UPPMAVX file systems

Douglas G. Scofield

UPPMAVX Data Management Seminar
2016-03-18
What we'll cover

• What is a computer 'disk'?
• What is a 'file system'?
• What is a 'file'?
• What is high-performance storage?
• What do the following mean at UPPMAX?
  ▪ backup storage
  ▪ nobackup
  ▪ private
  ▪ glob
  ▪ scratch
  ▪ node-local

• When do I use what?
What is a computer 'disk'?

- Storage organised into **sectors**
  - all sectors of fixed, standard size
  - each has unique hardware address
- Can be virtualised (RAID)
  - many disks appear as one
  - redundancy ('hot-swap')
  - performance

https://en.wikipedia.org/wiki/RAID
What is a 'file system'?

• A method for organising a disk's sectors
  – ext3, HFS+, NTFS, FAT32, many others
  – some are organised to 'hide' the physical disk

• **Blocks** are (usually) larger than disk sectors
  – ... also equally-sized and uniquely addressable

• Some blocks are very special (**where is the OS?**)

• **Everything** is built out of blocks
  – In storage, a file can be no smaller than a block
  – Large files are split into separate blocks
  – Files are read by blocks

https://en.wikipedia.org/wiki/File_system
**FIGURE 1** RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN ALLOCATED FILE

**ls -li** shows inode numbers
Consequences of file systems

• File contents are separate from file metadata
• A file cannot be located without its metadata
• Files need not be contiguous on disk
• Read/write speeds may be subject to substantial physical constraints
High-performance storage

• Accounts for physical constraints (RAID)
• Intensive file and metadata management
• One file may be spread across many disks
• ... often 2+ copies

• Always tradeoffs

'Streaming' read/write speed vs Speed of random access

https://commons.wikimedia.org/wiki/File:HuaweiRH2288HV2.JPG
When high-performance storage, isn't

- Intensive management requires rapid decisions
- If many reads/writes, many decisions
- As storage fills, more time required per decision
  - Many users + full storage = perfect storm
- Random access requires more time per decision
- **Disks** >90% full likely to experience bottlenecks

```
milou-b: /proj/b2011141/nobackup $ df -kh .
Filesystem  Size  Used  Avail  Use% Mounted on
pica1-v2:/pica/v2   9.0T  7.2T  1.9T  80% /pica/v2
milou-b: /proj/b2011141/nobackup $ df -kh | grep v2
apus1-v2:/apus/v2   208T  144T  65T  69% /apus/v2
pica2-v20:/pica2/v20  50T  30T  20T  60% /pica/v20
pica1-v2:/pica/v2  228T  161T  68T  71% /pica/v2
```
Good to know...

• Moving a file (mv) on the same disk requires a simple change to the metadata

• Copying (cp, rsync) or a move between disks requires copying file contents

• A hard link (ln) creates a copy of the metadata (new inode) but not the contents

• A symbolic link (ln –s) is a simple entry with an alternate name
  – not a copy of the metadata nor the contents
UPPMAX Backup Storage

- Files in `/proj/project-name/` that are not under `nobackup/` and `INBOX/`

```bash
milou-b: /proj $ ll /proj/b2011141
lrwxrwxrwx 1 root root 17 Jan 21 09:38 /proj/b2011141 -> /pica/v5/b2011141
milou-b: /proj $ ll /proj/b2011141/
total 146336
-rw-r--r-- 1 root b2011141 149518407 Feb 16 2015 CORRUPTED_FILE_LIST.20150216
-rw-r--r-- 1 root b2011141 2048 Mar 4 2014 INBOX
-rw-r--r-- 2 biyue b2011141 2048 Jun 18 2015 keepData
lrwxrwxrwx 1 root b2011141 23 May 29 2015 nobackup -> /proj/nobackup/b2011141
drwxrwxrwx 73 nath b2011141 14336 Feb 17 13:21 pipeline
drwxrwxrwx 2 root b2011141 2048 Jun 26 2015 private
drwxrwxrwx 2 douglas b2011141 2048 Jul 2 2013 scripts
drwxrwxrwx 9 nath b2011141 2048 Dec 21 2013 sequenceData
drwxr-xr-x 4 root b2011141 2048 Sep 8 2011 stesstore
drwxrwxrwx 108 douglas b2011141 18432 Mar 1 12:26 tools
```

- Files in your home directory
Snapshots

- Each directory in backup storage has a hidden `.snapshot/` folder containing dated backup copies.

```
milou-b: /proj/b2011141 $ cd scripts
milou-b: /proj/b2011141/scripts $ ll
  total 128
  -rwxrwx--- 1 douglas b2011141 316 Nov 20  2012 base-qual-check.pl
  -rwxrwx--- 1 douglas b2011141 447 Nov 20  2012 base-qual-check.sh
  -rwxr-x--- 1 douglas b2011141 286 Jul  2  2013 launch.sh
  -rwxr-x--- 1 douglas b2011141 410 Jul  2  2013 nlaunch.sh
milou-b: /proj/b2011141/scripts $ cd .snapshot
milou-b: /proj/b2011141/scripts/.snapshot $ ll
  total 160
  drwxrws--- 2 douglas b2011141 2048 Jul  2  2013 2016-03-13_0530+0100.Daily
  drwxrws--- 2 douglas b2011141 2048 Jul  2  2013 2016-03-14_0530+0100.Daily
  drwxrws--- 2 douglas b2011141 2048 Jul  2  2013 2016-03-15_0530+0100.Daily
  drwxrws--- 2 douglas b2011141 2048 Jul  2  2013 2016-03-16_0530+0100.Daily
  drwxrws--- 2 douglas b2011141 2048 Jul  2  2013 2016-03-17_0530+0100.Daily
```
Backup storage is expensive

• Snapshots reduce available storage
• Additional computation by the storage system
• Within projects, only the most valuable files
  ▪ raw data
  ▪ scripts
▪ tools
▪ final results
• If it can be regenerated, it doesn't go here!
• Still not a comprehensive backup solution
  
  What if a tsunami flooded Ångströmlaboratoriet?...
UPPMAX Nobackup Storage

- `/proj/project-name/nobackup/`
- Not backed up
- Contains everything else
  - anything that can be replaced or regenerated
- May be on a different disk ('disk volume')

```
milou-b: /proj/b2011141 $ cd nobackup
milou-b: /proj/b2011141/nobackup $ ll
```
```
total 5344
drwxrws---  4 jingwang b2011141    2048 Jan 16 2014 24_tremula_paper
drwxrws---  4 jingwang b2011141    2048 Jun 9 2014 24_tremula_trichocarpa
drwxrws---  3 jingwang b2011141    2048 Nov 13 23:57 3species
drwxrws---  4 nath b2011141        2048 Jun 15 2015 alignments
drwxrws--- 10 jingwang b2011141    2048 Apr 27 2015 all_populations
drwxrws---  5 jingwang b2011141    2048 Jan 27 13:05 all_populations_trichocarpa
drwxrws---  8 nath b2011141        2048 Apr 10 2015 annotation
drwxrwxr-x  2 root b2011141       24576 Apr 10 2015 backup_log
drwxrws---  5 jingwang b2011141    2048 Feb 15 2015 biyue
drwxrws---  7 jingwang b2011141    2048 May 15 2014 coverage
drwxrwx---  2 jingwang b2011141    2048 Apr 10 2015 environmental_phenotype
```
UPPMAX private/

• Contents only accessible to project members
  - /proj/project-name/private/
  - /proj/project-name/nobackup/private/

• Contents only accessible to you
  - $HOME/private/

```
milou-b: ~ $ ls -ld /proj/b2011141/private /proj/b2011141/nobackup/private
drwxrws--- 3 root b2011141 2048 Mar 2 2015 /proj/b2011141/nobackup/private
drwxrws--- 2 root b2011141 2048 Jun 26 2015 /proj/b2011141/private
milou-b: ~ $ ls -ld $HOME/private
drwx--S--- 2 douglas douglas 2048 Jan 6 2015 /home/douglas/private
```
UPPMAX Quotas

• `uquota [ project-number ]`

• **Hard quota: cannot be exceeded!!!!!**

• True for both *projects* and *home directories*

• Quota increases are **temporary**
UPPMAX glob

• `$HOME/glob/`
• Nobackup for your home directory
• Quota, but not a hard quota
  – ideal for large, short-lived files of unknown size
• Currently located on `/gulo` disk
  – (usually) higher performance than `/pica`
  – limited remaining lifetime (less than a year)
UPPMAX Scratch Storage

- A disk located on the node itself ('node-local')
- `$SNIC_TMP` is always set to its location
  - `/scratch` on a login node
  - `/scratch/job-number` within a SLURM job
- Explicitly short-lived and nobackup
  - deleted at end of job
- 'Small' size shared with other users
  - about 3 TB on milou compute nodes

```
m118: ~ $ df -kh /scratch
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/vg_local-scratch 3.6T 178M 3.4T 1% /scratch
```
Using scratch might really help

• UPPMAX high-performance disks are not optimised for random access (RA)
• RA occurs during database creation and usage
  – k-mer counts, custom blast databases, ...
  – some jobs will run 10x faster or more
• For a single job, it might not matter
• For multiple jobs it is really worth considering
  – especially if many jobs could run at the same time
• Do some tests!
milou-b: ~ $ module load bioinfo-tools
milou-b: ~ $ module load jellyfish/2.2.4
We strongly suggest the use of node-local temporary storage when creating and accessing large jellyfish databases on Uppmax systems. For more information use 'module help jellyfish'

milou-b: ~ $ module help jellyfish/2.2.4

--------------------------------- Module Specific Help for "jellyfish/2.2.4" ---------------------------------

    jellyfish - use jellyfish 2.2.4

    Version 2.2.4

We strongly suggest the use of node-local temporary storage when creating and accessing large jellyfish databases on Uppmax systems. The location of job-specific node-local temporary storage is available via the $SNIC_TMP variable set by SLURM. The storage is deleted when the job completes. The examples below are thus only valid when used within a SLURM job.

Ex: Create a jellyfish database in node-local storage and copy it to project storage

    jellyfish count -o $SNIC_TMP/mer_counts.jf [other options]
    cp $SNIC_TMP/mer_counts.jf .

Ex: Copy jellyfish database to node-local temporary storage and use it from there

    cp mer_counts.jf $SNIC_TMP/
    jellyfish histo [other options] $SNIC_TMP/mer_counts.jf
UPPMAX storage, by task

• Read large permanent files (FastQ)
  – project backed-up storage
• Read/write of large created files (BAM, ...)
  – glob, project nobackup
• Database creation and use?
  – under 3TB? yes: scratch no: project nobackup
• Ensured privacy?
  – private storage in project or home directory
• Uncertain, or have specific questions? Ask!