n009 gpc-f134n010 gpc-f134n011 gpc-f134n012  
gpc-f134n043 gpc-f134n044 gpc-f134n045 gpc-f134n046 gpc-f134n047 gpc-f134n048  
[...]  
End PBS Prologue Tue Sep 14 17:14:50 EDT 2010 1284498890  
-----  
[ Your job's output here... ]  
-----  
Begin PBS Epilogue Tue Sep 14 17:36:07 EDT 2010 1284500167  
Job ID:      3053514.gpc-sched  
Username:    ljdursi  
Group:      scinet  
Job Name:    fft_8192_procs_2048  
Session:     18758  
Limits:     neednodes=256:ib:ppn=8,nodes=256:ib:ppn=8,walltime=01:00:00  
Resources:   cput=713:42:30,mem=3463854672kb,vmem=3759656372kb,walltime=00:21:07  
Queue:      batch_ib  
Account:  
Nodes:      gpc-f134n009 gpc-f134n010 gpc-f134n011 gpc-f134n012 gpc-f134n043  
[...]  
Killing leftovers...  
gpc-f141n054:  killing gpc-f141n054 12412  
  
End PBS Epilogue Tue Sep 14 17:36:09 EDT 2010 1284500169  
-----
```

Data Management

Common Uses

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

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

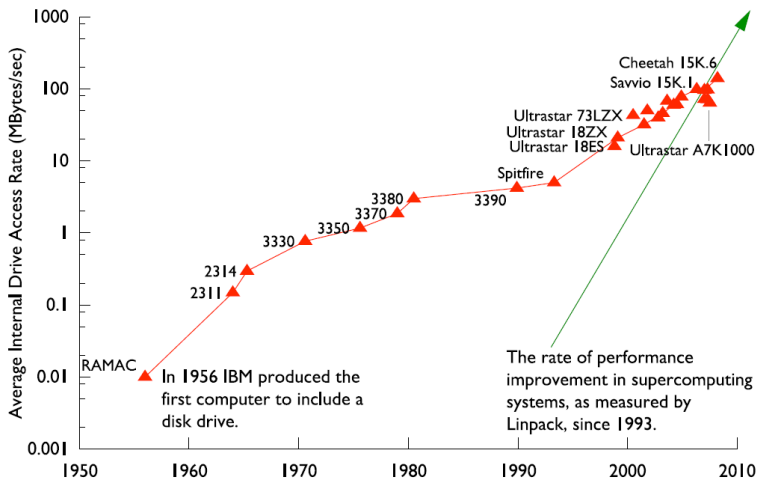


Figure by R. Ross, Argonne National Laboratory, CScADS09

Memory/Storage Latency

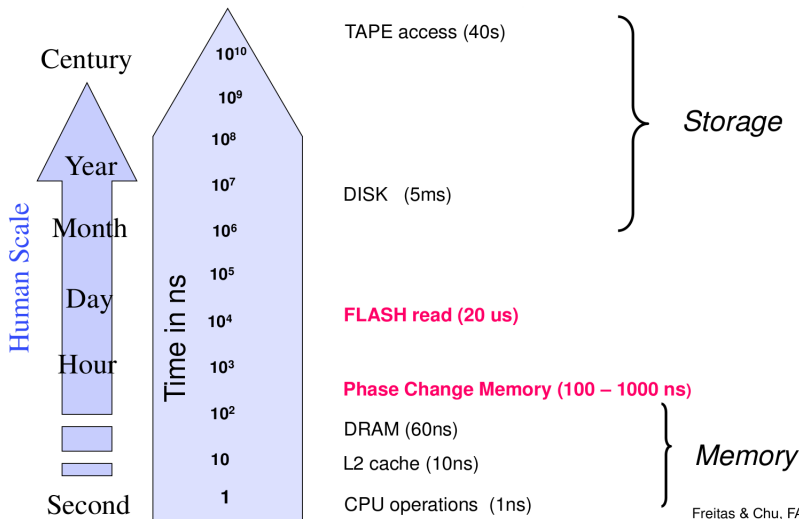


Figure by R. Freitas and L Chiu, IBM Almaden Labs, FAST'10

Freitas & Chu, FAST'10

Definitions

IOPs

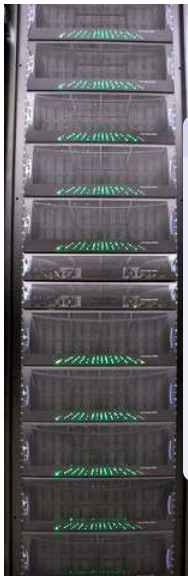
Input/Output Operations Per Second (read,write,open,close,seek)

I/O Bandwidth

Quantity you read/write (think network bandwidth)

Comparisons

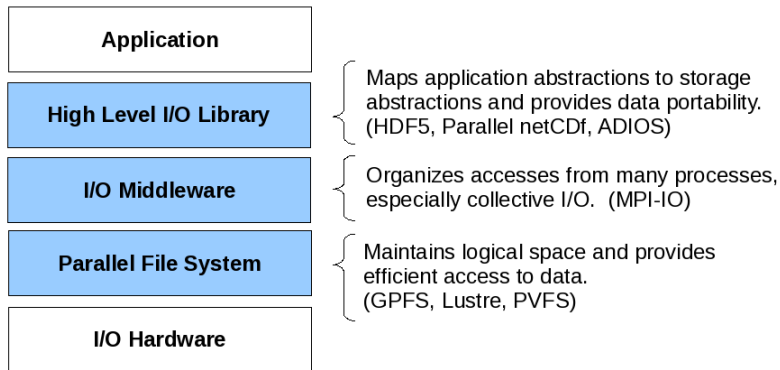
Device	Bandwidth (MB/s)	per-node	IOPs	per-node
SATA HDD	100	100	100	100
SSD HDD	250	250	4000	4000
SciNet	5000	1.25	30000	7.5



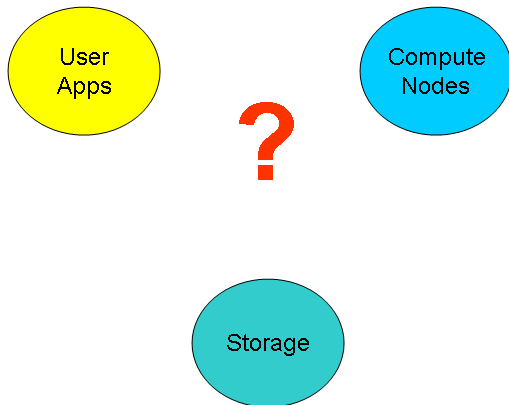
File System

- 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 infiniband (as of April 2012)
- File system is **parallel!**

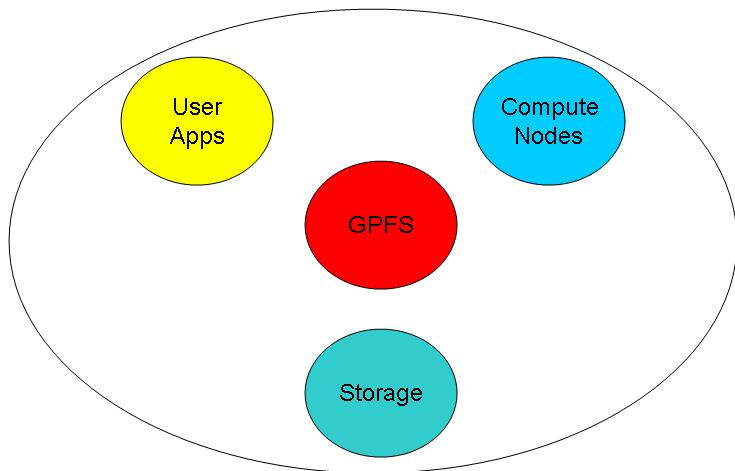
I/O Software Stack



Basic Components



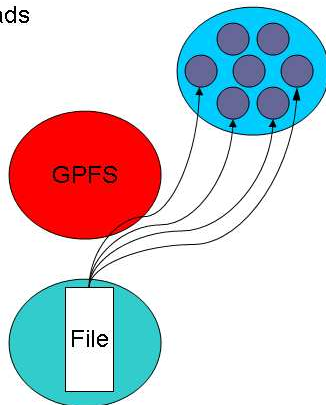
Basic Components



General Parallel File System

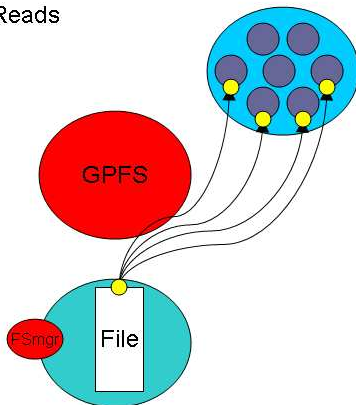
Basic Components

Parallel Reads



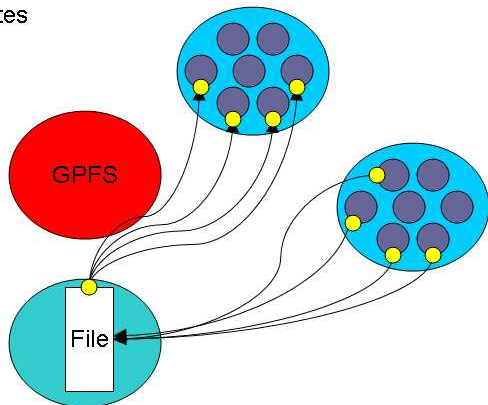
Basic Components

Parallel Reads



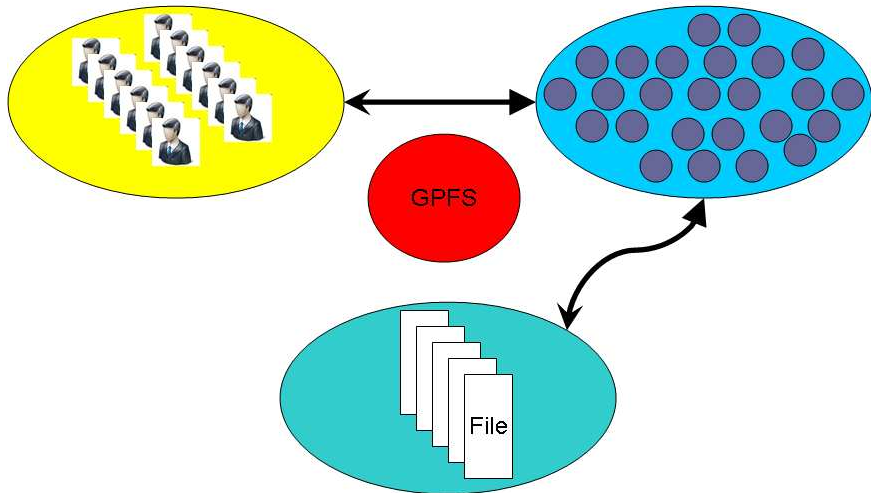
Basic Components

Parallel Writes



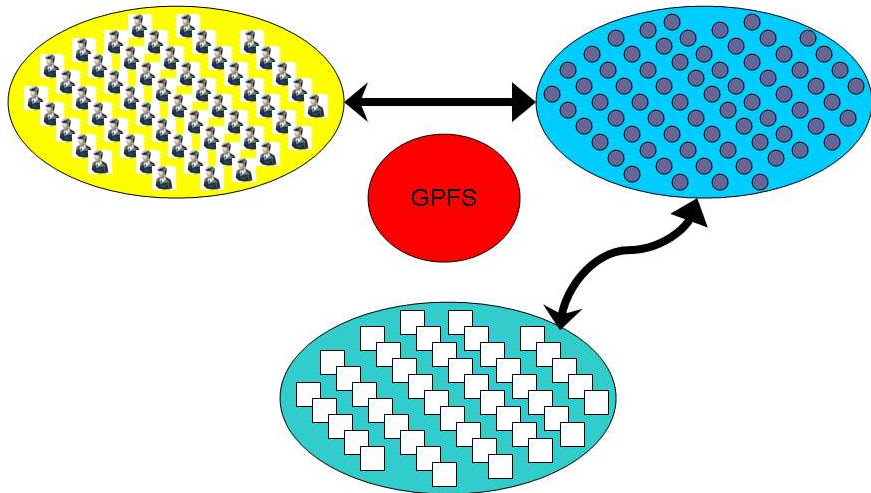
Parallel File System

Basic Components (scaled)

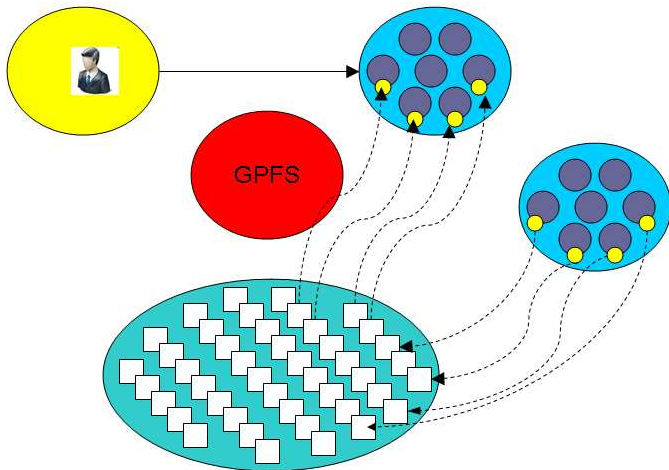


Parallel File System

How can we push the limit?



How can we BREAK the limit?



File Locks

Most parallel file systems use locks to manage concurrent file access

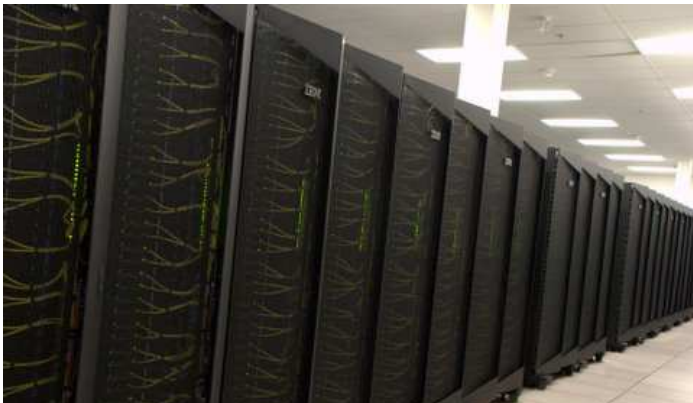
- Files are broken up into lock units
- Clients obtain locks on units that they will access before I/O occurs
- Enables caching on clients as well (as long as client has a lock, it knows its cached data is valid)
- Locks are reclaimed from clients when others desire access

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 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!



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/HPSS allocation at next RAC call (but space is very limited);
- 3 Change storage format.

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 gzipping your *data* files. 30% not atypical!
 - Delete files that are no longer needed
 - Do "housekeeping" (`gzip`, `tar`, `delete`) regularly.

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.

Formats

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

American Standard Code for Information Interchange

Pros

- Human Readable
- Portable (architecture independent)

Cons

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

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	<code>fprintf()</code>	<code>open(6,file='test',form='formatted')</code> <code>write(6,*)</code>
Binary	<code>fwrite()</code>	<code>open(6,file='test',form='unformatted')</code> <code>write(6)</code>

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>
  
  <date> January 15th, 2010 </date>
  <loc> 47 23.516 -122 02.625 </loc>
</slice_data>
```

Beyond flat files

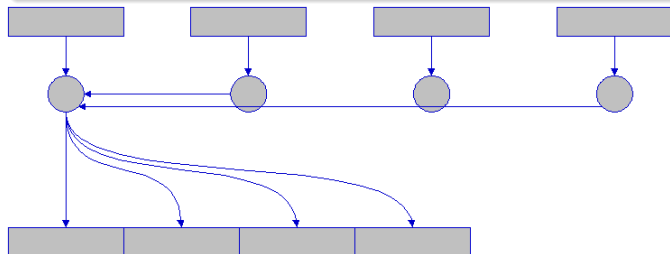
- Very powerful and flexible storage approach
- Data organization and analysis can be greatly simplified
- Enhanced performance over seek/sort depending on usage
- Open Source Software
 - SQLite (serverless)
 - PostgreSQL
 - MySQL

“Standard” Formats

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

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

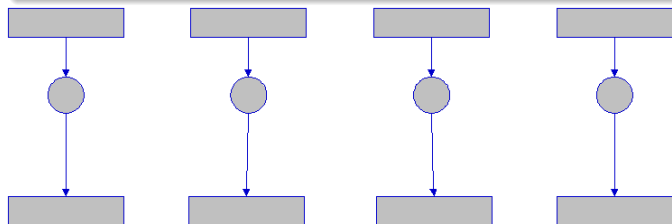
- Pro
 - Trivially simple for small I/O
 - Some I/O libraries not parallel
- Con
 - Bandwidth limited by rate one client can sustain
 - May not have enough memory on node to hold all data
 - Won't scale (built in bottleneck)



Common Ways of Doing Parallel I/O

N files for N Processes

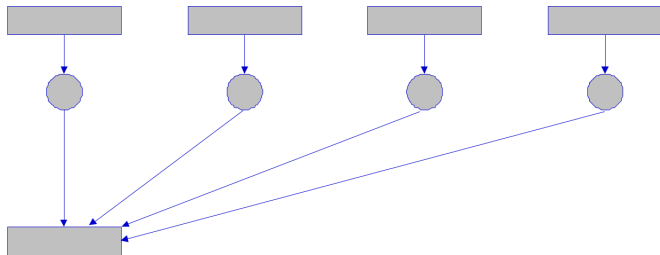
- Pro
 - No interprocess communication or coordination necessary
 - Possibly better scaling than single sequential I/O
- Con
 - As process counts increase, lots of (small) files, won't scale
 - Data often must be post-processed into one file
 - Uncoordinated I/O may swamp file system (File LOCKS!)



Common Ways of Doing Parallel I/O

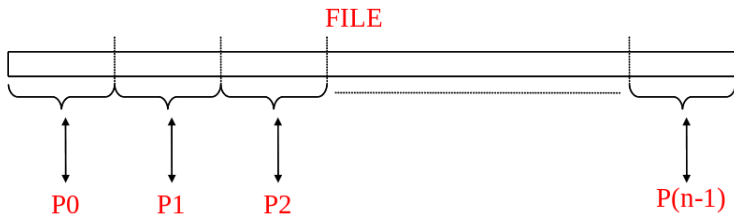
All Processes Access One File

- 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



What is Parallel I/O?

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



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

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

Software Development

Tools of the Trade

- Editors/IDE

Tools of the Trade

- Editors/IDE
- Version Control

Tools of the Trade

- Editors/IDE
- Version Control
- Build System (make)

Tools of the Trade

- Editors/IDE
- Version Control
- Build System (make)
- Compilers

Tools of the Trade

- Editors/IDE
- Version Control
- Build System (make)
- Compilers
- Libraries

Tools of the Trade

- Editors/IDE
- Version Control
- Build System (make)
- Compilers
- Libraries
- Debuggers (gdb, idb, Allinea DDT)

Tools of the Trade

- Editors/IDE
- Version Control
- Build System (make)
- Compilers
- Libraries
- Debuggers (gdb, idb, Allinea DDT)
- Memory (valgrind)

Tools of the Trade

- Editors/IDE
- Version Control
- Build System (make)
- Compilers
- Libraries
- Debuggers (gdb, idb, Allinea DDT)
- Memory (valgrind)
- I/O (strace)

Tools of the Trade

- Editors/IDE
- Version Control
- Build System (make)
- Compilers
- Libraries
- Debuggers (gdb, idb, Allinea DDT)
- Memory (valgrind)
- I/O (strace)
- Performance (gprof, Scalasa, IPM)

What is it?

- A tool for managing changes in a set of files.

What is it?

- A tool for managing changes in a set of files.
- Figuring out who broke what where and when.

What is it?

- A tool for managing changes in a set of files.
- Figuring out who broke what where and when.

Why Do it?

- Collaboration
- Organization
- Track Changes
- Faster Development
- Reduce Errors

Collaboration

With others and yourself

Questions

Collaboration

With others and yourself

Questions

- What if two (or more) people want to edit the same file at the same time?

Collaboration

With others and yourself

Questions

- What if two (or more) people want to edit the same file at the same time?
- What if you work on SciNet and on your own computer?

Collaboration

With others and yourself

Questions

- What if two (or more) people want to edit the same file at the same time?
- What if you work on SciNet and on your own computer?

Answers

Collaboration

With others and yourself

Questions

- What if two (or more) people want to edit the same file at the same time?
- What if you work on SciNet and on your own computer?

Answers

- Option 1: make them take turns
 - But then only one person can be working at any time
 - And how do you enforce the rule?

Collaboration

With others and yourself

Questions

- What if two (or more) people want to edit the same file at the same time?
- What if you work on SciNet and on your own computer?

Answers

- Option 1: make them take turns
 - But then only one person can be working at any time
 - And how do you enforce the rule?
- Option 2: patch up differences afterwards
 - Requires a lot of re-working
 - Stuff always gets lost

Collaboration

With others and yourself

Questions

- What if two (or more) people want to edit the same file at the same time?
- What if you work on SciNet and on your own computer?

Answers

- Option 1: make them take turns
 - But then only one person can be working at any time
 - And how do you enforce the rule?
- Option 2: patch up differences afterwards
 - Requires a lot of re-working
 - Stuff always gets lost
- Option 3: **Version Control**

Organize and Track Changes

Question

Question

- Want to undo changes to a file
 - Start work, realize it's the wrong approach, want to get back to starting point
 - Like "undo" in an editor...
...but keep the whole history of every file, forever

Question

- Want to undo changes to a file
 - Start work, realize it's the wrong approach, want to get back to starting point
 - Like "undo" in an editor...
...but keep the whole history of every file, forever
- Also want to be able to see who changed what, when
 - The best way to find out how something works is often to ask the person who wrote it

Organize and Track Changes

Question

- Want to undo changes to a file
 - Start work, realize it's the wrong approach, want to get back to starting point
 - Like "undo" in an editor...
...but keep the whole history of every file, forever
- Also want to be able to see who changed what, when
 - The best way to find out how something works is often to ask the person who wrote it

Answer

- **Version Control**

What Software to Use

Software

- Open Source
 - Subversion, CVS, RCS
 - Git, Mercurial, Bazaar
- Commercial
 - Perforce, ClearCase

available as modules on SciNet

Subversion (svn)

- Centralized Version Control
- Replaces CVS
- Lots of web and GUI integration
- Users: GCC, KDE, FreeBSD

Git

- Distributed Version Control
- *nix command line driven design model
- advanced features `git-stash`, `git-rebase`, `git-cherry-pick`
- Users: Linux kernel, GNOME, Wine, X.org

Compiler Flags and Optimizations

Numerical Libraries

Numerical Methods

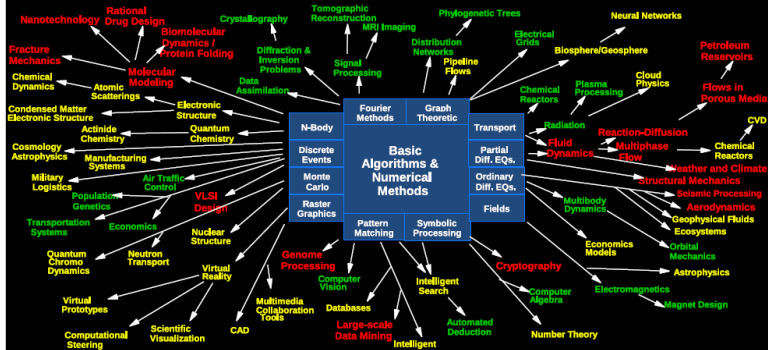
- Linear algebra
- Nonlinear equations
- Optimization
- Interpolation/Approximation
- Integration and differentiation
- Solving ODEs
- Solving PDEs
- FFT
- Random numbers and stochastic simulations
- Special functions

Top Ten Algorithms for Science (Jack Dongarra, 2000)

1. Metropolis Algorithm for Monte Carlo
2. Simplex Method for Linear Programming
3. Krylov Subspace Iteration Methods
4. The Decompositional Approach to Matrix Computations
5. The Fortran Optimizing Compiler
6. QR Algorithm for Computing Eigenvalues
7. Quicksort Algorithm for Sorting
8. Fast Fourier Transform
9. Integer Relation Detection
10. Fast Multipole Method

Numerical Algorithms

Good Better Best



Argonne National Laboratory GBB

Numerical Libraries

- BLAS (gotoblas, ATLAS)
- LAPACK (ESSL, MKL, ACML)
- ScaLAPACK
- GSL (GNU Scientific Library)
- FFTW
- PETS_c
- TAO
- IMSL
- NAG

Numerical Libraries

- BLAS (gotoblas, ATLAS)
- LAPACK (ESSL, MKL, ACML)
- ScaLAPACK
- GSL (GNU Scientific Library)
- FFTW
- PETSc
- TAO
- IMSL
- NAG

Don't re-invent the wheel!

Performance & Profiling