Triton file systems
- an introduction

File systems

- Motivation & basic concepts
- Storage locations
- Basic flow of IO
- Do's and Don'ts
- Exercises

File systems: Motivation

- Case #1: I'll just run my code from here

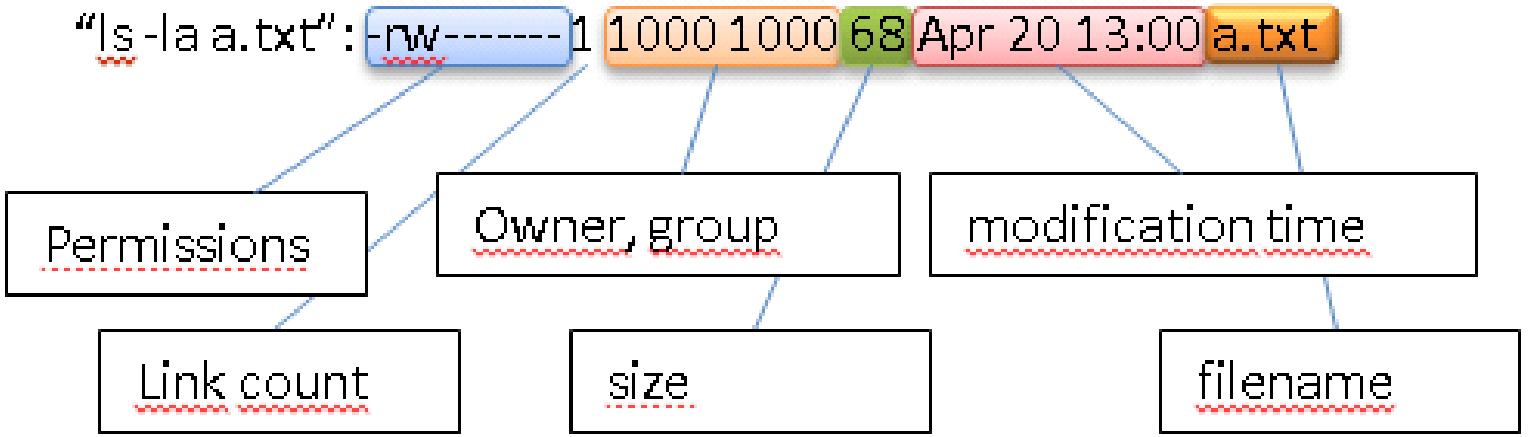
  … 15 min later other users cannot access their files!
- Case #2: My directory listing takes 10 min while on my laptop it takes 3s. Am I doing something wrong?
- Case #3: My files are organized in such a fashion that I can't show the important ones to my supervisor when he/she comes to visit.

  With IO/storage you can (easily) go horribly wrong

File systems: Basic concepts

What is a file?

- File = metadata + block data
- Accessing block data: **cat a.txt**
- Showing some metadata: **ls -l a.txt**



File systems: Basic concepts

What is a file system?

- Data store than manages multiple files
- Ext4 used in Triton/linux
- Can be local (physical disks on a server) or shared over network
- Some basic operations
  - `ls, cp, scp, rsync, rm, vi, less, …`

File systems: Basic concepts

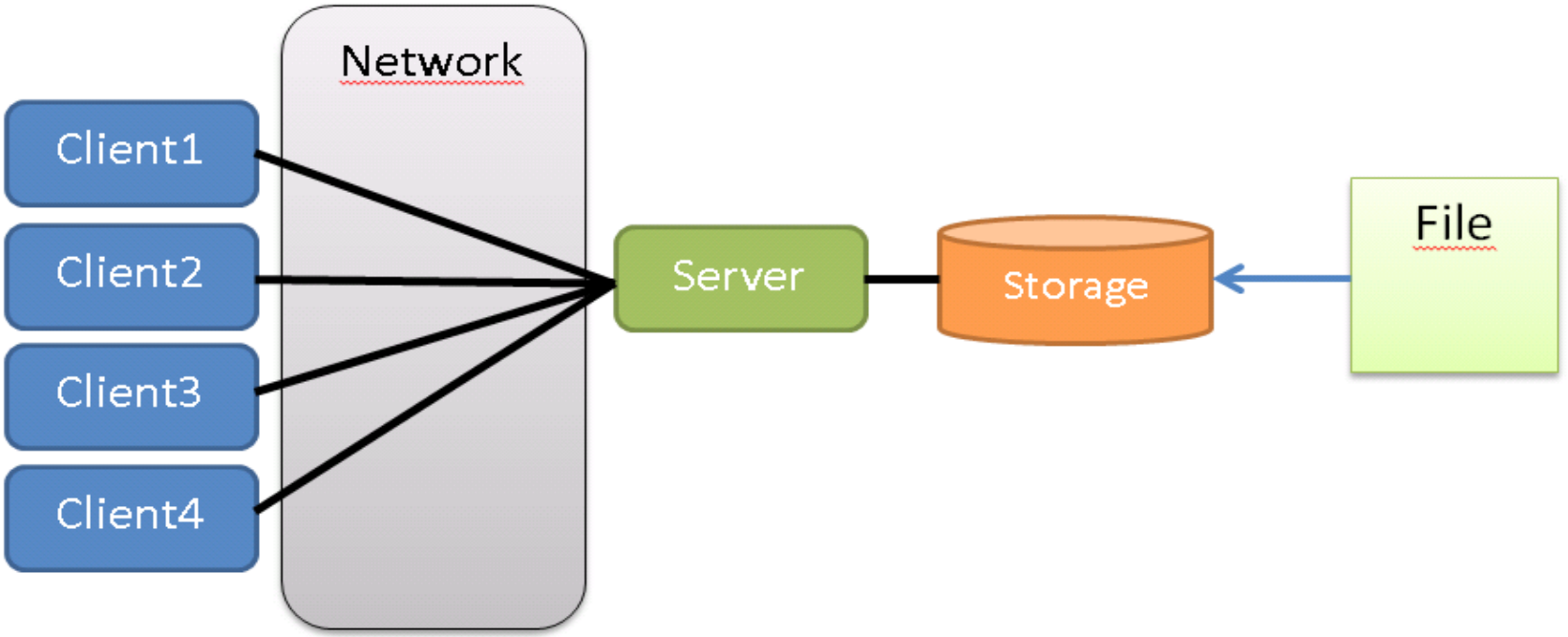
Access and limits

- Quota enforced, mainly for groups
- Linux filesystem permissions
- Some department export filesystem to department workstations (Becs, ICS)
- Rsync, scp, SSHFS are always the options when transferring files to/from the cluster

How storage is organized in Triton

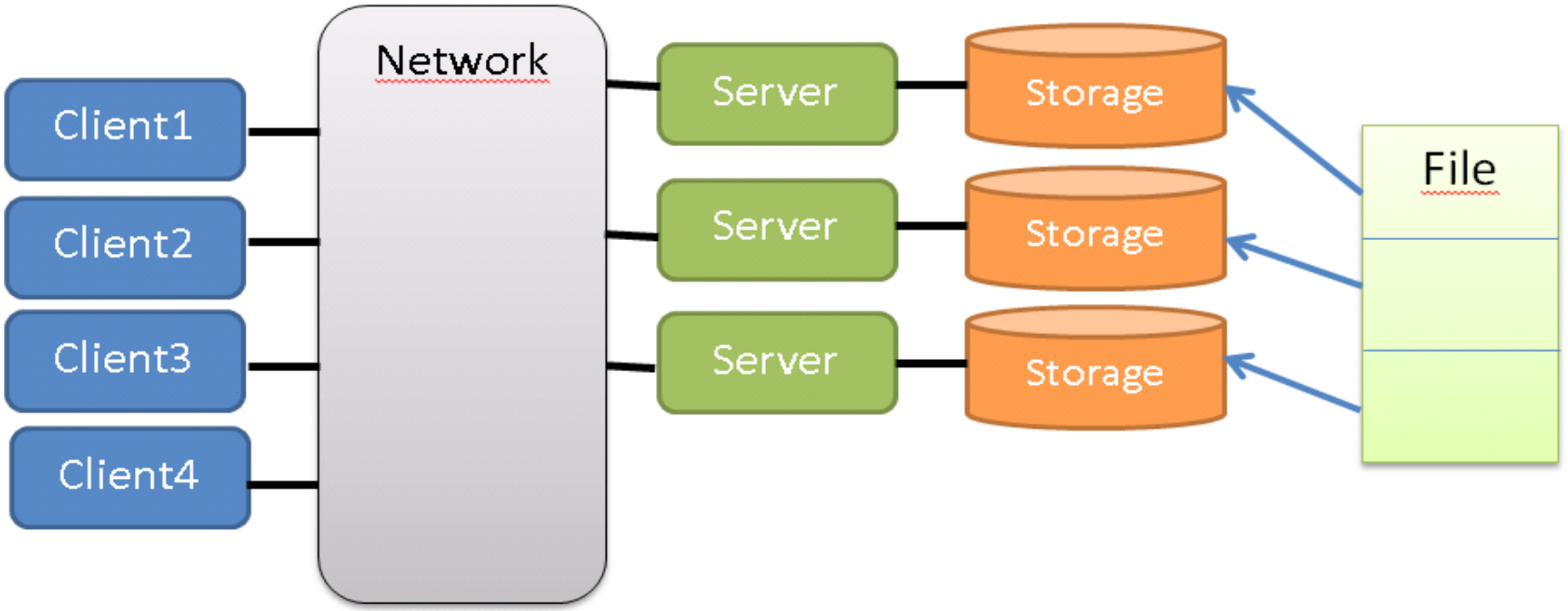
File systems: NFS

- Network filesystem
  - Server/storage SPOF and bottleneck



File systems: Lustre

- Many servers & storage elements
  - Redundancy and scaling
  - Single file can be split between multiple servers!



File systems: Network storage

User's Home folder (NFS filesystem)

- /home/<username> -directory
- Common for every computational node
- Meant for scripts etc
- Nightly backup, 1GB quota

Work/Scratch (Lustre filesystem)

- /triton/<department>/work/<username>
- Common for every computational node
- Meant for input/output Data
- Quota varies per project
- No backups (reliable system with RAID)
- Has an optimized find function 'lfs find'. In the autumn, when system is updated even more dedicated tools.

File systems: Local storage

Storage local to compute node

- /local -directory
- Best for calculation time storage
- Copy relevant data after computation, will be deleted on re-install
- $TMPDIR variable, directory automatically deleted after job. Clean yourself if using something else.

Ramdisk for fast IO operations

- Special location, similar to /local
- /ramdisk -directory (20GB)
- Use case: job spends most of its time doing file operations on millions of small files.

File systems: Summary

Location	Type	Usage	Size/quota
/home	NFS	Home dir	1 Gb
/local	local	Local scratch	~800Gb (varies)
/triton	Lustre	Work/Scratch	200Gb (varies, even several TB's)
/ramdisk	Ramdisk	Local scratch	20GB

File systems: Meters of performance

- Stream I/O and random IOPS
  - Stream measures the speed of reading large sequential data from system
  - IOPS measure random small reads to the system – number of metadata/block data accesses
  - To measure your own application, profiling or internal timers needed
  - Rough estimate can be aquired from /proc/<pid>/io or by using strace

File systems: Meters of performance

- Total numbers

Device	IOPS	Stream
Sata disk (7.2k)	50-100	50 MB/s
SSD disk	3000-10 000	500 MB/s
Ramdisk	40 000	5000 MB/s
Triton NFS	300	300 MB/s
Triton Lustre	20 000	4000 MB/s

- With 200 concurrent jobs using storage...

Device	IOPS	Stream
Sata disk (7.2k)	50-100	50 MB/s
SSD disk	N/a	N/a
Triton NFS	1.5	1.5 MB/s
Triton Lustre	100	20 MB/s

- DON'T run jobs from HOME! (NFS)

File systems: Advanced Lustre

- By default striping is turned off
  - "`lfs getstripe <dir>`" shows striping
  - "`lfs setstripe -c N <dir>`" stripe over N targets, -1 means all targets
  - "`lfs setstripe -d <dir>`" revert to default
- Use with care. Useful for HUGE files (>100GB) or parallel I/O from multiple compute nodes (MPI-I/O).
- Real numbers from single client (16 MB IO blocksize for 17 GB):

Striping	File size	Stream Mb/s
Off (1)	17 GB	214
2	17 GB	393
4	17 GB	557
max	17 GB	508
max	11 MB	55
max	200 KB	10

File systems: Know your program

Questions that you should ask yourself

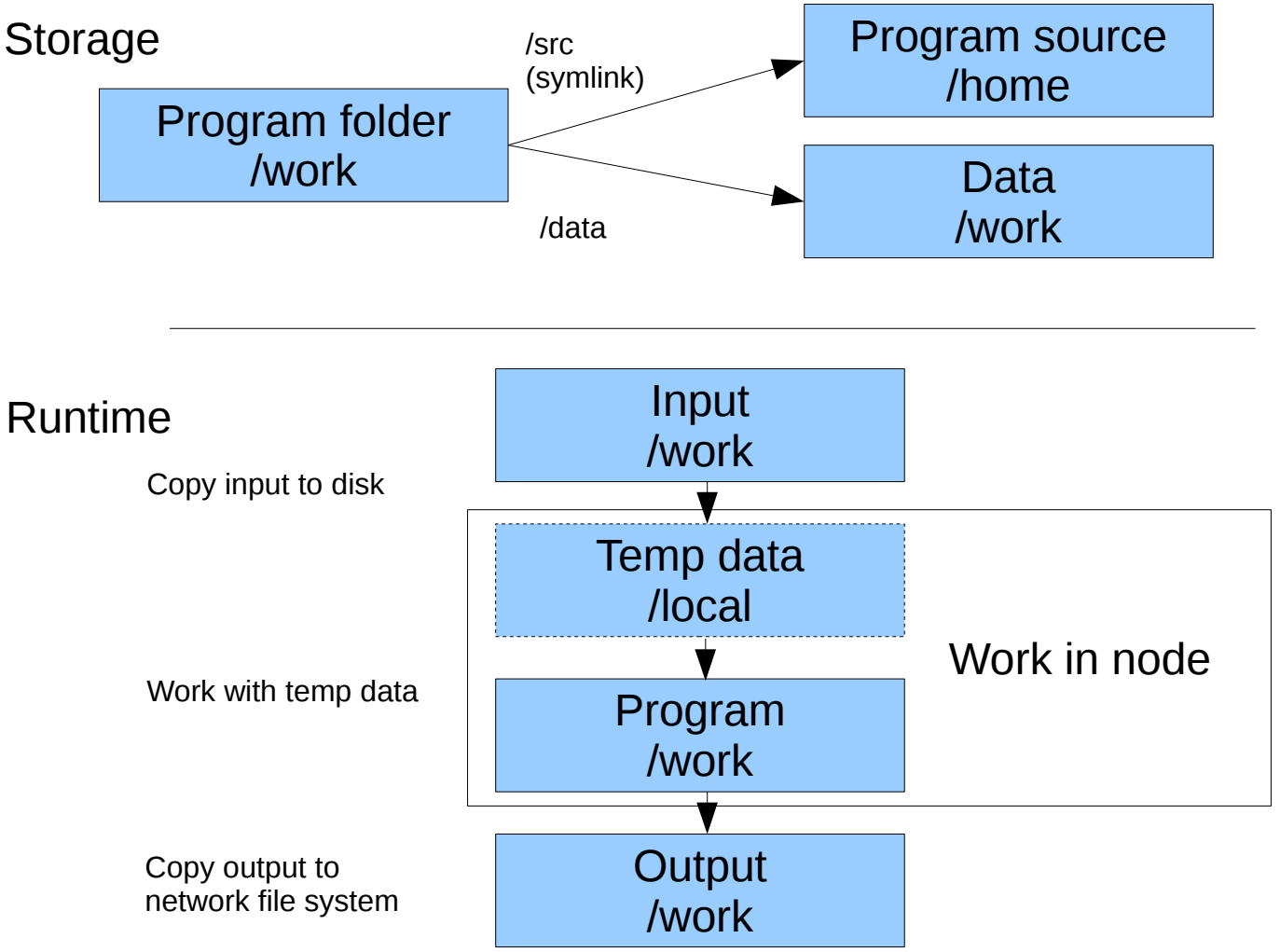
1. How long does it take to analyze a single dataset?
2. How big is a single input dataset (byte size/file number)?
3. Does your code read/write data that is not useful?
4. How often does the code access the files?
   Once or constantly?
5. Can my usage cause problem to others?

Why these are important?

1. Calculation is faster than disk access.
   Thus calculation time should be big compared to the data transfer time.
2. If dataset consists of small fragments grouping can lower metadata access.
   If dataset consists of large files bad read/write buffering can cause loads of IO operations.
3. This should be self-evident. IO and storage are resources just like CPU clock cycles or RAM usage.
4. Constant access slows down the file system. Once is better. If constant access is required, using /local drive will divert the load from network drives.
5. With great power comes great responsibility.

Do's and Don'ts

File systems: Workflow suggestion

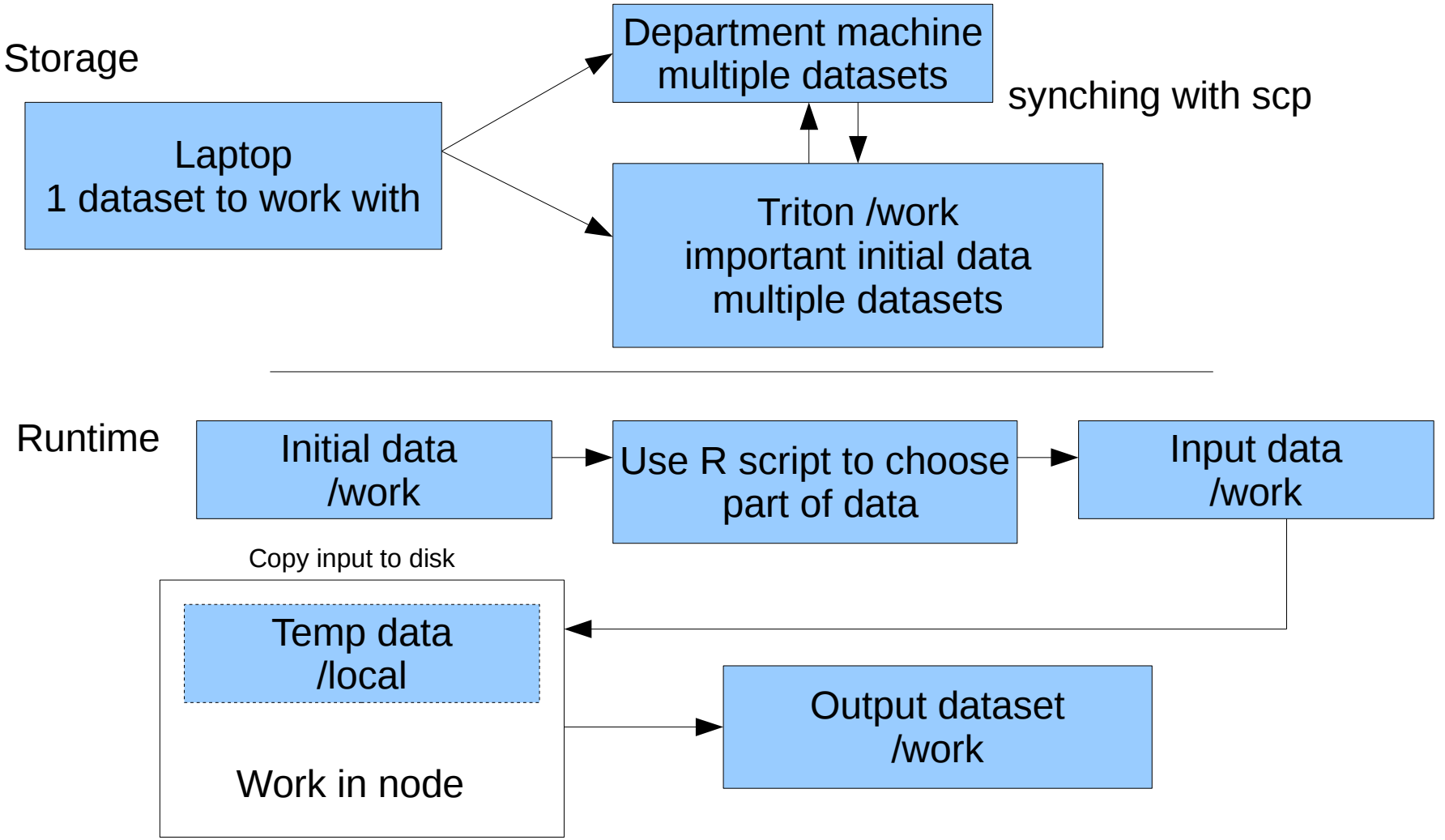


File systems: Do's and Don'ts

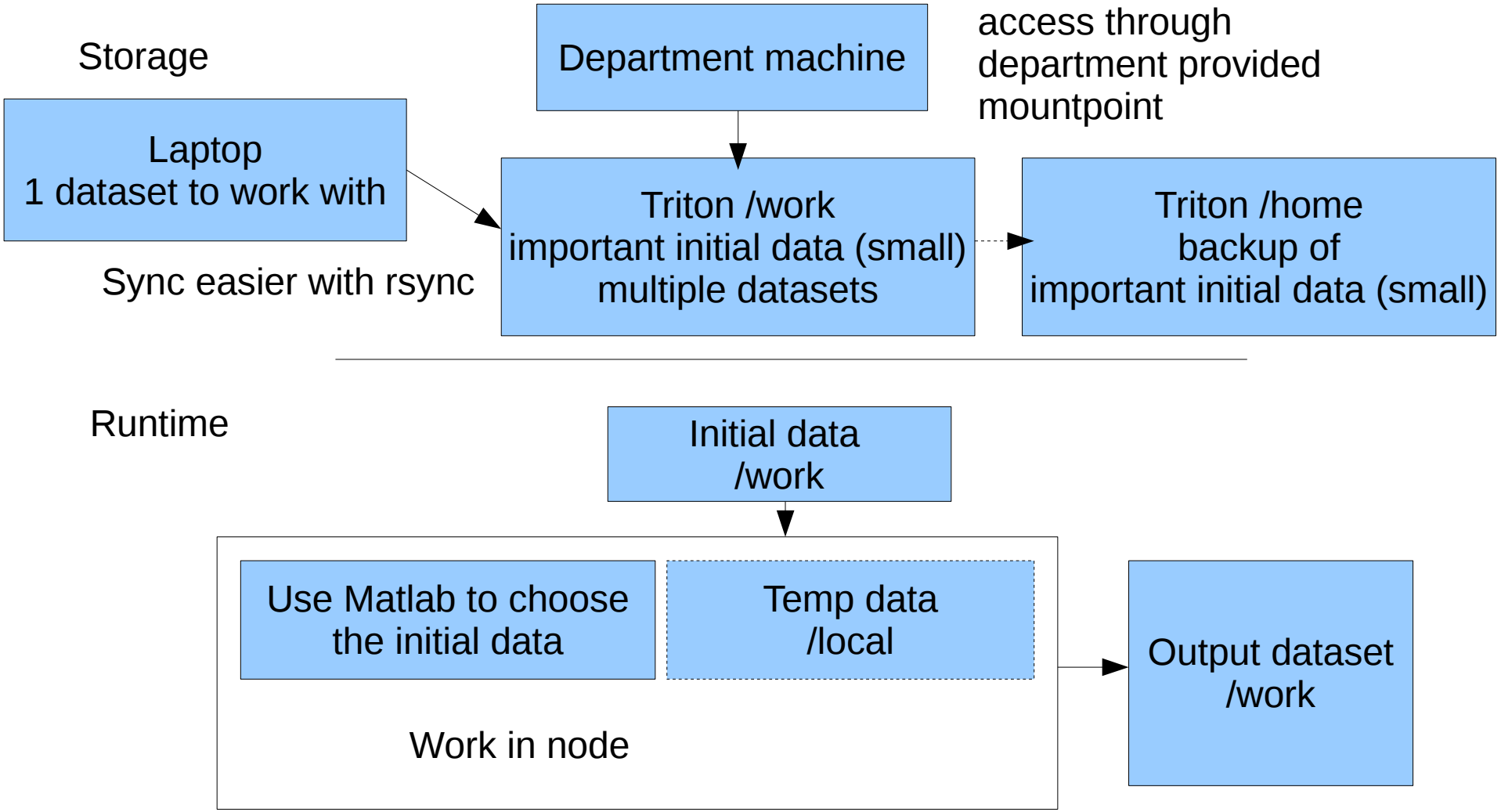
Use version control and keep your data tidy

- Pick one version control system: svn, hg, git
  Only for source code
- If you find stuff from your folders you didn't know existed, think about cleaning up. Using consistent naming or data format like HDF you can see your data sets with a glance (no need to use find etc.)
- Don't use cp as a backup to guard your code/data from your mistakes.

File systems: Real use case



File systems: Real use case



File systems: Do's and Don'ts

Lots of small files (+10k, <1MB)

- Well, bad starting point already in general. Though, sometimes no way to improve (e.g. legacy code)
  - /ramdisk or /local: Best place for these
  - Lustre: Not the best place. With many users local disk provides more IOPS and Stream in general
  - NFS (Home): Very Bad idea, do not run calculation from Home
- **The very best approach:** modify you code. Large file(s) instead of many small (e.g. HDF5). Or even no-files-at-all. Sometimes IO due to unnecessary checkpointing.

File systems: Do's and Don'ts

Databases (sqlite)

- These can generate a lot of small random reads (=IOPS)
  - /ramdisk or /local: Best place for these
  - Lustre: Not the best place. With many users local disk provides more IOPS and Stream in general
  - NFS (Home): very Bad idea

File systems: Do's and Don'ts

“**ls**” vs “**ls -la**”

- ls in a directory with 1000 files
  - Simple ls is only a few IOPS
- ls -la in a directory with 1000 files
  - Local fs: 1000+ IOPS (stat() each file!)
  - NFS: a bit more overhead
  - Lustre (striping off) 2000 IOPS (+rpcs)
  - Lustre (striping on) 31000 IOPS! (+rpcs)

    => Whole Lustre stuck for a while for everyone
- Use “ls -la” and variant (ls --color) ONLY when needed

Exercise: File systems

15 minutes to proceed, use wiki/google to solve

Simple file system operations

- Use git and clone code repository in /triton/scip/lustre to your work directory.
- Use 'strace -c' to 'ls <file>' and 'ls -l <file>'. Compare output with eg. grep/diff.
- Run create_iodata.sh to create sample data. Compare 'strace -c' of 'lfs find <directory>' and 'find <directory>' searches to the 'data' directory.
- Submit iotest.sh with sbatch. What does the output mean?
- Try to convert the code to use $TMPDIR. Once you're sure it works, change 'ls' to 'ls -l'. Compare the results.
- Convert the code to use tar/zip/gzip/bzip2. Can you profile the load caused to network drive?

File systems: Best practices

- When unsure what is the best approach
  - Check above Do's and Don'ts
  - Google?
  - Ask your local Triton support person
  - tracker.triton.aalto.fi
  - Ask your supervisor
  - Trial-and-error (profile it)

File systems: Further topics

These are not covered here. Ask/Google if you want to learn more.

- Using Lustre striping (briefly mentioned)
- HDF5 for small files
- Benchmarking, what is the share of IO of a job
- MPI-IO
- Hadoop

Questions or comments regarding Triton file systems?