# Introduction to Lustre

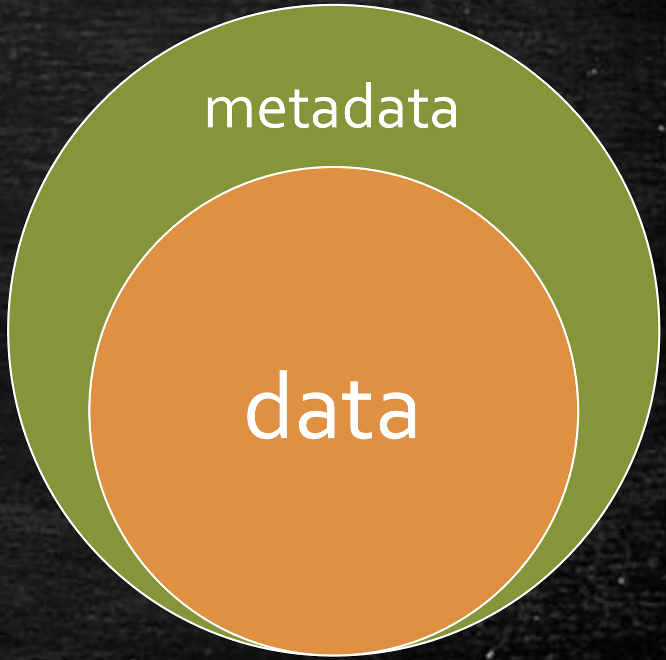
Mark Cheeseman
*Mark.Cheeseman@csiro.au*

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

# Parallel filesystem basics

# What is a filesystem?

- A way of organizing data on a storage device
- We *see* data as having two components:
  - Data: raw information
    - numbers, characters, images, etc
  - Metadata: description of the stored data
    - array length
    - type (integer, float, double, etc)
    - address at which data is stored
- I/O operations use both components



# What’s different with parallel filesystems?

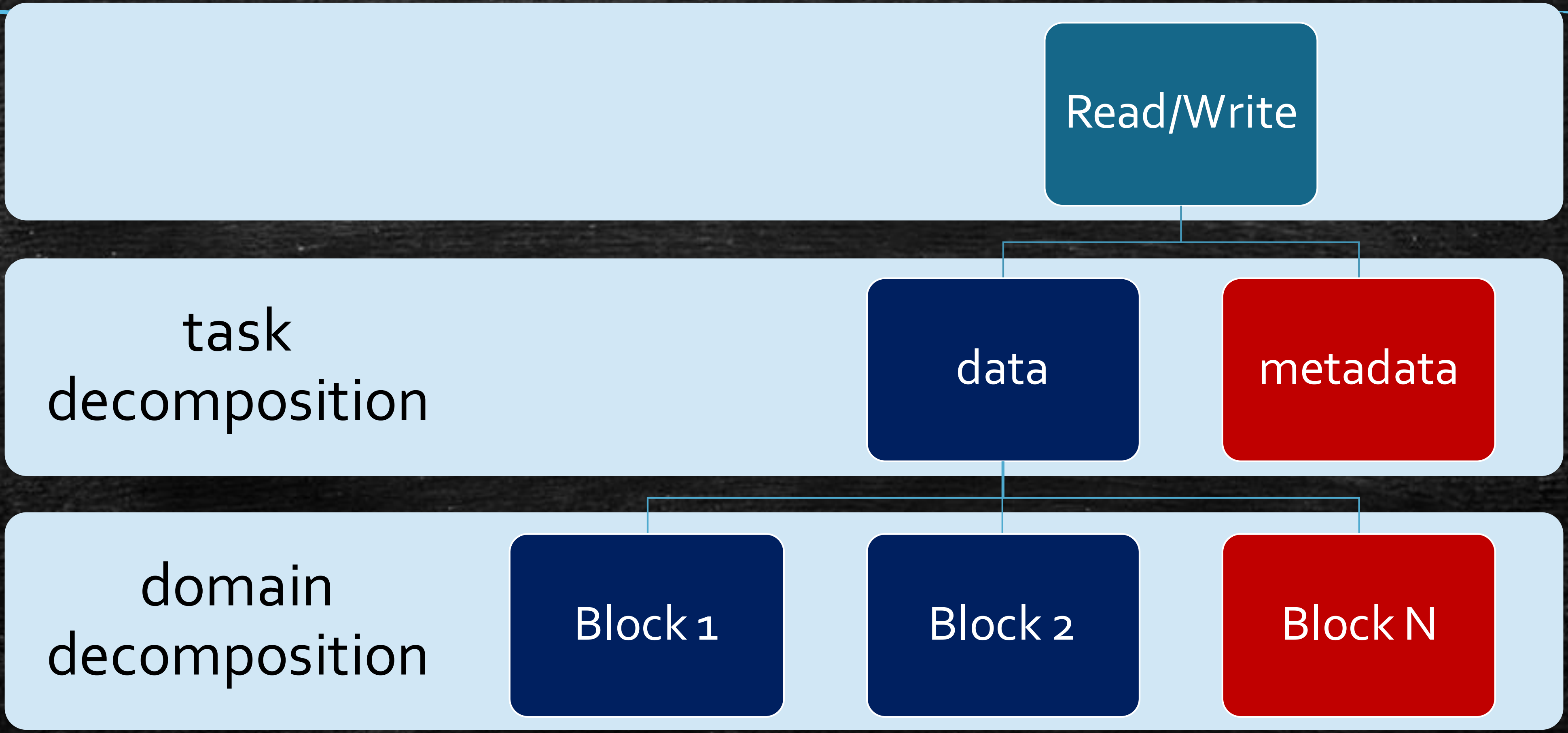
A parallel filesystem employs both task and domain parallelism to increase throughput of I/O operations.

**Task Parallelism**

Separation of metadata and data operations

**Domain Decomposition**

Break data into pieces that are to be handled on different hardware



Read/Write

task decomposition

data

metadata

domain decomposition

Block 1

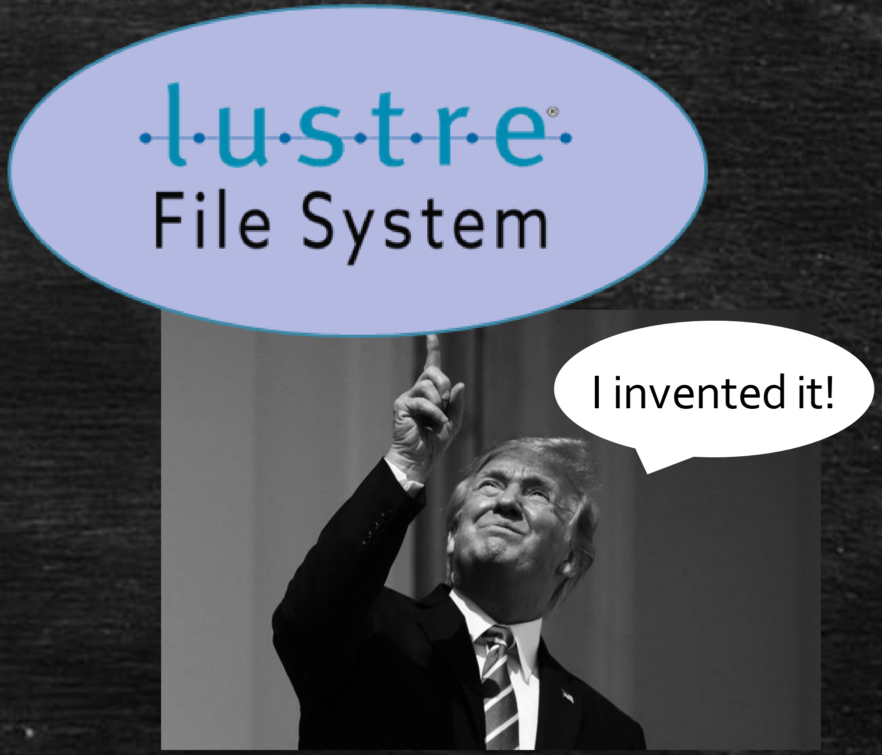
Block 2

Block N

# Common Parallel Filesystems

- General Parallel File System (GPFS) / Lustre
  - Commonly associated with supercomputing
  - Employed at NCI, Pawsey, CSIRO and Swinbourne
  - Can be used on AWS and Azure

- Gluster, BeeGFS
  - Parallel distributed file systems
  - More associated with cloud computing



# What is Lustre?

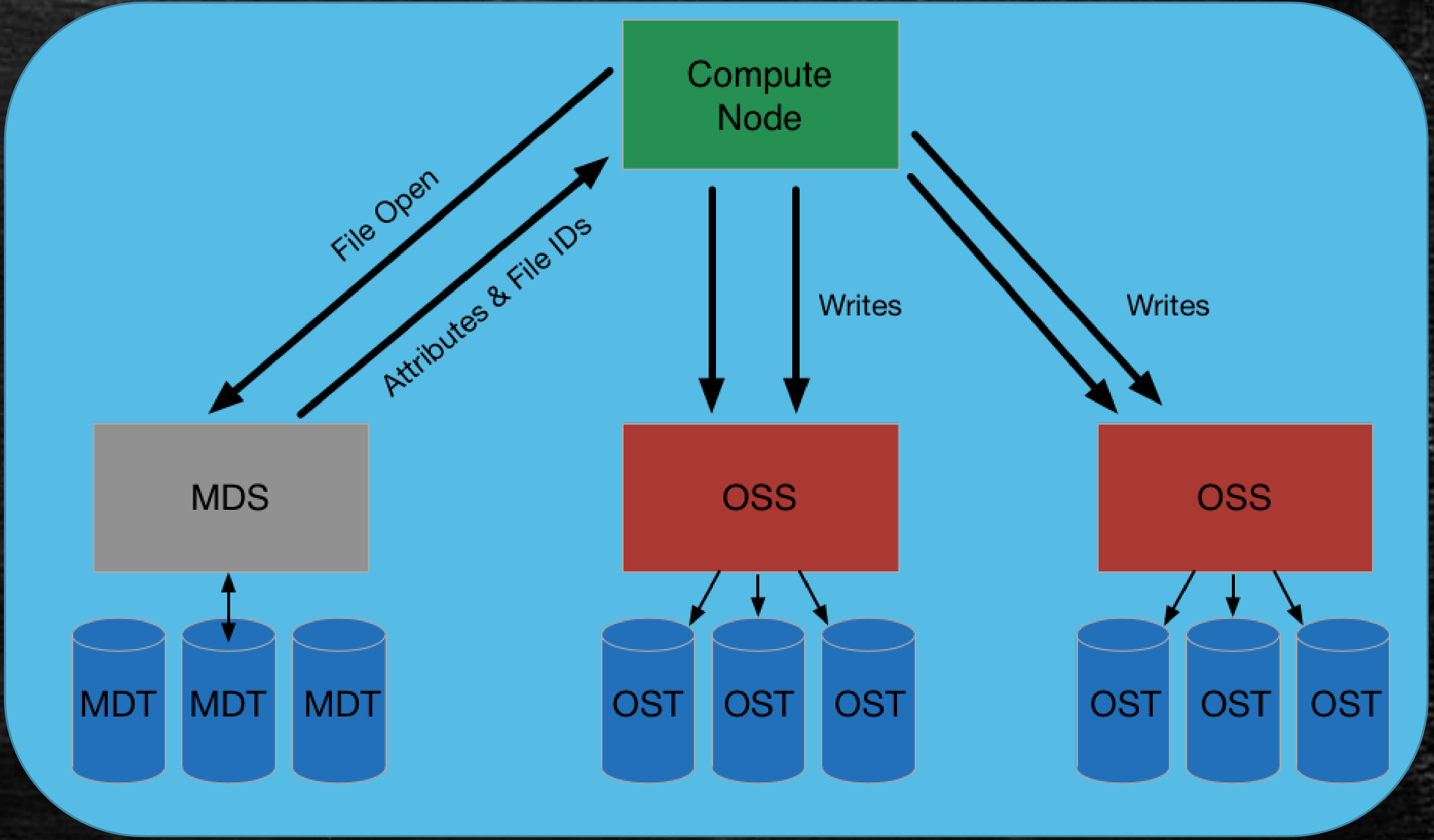
# Points of Interest

- Widely used
  - Most hardware vendors offer support
  - Majority of Top 500 systems use it
- Built for heavy duty performance
  - Used with hundreds of thousands of compute nodes
  - Deployed on filesystems with petabytes of capacity
  - capable of Terabyte/sec throughputs
- Industry & community acceptance
  - Lustre users group
  - Application support in libraries such as HDF5
  - Can get service contracts through vendors

# Overview

Lustre uses both task and domain parallelism

Different hardware is used for metadata and data operations.



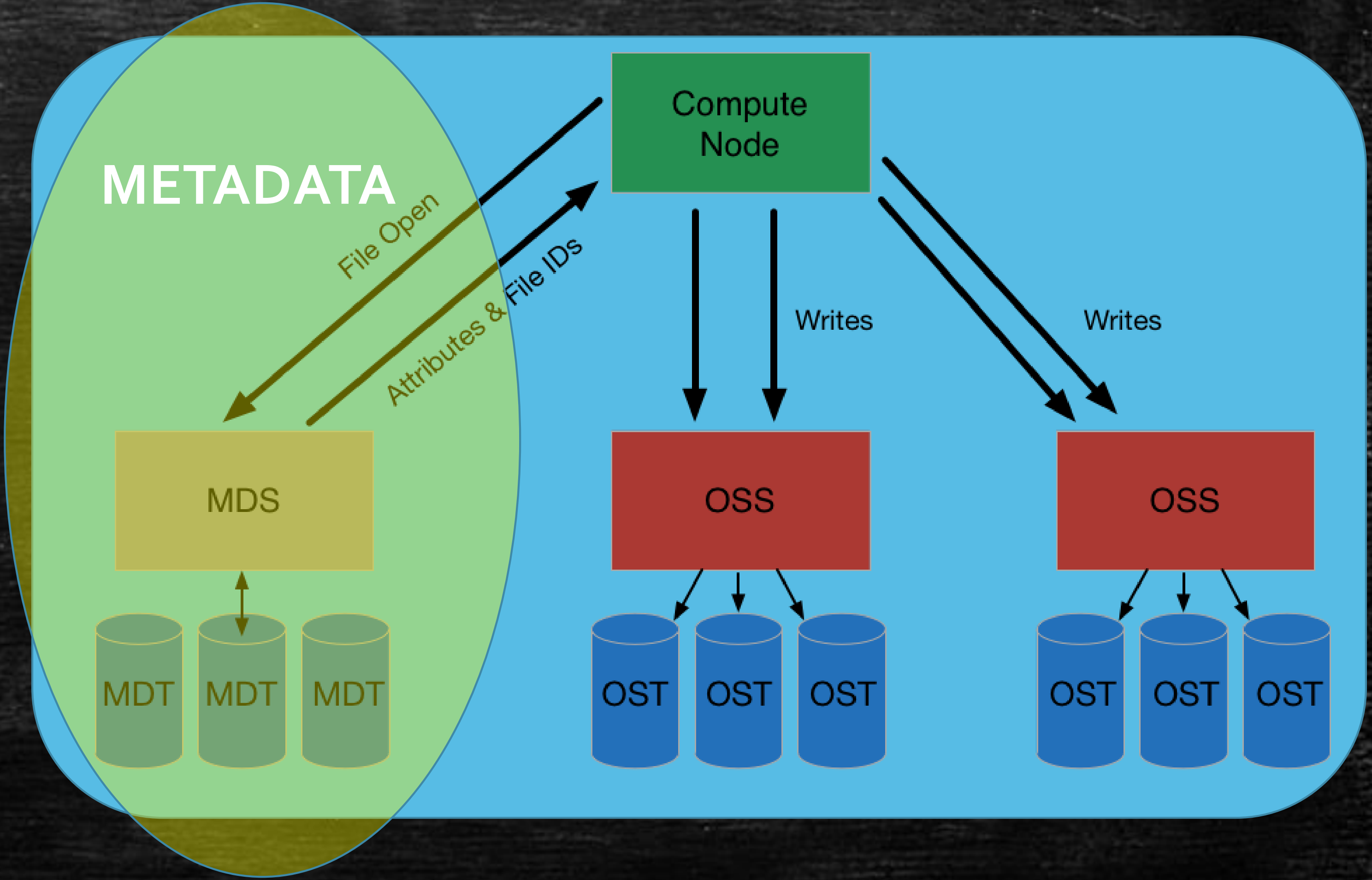
# Overview

## Meta-Data Server (MDS)

- Tracks storage locations of all files
- Directs I/O requests to appropriate OSTs and OSSes

## Meta-Data Target (MDT)

- Stores all metadata
- Filenames, permissions, paths, layout, etc



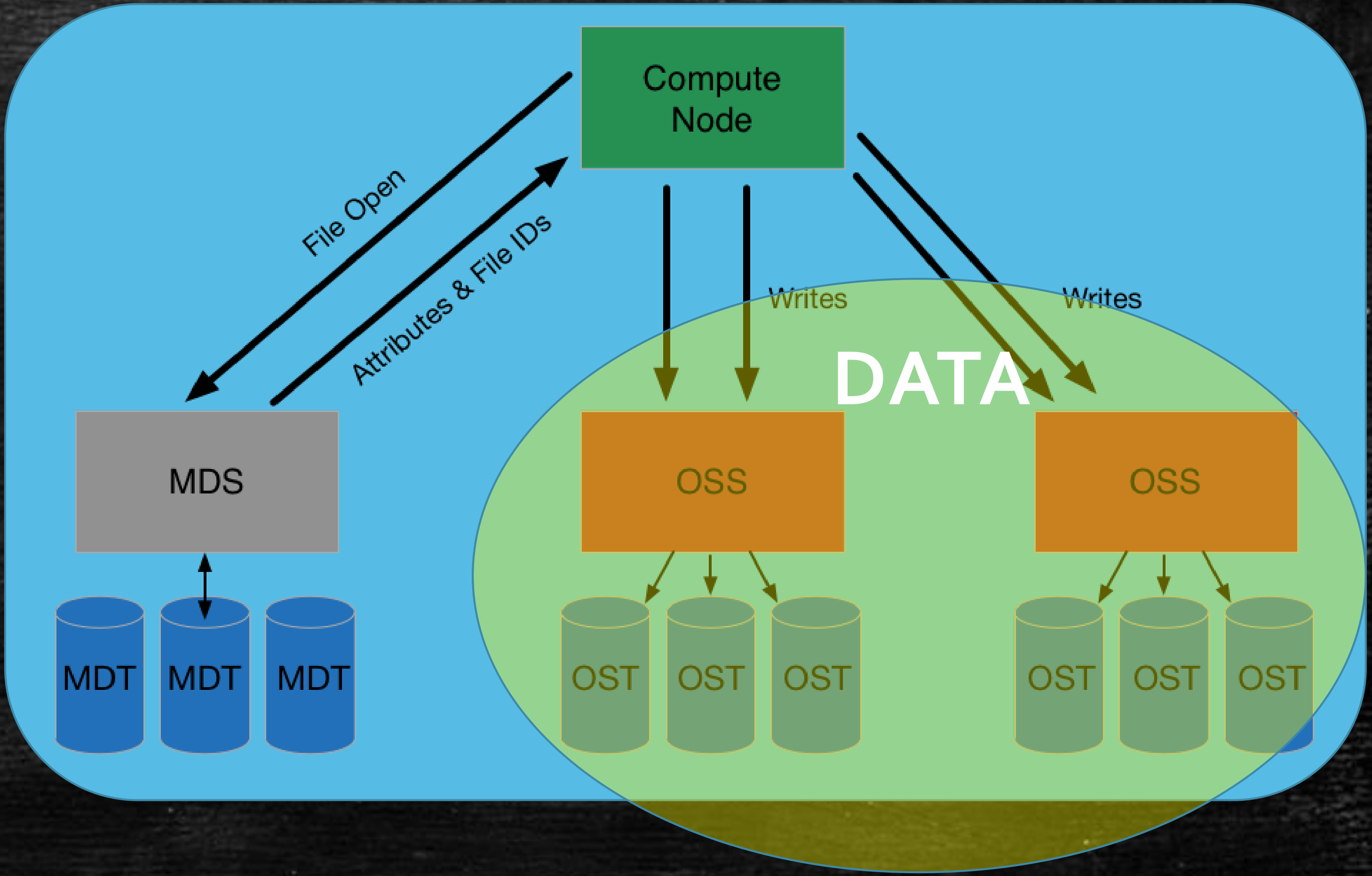
# Overview

Object Storage Server (OSS)

- manages a subset of OSTs
- handles network and I/O requests

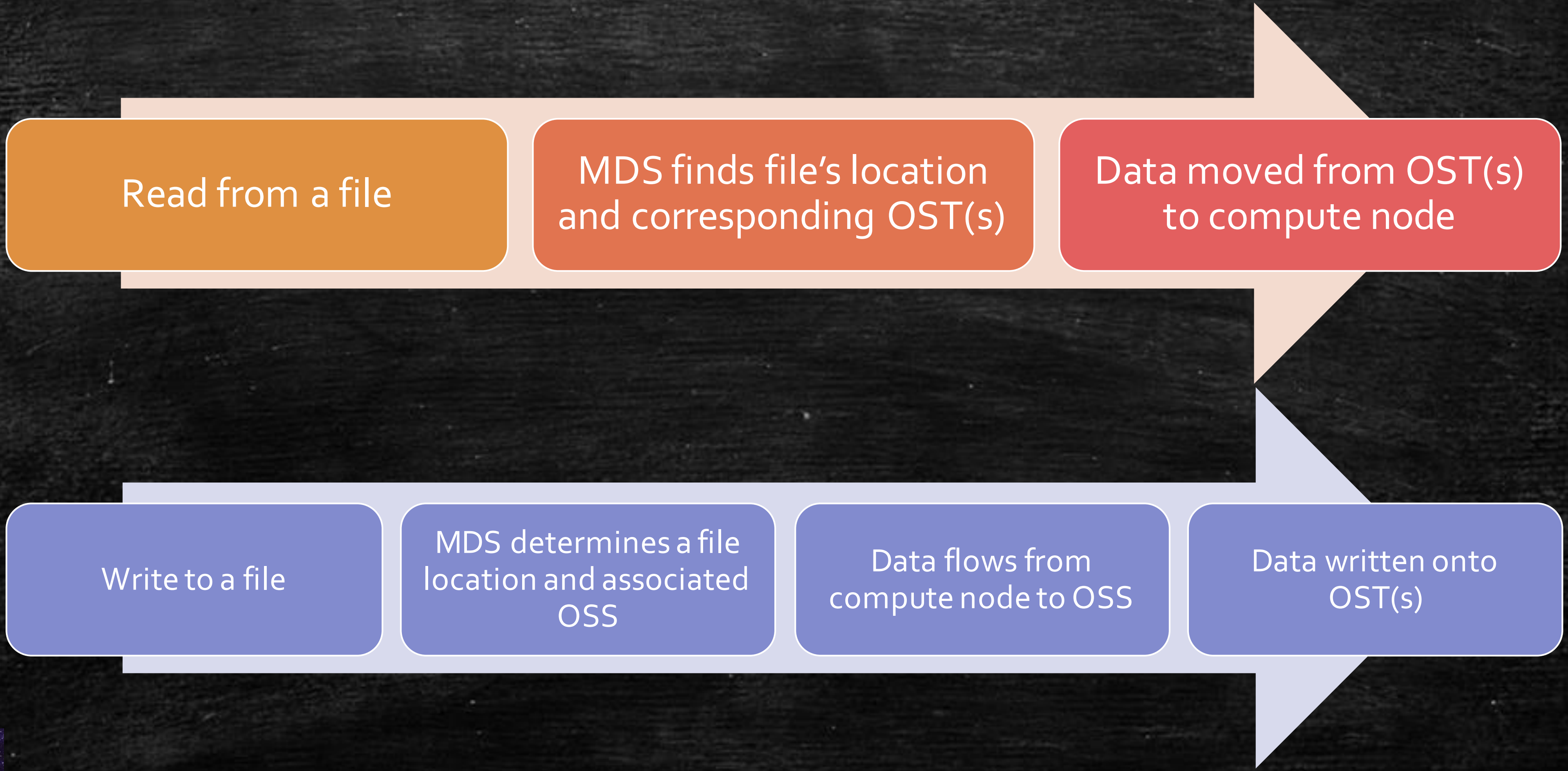
Object Storage Target (OST)

- stores data stripes
- RAID array of hard disks



More hardware dedicated to the DATA side

# What happens during a read/write?



Read from a file

MDS finds file’s location and corresponding OST(s)

Data moved from OST(s) to compute node

Write to a file

MDS determines a file location and associated OSS

Data flows from compute node to OSS

Data written onto OST(s)

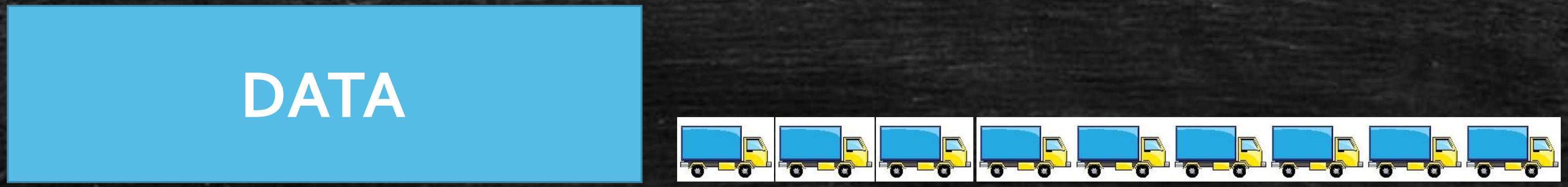
# Tuneable Parameters in Lustre

Will stick to two parameters from that effect IO performance:

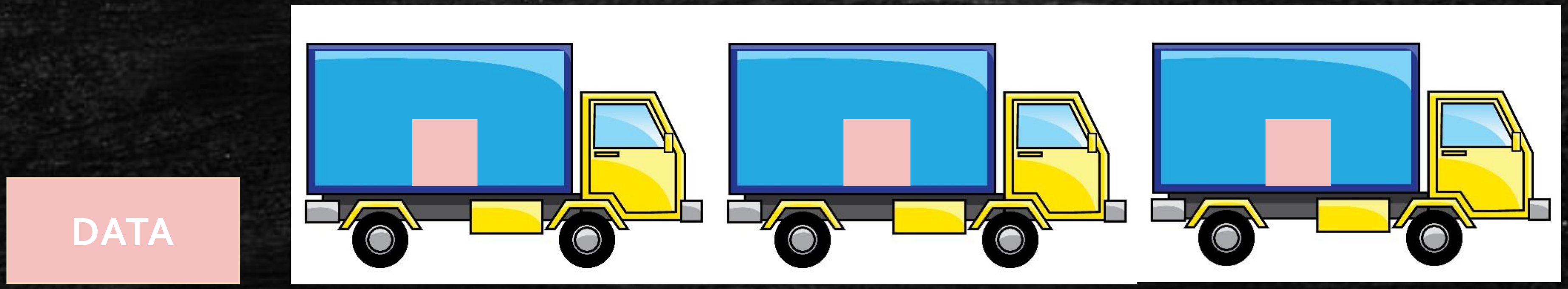
# Block Size

The size of the individual data chunks that are routed to/from OSTs.

**Too small**: there will be too many blocks leading to bandwidth issues


DATA

**Too large**:  partially filled blocks are used leading to latency issues


DATA

# Number of OSTs

A file or directory can have a specified number of OSTs assigned to it. It will have its data blocks spread among these OSTs.

**Too few OSTS**: Bandwidth issues arise as the data blocks are blocked by the lack of "connections" to hard disk.

**Too many OSTS**: Latency can become an issue as data blocks are too spread. Metadata tracking can become prohibitively expensive.

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

# Changing Lustre Parameters

- These parameters can be attributed to a single file or a directory of individual files.
- ***Be careful when modifying these parameters.  Know what you are doing!***
- System defaults are generally good. Only change when observed IO performance is bad.
- Remember that filesystems are a shared resource!

# Demo: Striping and IO performance

Two Python demo scripts are available:

lustre_block_size_test.py

lustre_num_stripes_test.py

We will investigate the effect that stripe size and number of OSTs play in write performance on a Lustre filesystem.

Which factor has the greatest effect? Why?

Can you think of situations when the each factor is dominant in determining observed IO performance?

ADACS
ASTRONOMY DATA AND COMPUTING SERVICES

# Demo: How to Run

1. Click on the terminal in your JupyterHub screen
2. Connect via ssh to the Athena cluster located at Pawsey

   ```
   ssh couXXX@athena.pawsey.org.au
   ```

3. Go into the Lustre demo directory

   ```
   cd HPC-Workshop/lustre
   ```

4. Submit the 2 demo jobscripts to SLURM

   ```
   sbatch jobscript.slurm_block_size
   sbatch jobscript.slurm_num_osts
   ```

5. Look at the output with the vi text editor

   ```
   vi LustreTest_VaryingBlockSize
   vi LustreTest_VaryingNumOSTs
   ```

6. Exit vi by pressing :q and then ENTER

# Getting the most out of Lustre

# The Good and Bad of Lustre

It excels at:

- large contiguous or uniformly positioned reads/writes
- low frequency IO patterns
- using MPI-IO

It struggles with:

- high frequency reads/writes
- discontinuous / non-uniform read or write patterns
- accessing large numbers of files simultaneously
- Metadata heavy operations (ls –la)

# Useful Lustre-specific Commands

| Command | Description |
|---|---|
| **lfs mkdir –c <# ost> -s <stripe size> <dir>** | Create a directory over a given # of OSTs and stripe size |
| **lfs getstripe <file / dir>** | Get the stripe size and # of OSTs assigned to a file or directory |
| **lfs setstripe –c <# ost> -s <stripe size> <file>** | Set the stripe size. # is number of bytes |
| **lfs migrate –c <# ost> -s <stripe size> <file>** | Modify the stripe count and/or size for an existing file or directory |
| **lfs find <dir>** | List the contents of a directory |
| **lfs df** | Check disk space usage |

# Other recommendations

- Avoid doing the following:
  - using auto/tab complete
  - wildcard expressions (especially `tar` and `rm`)
    - Generate an explicit list of files and operate on each individually
  - thousands of files in a single directory
    - use a directory structure
  - Using ls –l unless absolutely necessary
- Open files as `READ_ONLY` if they will only be accessed for reading
- Don’t stripe small files (<1 MB) across multiple OSTs
  - Set stripe count to 1 for a directory and write files in it
  - Be careful not to specify a specific OST!!

# Other recommendations

- Store executables in non-parallel file system
- Don’t edit text/source files in a Lustre file system
  - vi creates temporary “snapshot” files that increase metadata loads
- Remember to appropriately stripe when moving files from a non-parallel filesystem to the Lustre filesystem
- System-level caching can effect expected performance