CANCER RESEARCH UK | CAMBRIDGE INSTITUTE

Strategies for backup*

*with informative and TRUE TALES of a CAUTIONARY NATURE not suitable for the faint hearted or weak of spirit

Why do bioinformaticians need to know about backup?

The same reason a lab scientist needs to know about QC and calibration: the same reason you don't trust a used car salesman.

- Because your group may assume you are the 'IT guy/gal'.
- Because it's better not to lose data due to a misunderstanding.
- Because firing the IT department won't bring your data back.
- Because the person who suffers most when you lose your data is, almost invariably, you.

In this talk we will learn:

Simple backup is not simple

- RAID levels, synchronisation, snapshots, recovery, global financial meltdown

To worry about corruption and mistrust

- Viruses, obsolescence, checksums, WORM

Fear of drowning

- (in too much data)

And finally:

How we do it here

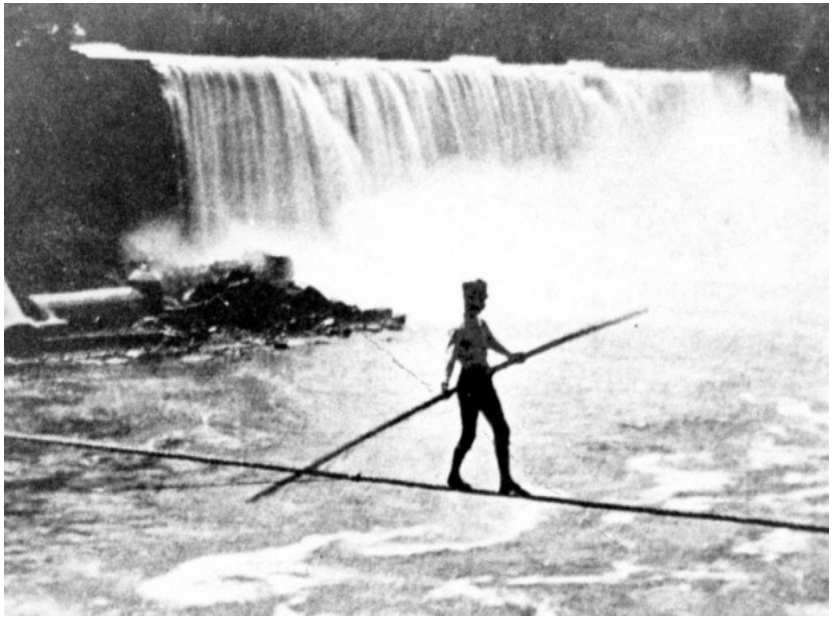
- The Institute's different systems and which approaches they use.

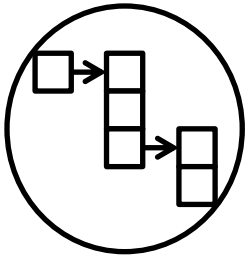
Strategy: no backup

This is not always a bad idea.

If you lose data, usually the thing of value lost is your time.

Any backup strategy which wastes your time might be doing more hard than good.





What can go wrong?

Disk failure rates are around 1.5% per year. This equates to a half-life of 46.2 years.

If your career is 40 years long, you might be lucky. The shorter your career, the less you have to worry (ironically).

in 2007, 2008, 2009, …, 2016:

Institute staff lose several hard drives per year. Usually, the OS stops working before any data is permanently lost; we can take apart the machine and read a copy.

Strategy: copies, RAID

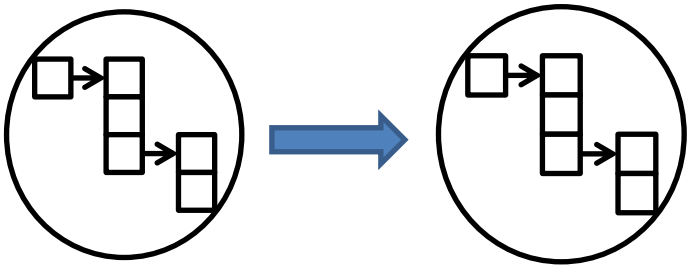
The most straightforward way to backup your data is to copy it to an external or internal hard drive.

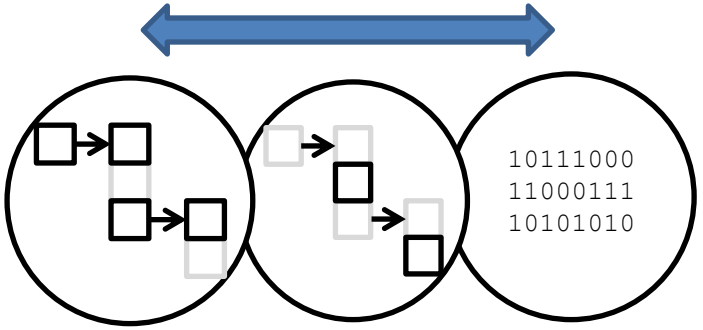
Doing this manually is time consuming so not always practical.

There are technologies which will do it internally, automatically. The different technologies are collectively known as RAID ('Redundant Arrays of Inexpensive Disks').

RAID1 refers to disk to disk mirroring. RAID3-6 use 'parity' bits to store the data more efficiently:

```
Disk 1: 1100
Disk 2: 0110
Disk 3: 0000
Parity: 1011
```





What can go wrong?

'RAID0' does not provide a copy of the data! It is used to make disk arrays faster.

in 2010:

We look after 2.2 PB in RAID arrays. That means ~2,200 1TB hard drives, or a failure rate of 3-4 per month.

In 2010, we used to use RAID5, which allows for a single disk failure; but disks more often fail when they are busy, so the RAID rebuild was always an anxious time.

We moved from RAID5 to RAID6, which uses extra parity checks and can survive two or more disk failures.

Synchronisation

Automated replication.

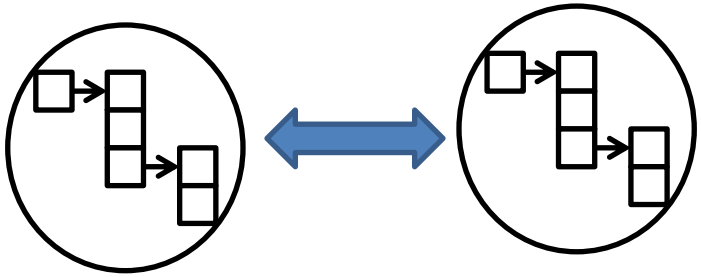
Ideally, automated replication to another machine.

Really ideally, automated replication to another machine in another building.

`rsynch`

You can achieve this yourself in linux systems using ‘rsynch’.

rsynch copies one file at a time, so is also a useful way to move large file-systems over the network if there is a risk of interruption.



What can go wrong?

in 2009:

A senior group leader's local laptop clock got set wrong (in the past).

Automated synchronisation set up for her laptop hard drive reversed direction, removing 2 weeks' work.

in 2010:

A virus missed by our antivirus software infected a networked machine, which then proceeded to corrupt every writeable file. Those corrupt files were then diligently synchronised to the replica.

We reverted to tape, and have stopped having 'public share' file systems.

Snapshots and increments

As a file system grows in time, most of the files appear, and don't change;only a few key ones go through many versions.

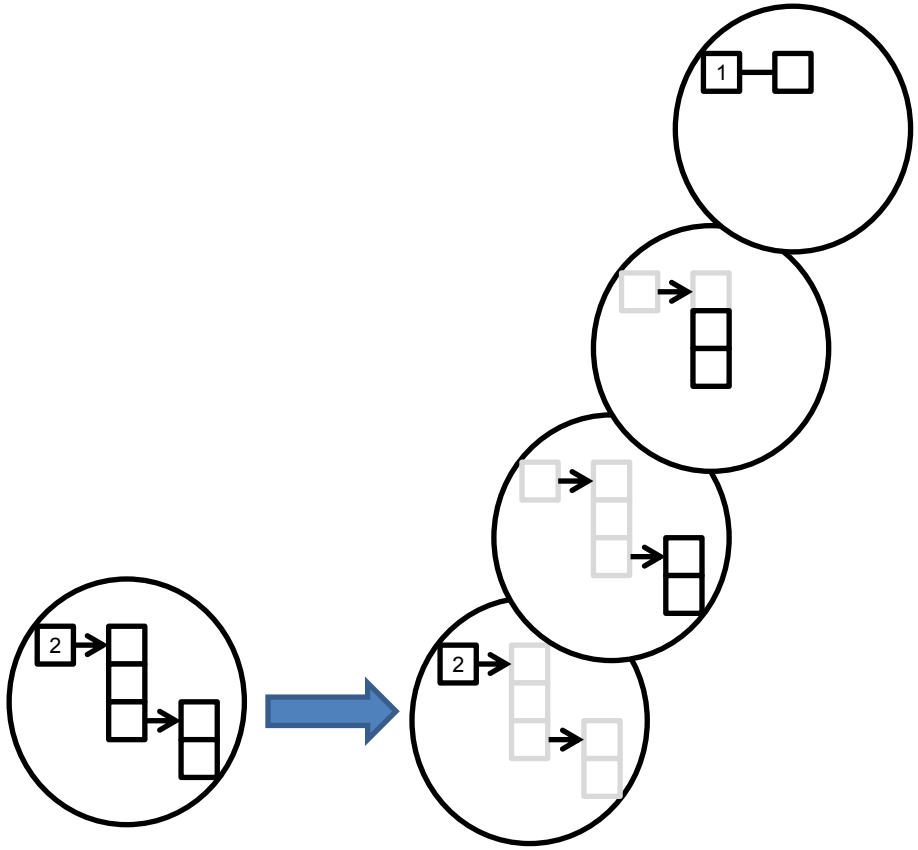
Snapshot technologies build an initial copy, then represent later versions of the file system using links to older copies.

If the file system is growing, the extra storage required is tiny.

Tape backup systems use an equivalent technology – they keep a database of what is already stored, and only backup the 'incremental' copies each night.

`rsnapshot`

You can achieve this yourself in linux systems using 'rsnapshot'. This includes settings to decide how far back in time you can go, which affects the space used.



What can go wrong?

in 2013:

The power to the cooling failed in our server room. The power to the servers didn't. Several large file systems got 'baked' to >45°C. This included a primary file system and its backup array.

These days, we hold replica file systems off site wherever possible.

Tape backup

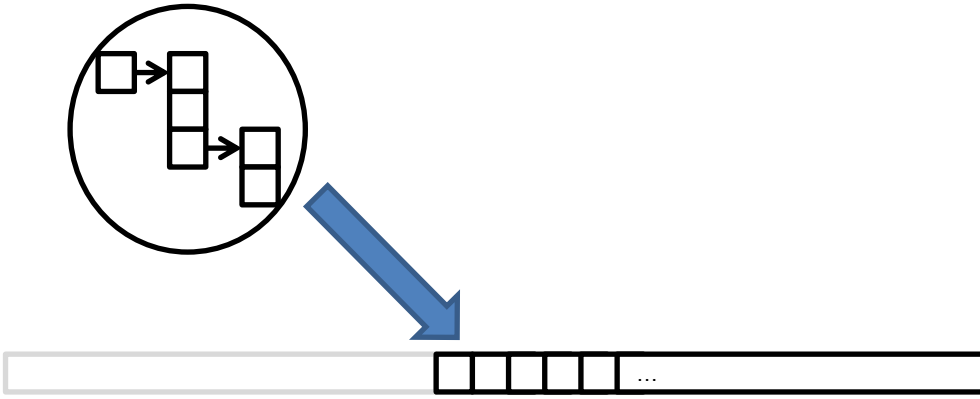
Every year, the industry predicts the ‘death of tape’.

Every year, the tape industry (led by IBM) make tape better so it doesn’t happen.

- Tape copies are slow and inefficient, but the tapes can be taken away and stored for years.
- The slowness can be a benefit; viruses and saboteurs can quickly damage file systems, but rarely have access to tape-recall systems.

`tar`

The Unix ‘archive to tape’ command is widely used to package up sets of files – even if they are going nowhere near a tape.



What can go wrong?

in 2008:

We had to recover the Institute's (un-replicated) main file systems from tape after a power spike + cut.

The process took **3 months**.

You need to think about 'recovery' plans as well as 'backup' plans…

Cloud backup

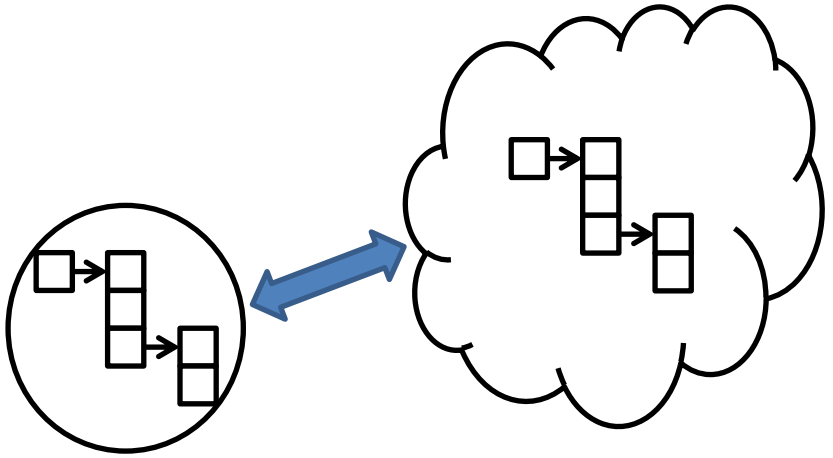
There are many solutions which can provide a copy of your data in a remote location, 'free' of charge.

- The actual charges you will incur will usually be network costs (moving 1TB over an external network costs us ~£25).
- Where the service includes automated replication, it is usually synchronisation.

amazon web services™

Dropbox

Google Drive



What can go wrong?

in 2012:

A trusted provider of business services lost ¼ of its customers' passwords to hackers.

in 2001:

A multi-$B US corporation, at one point responsible for every energy trade transaction in North America, which had never made a profit, was declared bankrupt and disappeared overnight.

Avoiding Corruption

Data can go bad.

- Viruses. Either by design or as a side effect, viruses can seep through all of your files corrupting them *en masse* or individually.

- Incompetence (or sabotage). Anyone else who has access to your files can damage or delete them. Incompetence/accident is 1000x more likely than sabotage.

- Silent and on-copy corruption. A file is a set of bits. Some file types, especially compressed files and databases, can become unusable after even a single bit change.

Checksums

There are mathematical algorithms – 'hash functions' – which will generate a unique, fixed-length pseudorandom string based on the content of a file. They can be used to see if a file has changed at the bit level.

```
md5sum
```

This is the most widespread 'cryptographic hash'. It's no longer used in cryptography, but provides a quick effective way of checking that two files are bit-by-bit identical.

Permissions

Sharing your data is good; but you can protect yourself by learning how permissions systems work for your files systems and only sharing your files read-only.

Avoiding Mistrust

There are reasons your data's provenance might come into question.

Bad reasons:

- Accusations of misconduct
- Any source of corruption or sabotage

…and some really good reasons:

- You want your data to validate a clinical trial data
- You want to defend a valuable patent
- You want to defend your claim to a Nobel Prize

WORM

'Write-Once Read-Many (times)' storage sets the attributes on files so they can't be modified once written.

This can obviously be a little impractical in day-to-day use but is good practice in the long term – even if it's not a formal requirement.

It effectively turns any file system into a giant CD burner, or a journal (with super-friendly reviewers and ultra-low impact factor).

Avoiding Obsolescence

As with file formats, not all physical devices are good places to put your data.

People of a certain age have their PhD on:

- A 5½ inch floppy drive
- **A 3¼ inch floppy drive**
- A DAT tape
- A Zip drive
- A CD

Readers for these formats now exist largely in museums of computing.

Keep it all, online, always

1. Package up your old data, to copy to permanent media before deleting.

2. Copy it to the permanent media.

3. Lose the permanent media. You will almost certainly never use it again.

4. Forget to delete the original data. Leave a copy of it on disk somewhere.

When will you run out of space?

Kryder's Law*

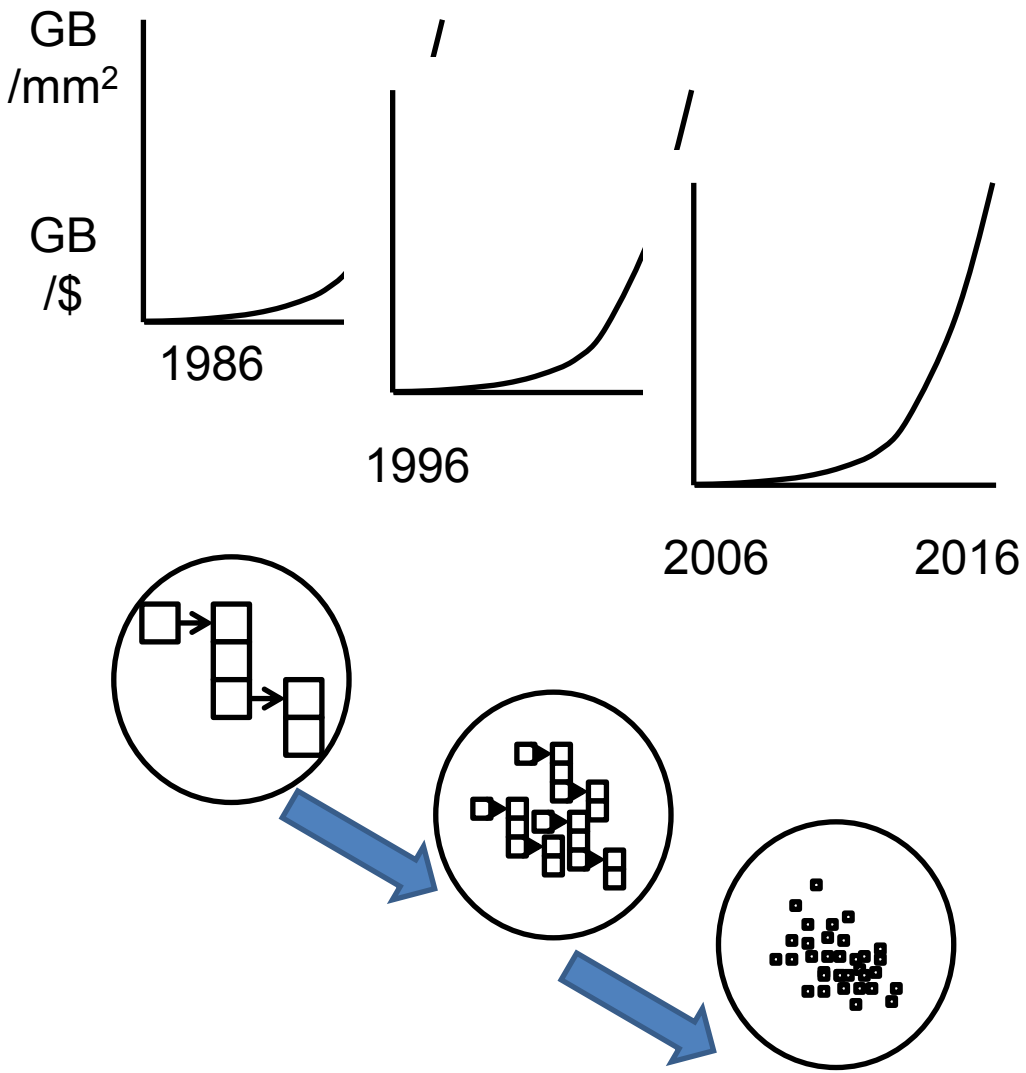
'Kryder's Law' is Moore's Law for data storage devices.

The density of storage increases, and the cost decreases, exponentially.

This means that archiving all of your old data can be done at a fixed cost – the total share of data more than 5 years old will rarely exceed 20% of the total data you own.

*Note that in Computer Science, 'Laws' have a use-by date.

In the past, physical limits to Kryder's Law have been overcome by technology changes. Unlike Moore's Law, there is no suggestion that we are anywhere near the end of Kryder's Law.



What can go wrong?

in 2016:

We have 1.2 PB of archived data.

Try finding something in that…

What we use at CRUK

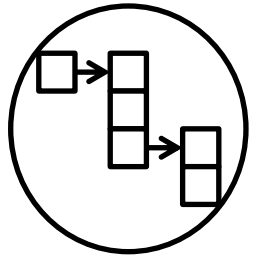
We use a range of techniques in different places

- Tier I for active data capture and analysis
- Tier II for current project work
- Tier III for archive

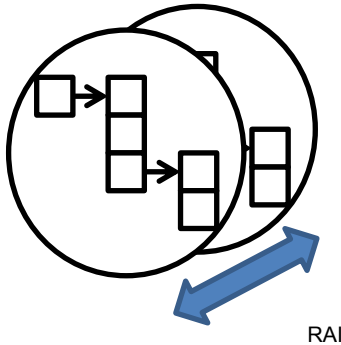
You must decide what stage your data is at!

‘Tier I’ Storage

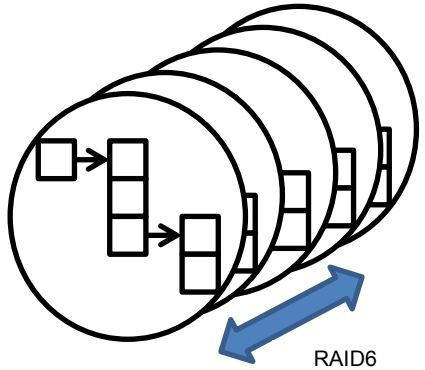
- Hard drives: ‘Nothing’
- Some acquisition machines: ‘RAID1’
- Cluster file system: ‘RAID6’



Nothing



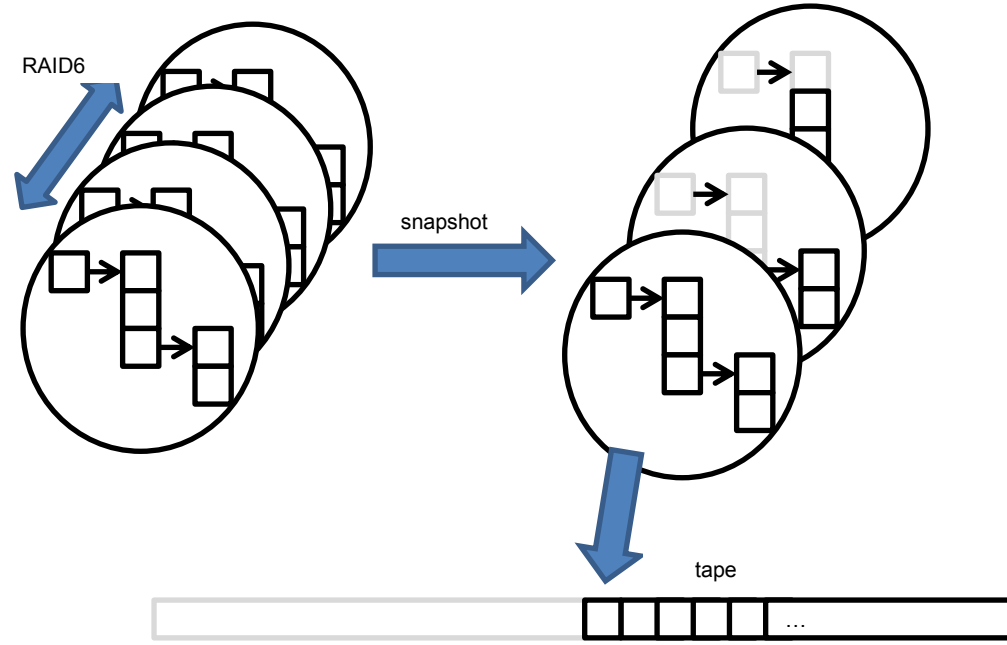
RAID1

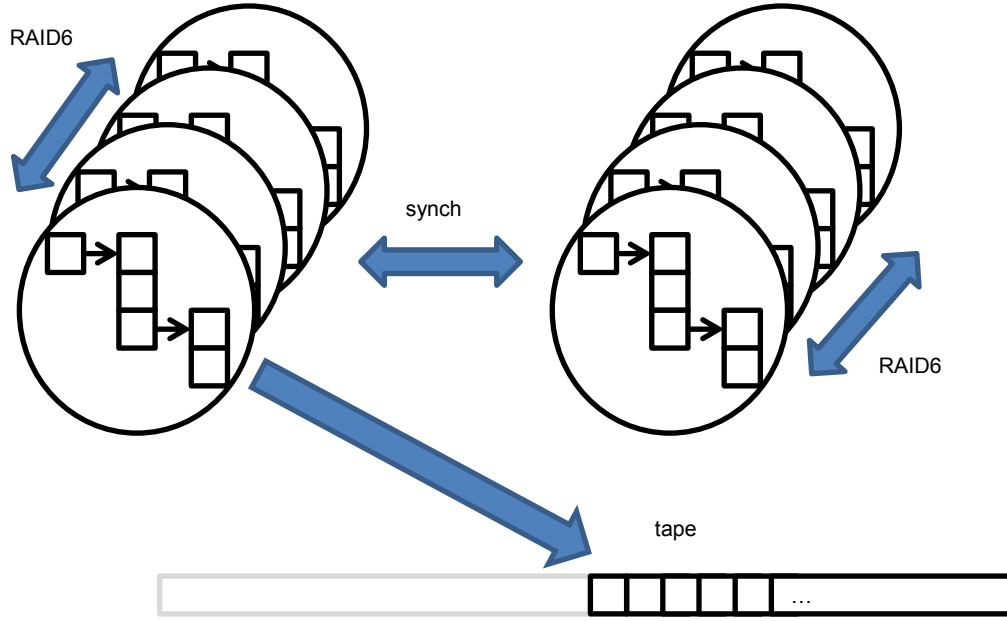


RAID6

‘Tier II’ storage

- /data, /nas storage: ‘RAID6, snapshot, tape’
- H: and G: drives: ‘RAID6, synch, tape’





‘Tier III’ storage

- /ibrix, /ark archive: ‘RAID6, WORM, checksum, tape’