# XRootD Architecture for modern storage use cases

Jyothish Thomas

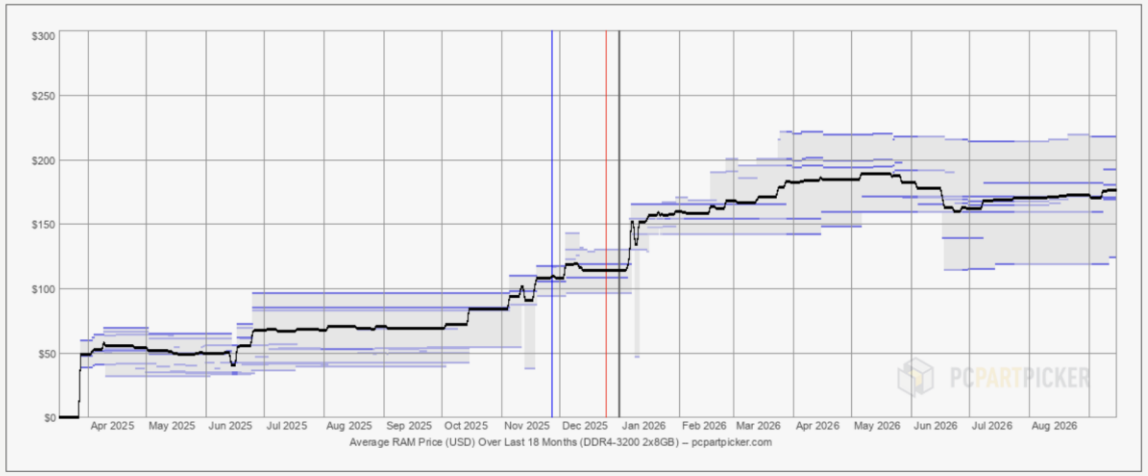
jyothish.thomas@stfc.ac.uk

# The Building blocks

- RAM
- SSD
- HDD
- Network
- CPU Core count

# RAM

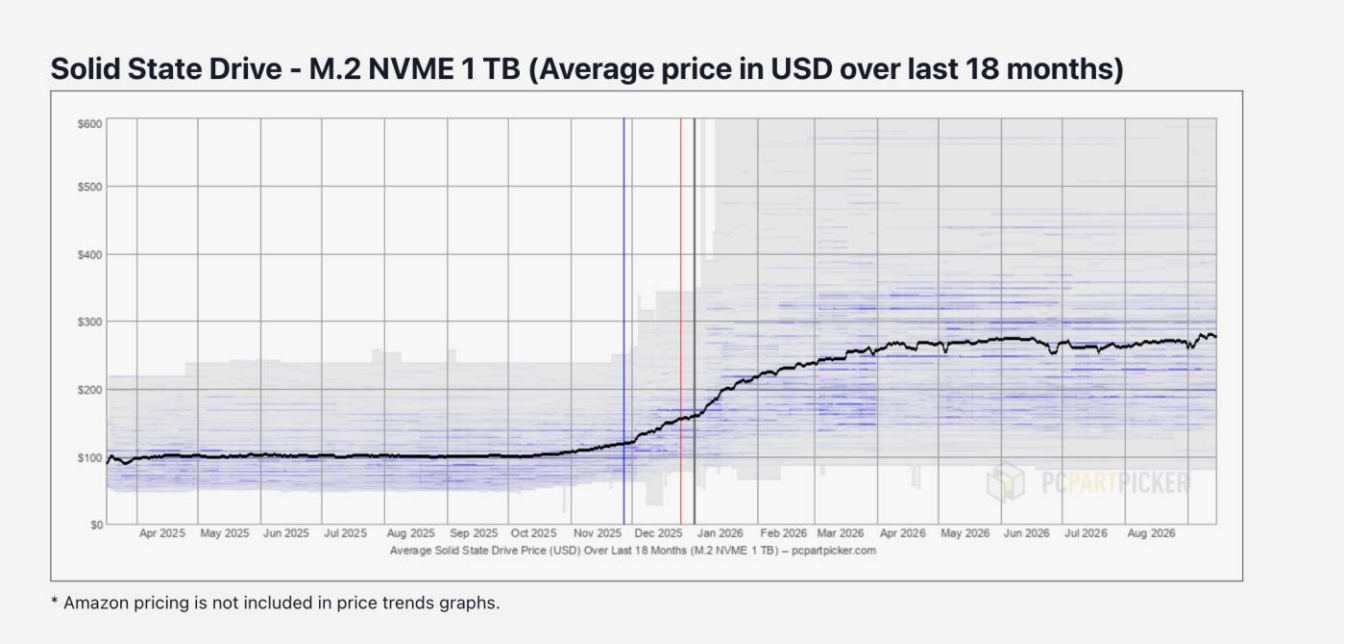
- Speed: 4000-6400 MT/s (256-384Gbps per DIMM)
- Size 16-64GB per card cost effective range
- Use: XRootD process memory, transient caches for transfers
- Recent Price trend:




Science and Technology Facilities Council

# SSD

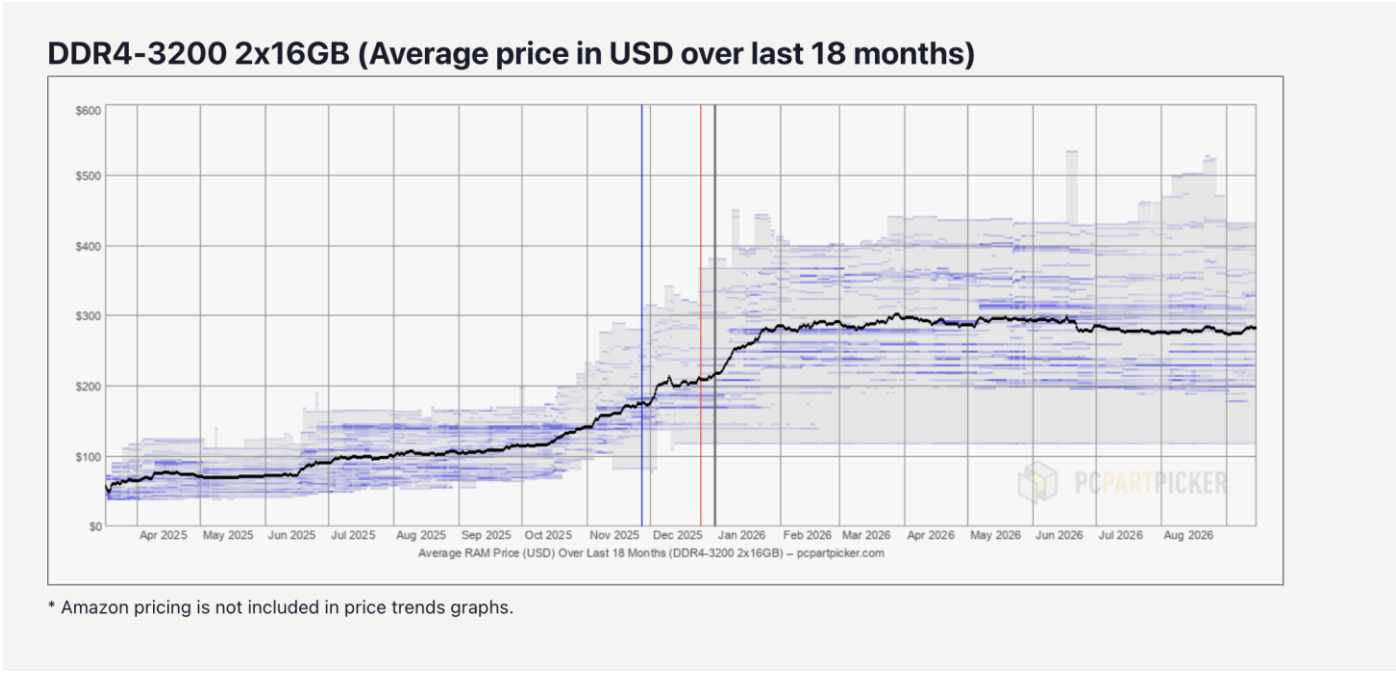
- Speed: 3000-7000MB/s (24-56 Gbps)
  - 80-150Mbps random access
- Size: 512GB -  32TB per disk
- Iops: 40-50k write, 500-2000k read
- Use: Fast, high-performance storage, persistent caches
- Recent Price trend:




Science and Technology Facilities Council

# HDD

- Speed: 80-160MB/s (0.64-1.28 Gbps)
  - 0.8-1.6Mbps random access
- Size: 512GB -  32TB per disk
- Iops: 55-200 read/write
- Use: Large low cost storage
- Recent Price Trend:



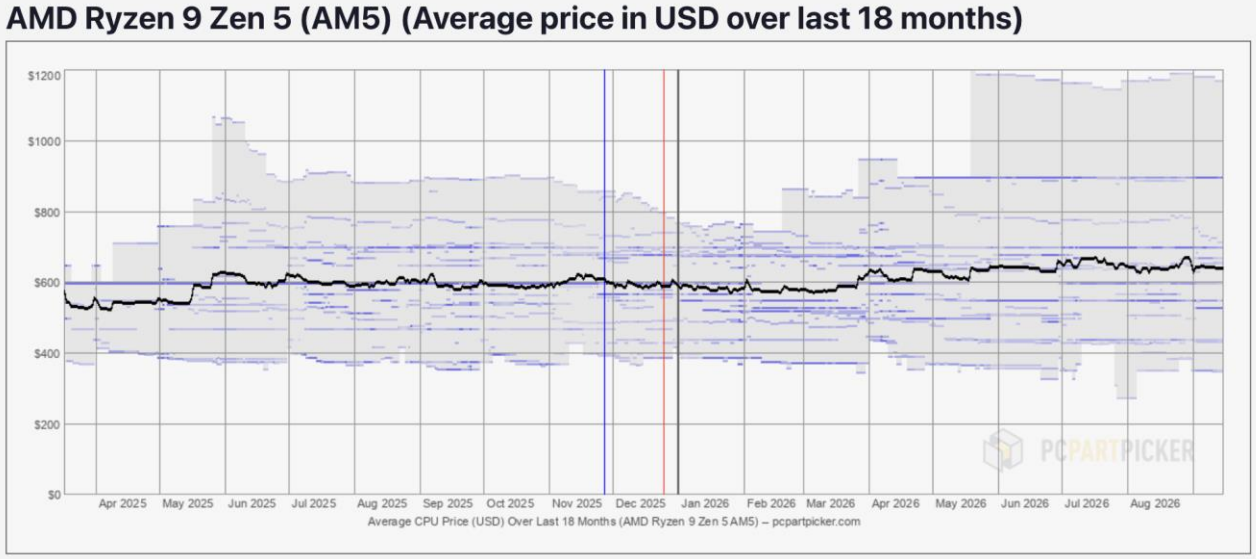

Science and Technology Facilities Council

# Network

- Speeds: 25-100Gbps standard enterprise nodes, 200-800Gbps available for high end nodes.
- RNE WAN network ranging from 1-800Gbps
  - median 100Gbps for large sites, lot of variance depending on the country

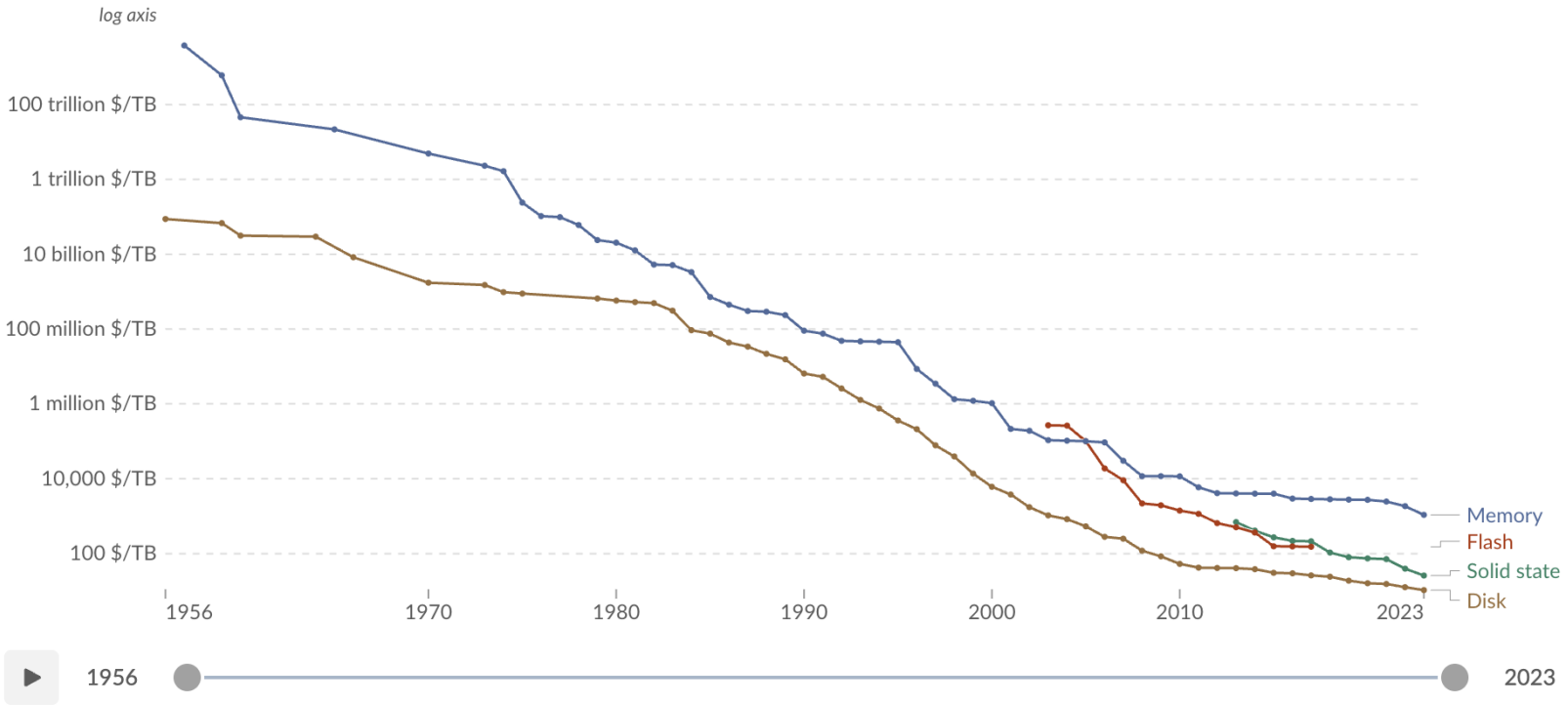
# Core count

- 4-64 average core counts for XRootD machines
- ReadV operations require faster CPUs and higher core counts
- More info in [Andrea Picinelli's CHEP talk]




Science and Technology Facilities Council

# A glimpse back in history




**Data source:** John C. McCallum (2023); U.S. Bureau of Labor Statistics (2026) – Learn more about this data

**Note:** For each year, the time series shows the cheapest historical price recorded until that year. This data is expressed in constant 2020 US$.

OurWorldinData.org/technological-change | CC BY

Science and Technology Facilities Council

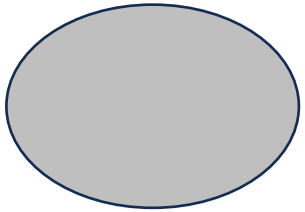
# XRootD Architecture basics

- Servers
- Proxy
- XCache
- Redirector

UKRI Science and Technology Facilities Council

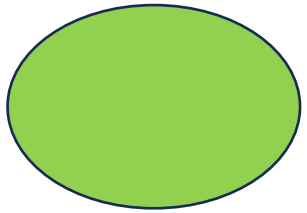
# Servers

- Standard XRootD nodes.
- Mount a storage onto the node, set up port permissions and they’re good to go!
- Supports read/write traffic



# Proxies (XrdPfc)

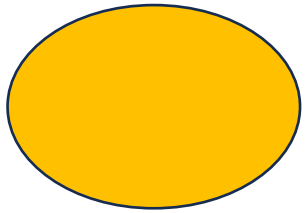
- Intercept client requests and forward them to a data server
- Acts as gateways to access secured storage unreachable via the internet



Science and Technology Facilities Council

# XCache

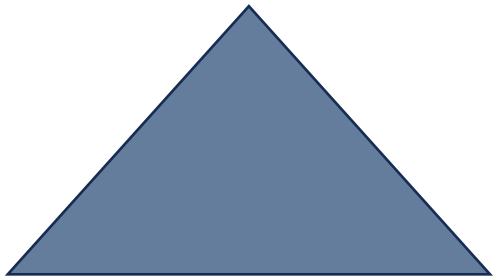
- It’s a caching proxy
- When a file is requested, it’s fetched from an origin and stored in a local cache.
- You can configure a readahead when partial files are requested



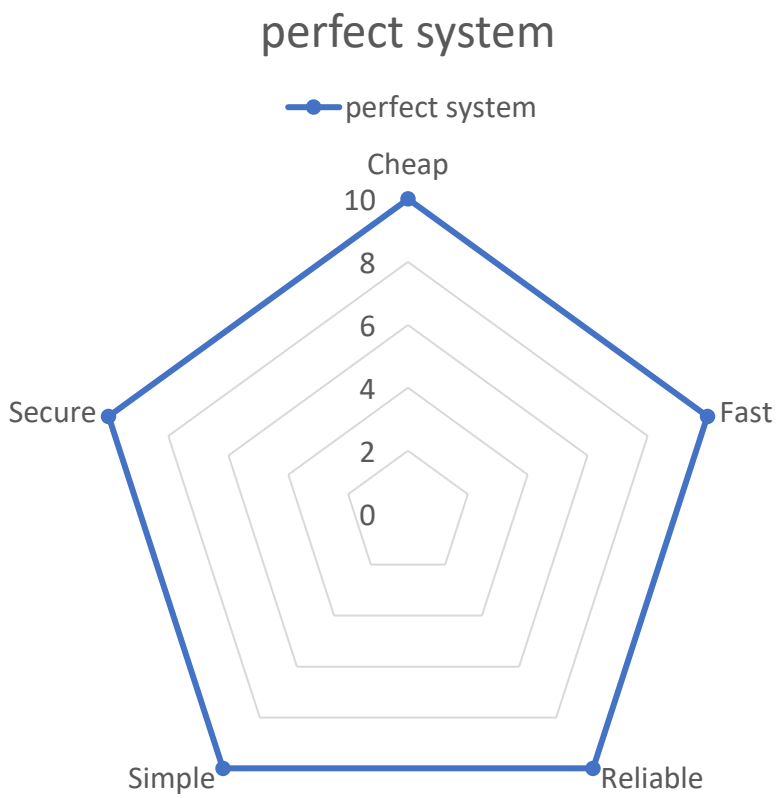
Science and Technology Facilities Council

# Redirector (Cmsd)

- Manages a cluster of servers, proxies and/or Xcaches
- Load balances and decides which server to send the transfer to



# Quick reference graphs

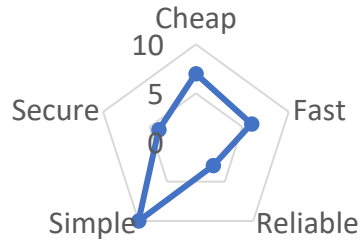


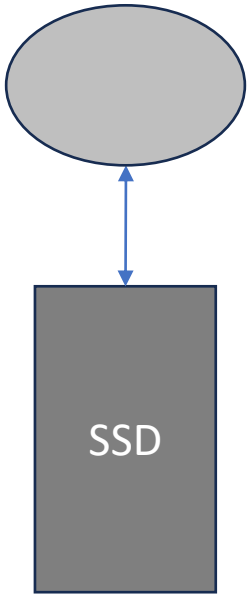

Science and Technology Facilities Council

# Cases, architectures, possible improvements

- Storage size and structure
  - Independent Small-scale storage
  - Small scale storage backed by a larger one
  - Large storage
  - Federated storage
- Activity type
  - Small reads and vector reads
  - Large sequential reads

# Simple server



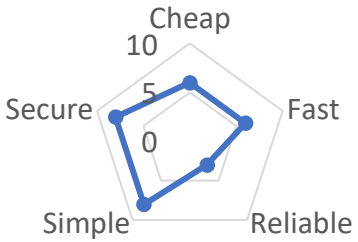

High Priority: memory, CPU, network and disk performance

Low priority: disk capacity

+ good for small storages or remote mounts
+ easy to implement
- Throughput limited by single server
- No resilience
- Does not scale for larger storages

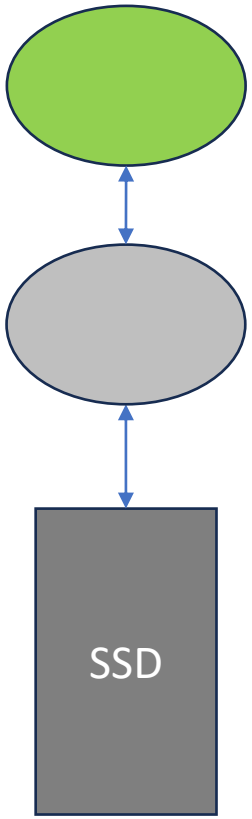
Science and Technology Facilities Council

# Simple Proxy




High Priority: memory, CPU

Low priority: disk capacity, performance network



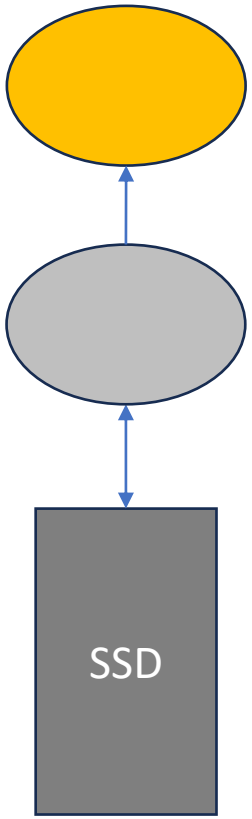

+ used to expose HPC storage resources
+ allows firewalling of storage nodes and servers
- Throughput limited by proxy
- No resilience

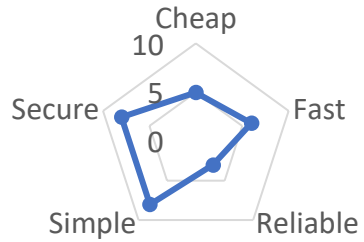
Science and Technology Facilities Council

# Simple XCache

High Priority: CPU, disk performance, memory

Medium Priority: disk capacity, Network

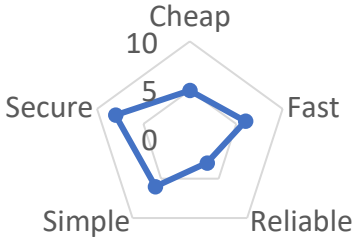


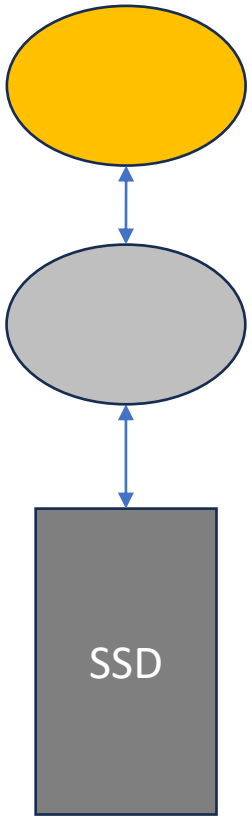

+ it’s a cache
+ it’s a proxy
+ better for remote origins
- Read only

Science and Technology Facilities Council

# Writetrough XCache



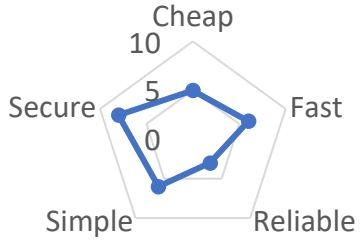

pfc.writemode writethrough

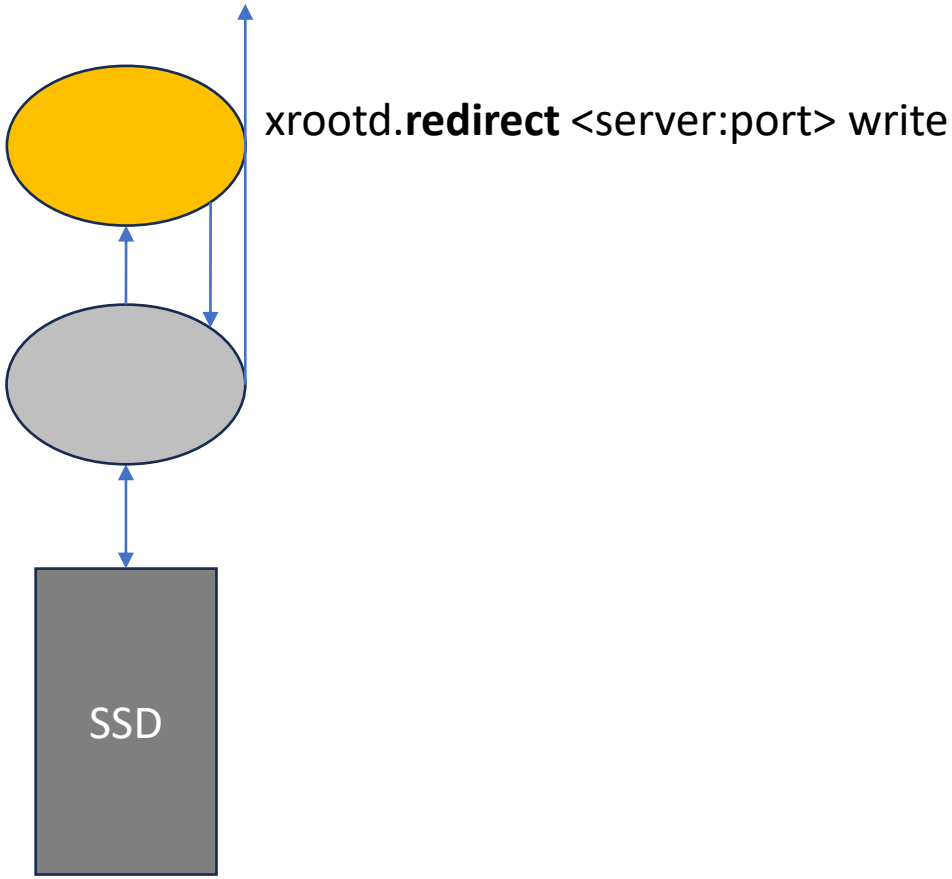



+ allows Xcache node to be used in non-caching proxy mode for writes
+ origin server does not have to be directly
+ better for near origins
- accessible by clients
- throughput limited by xcache node

Science and Technology Facilities Council

# Writable XCache

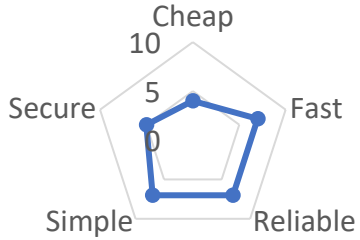



+ redirects writes to origin
+ offloads write traffic to origin
- Requires origin server to be accessible by clients

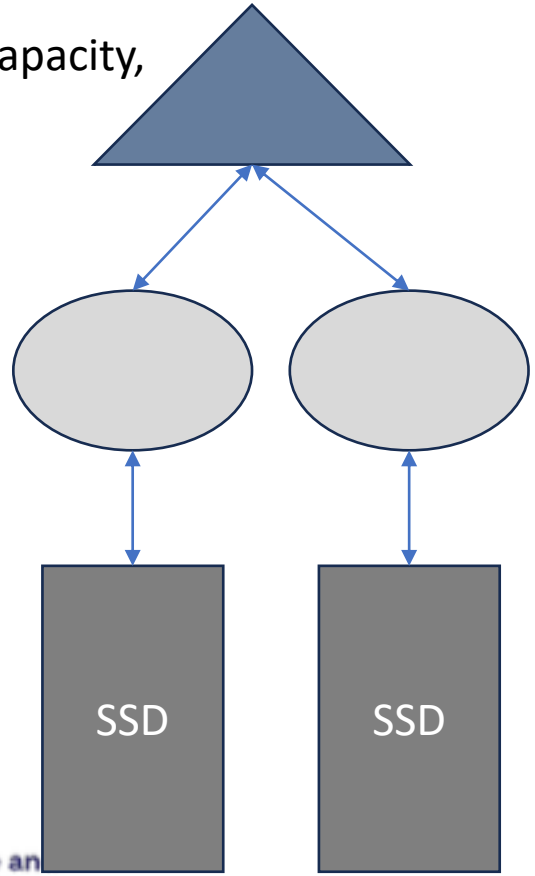
Science and Technology Facilities Council

# Simple redirected




High Priority: CPU

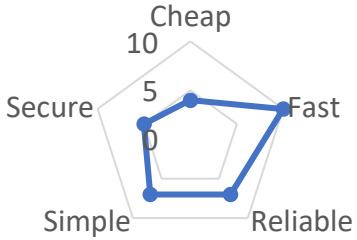
Low priority: disk capacity, performance network,memory

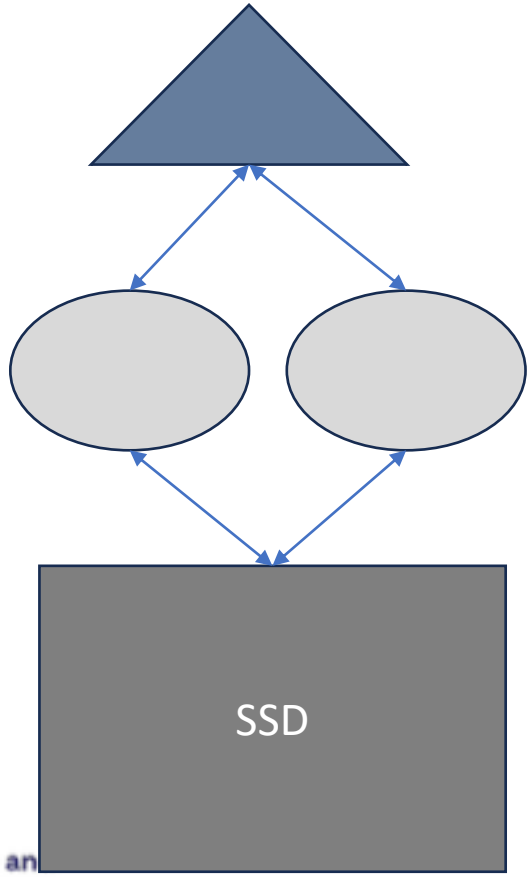



+ allows some storage resilience depending on the setup
+ allows scaling and possibility of clean data separation between local storage elements
- Less resilient to spikes on individual servers
- Disks can get filled unevenly

Science and Technology Facilities Council

# Distributed(Shared) Filesystem (dfs)



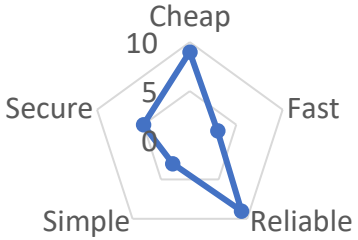



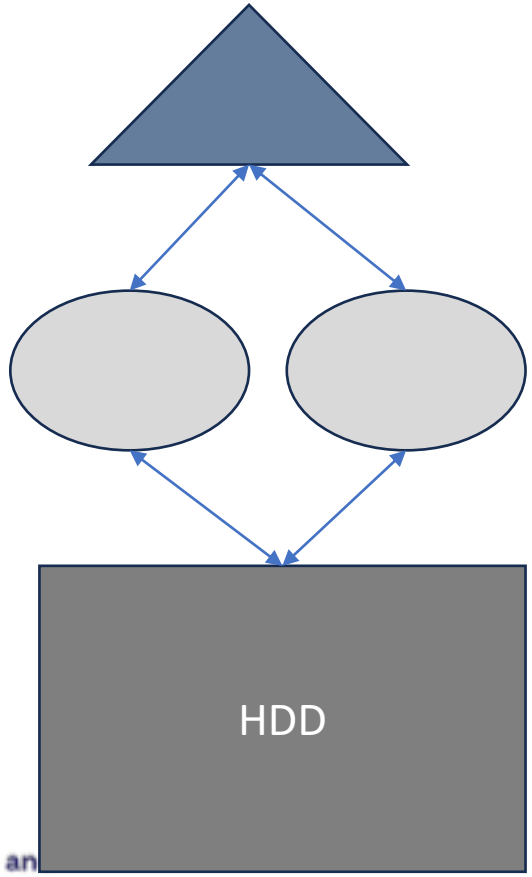

+ storage capacity is maximized
+ storage access HA trough multiple servers
+ throughput can be aggregated with multiple server nodes accessing the same storage system

- Filesystem issues can affect all gateways
- Bandwidth is shared across all users
- Possible race conditions under frequent read/write/delete of the same files

UKRI Science and Technology Facilities Council

# HDD storage



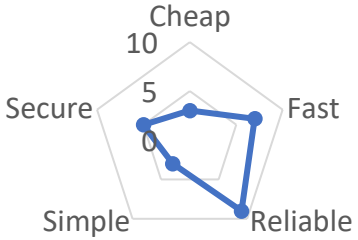

High Priority: CPU, memory, disk capacity, cumulative network

Low priority: disk performance

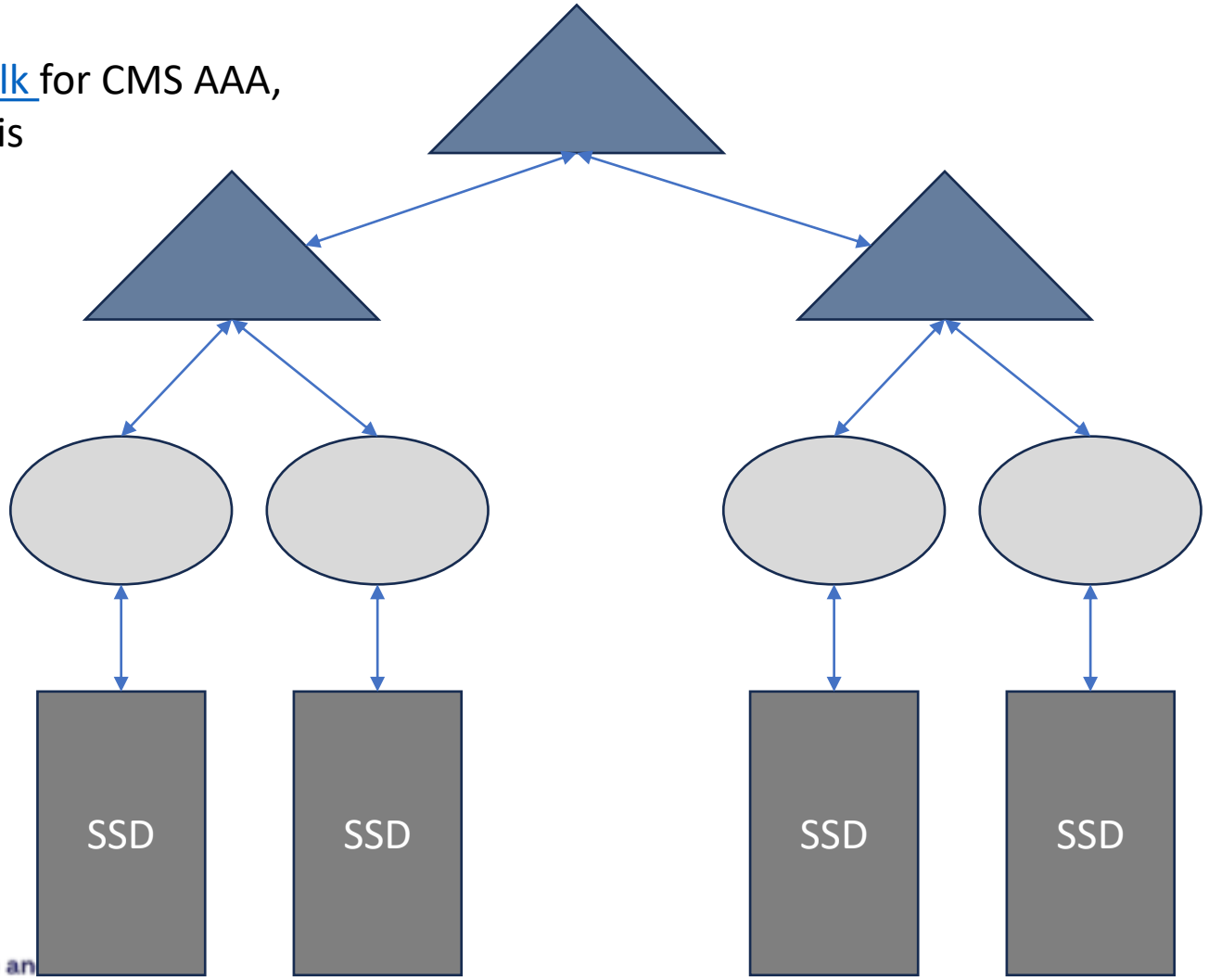
+ similar to DFS
+ cheaper to provision for similar storage capacity
- IOPs limit much lower than SSD
- High vector read-like traffic can exhaust IOPs quite fast

Science and Technology Facilities Council

# Multi-cluster redirects



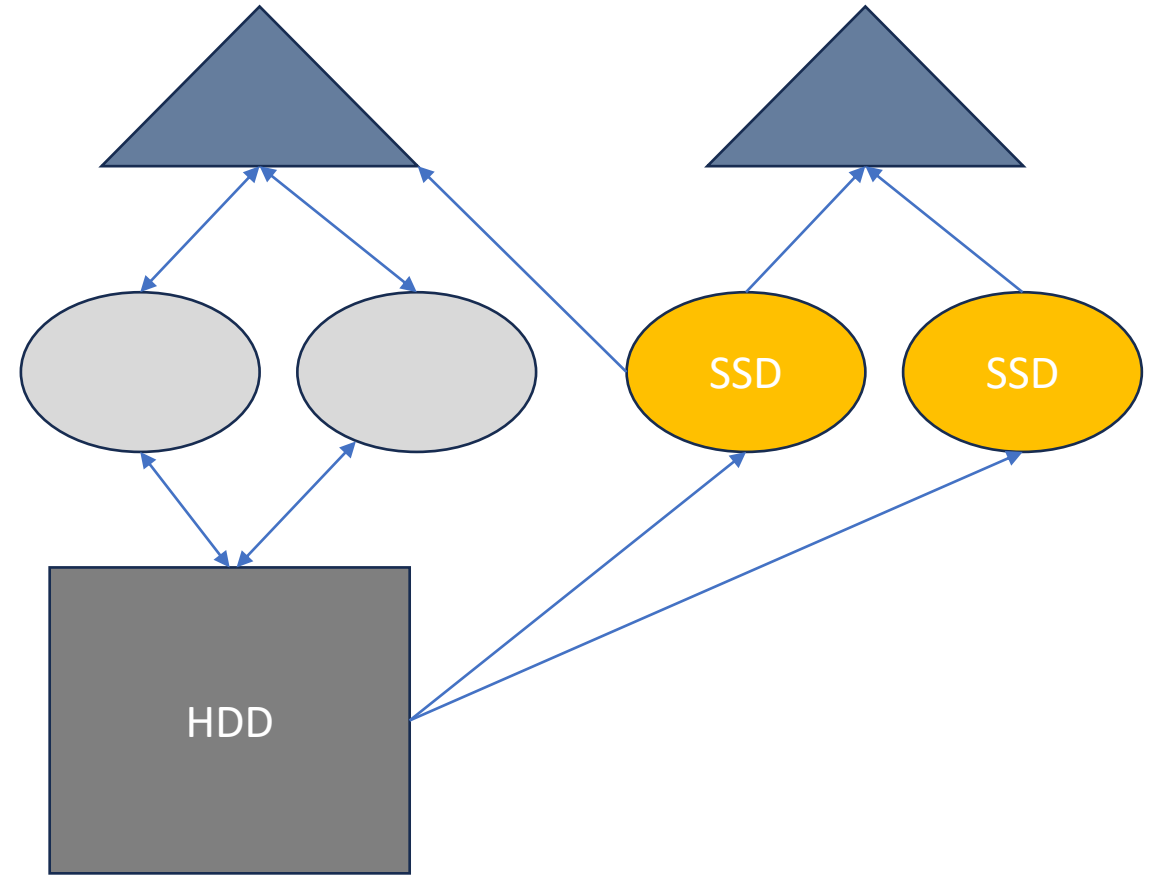

See Rahul's talk for CMS AAA, which uses this

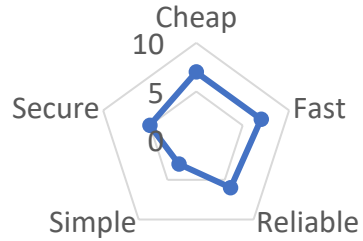



+ allows internal file retrieval across a large storage federation
+ datacentre level failure domain for the service
- High trust requirements
- Functionality can mostly be covered by metadata catalogues and Rucio

Science and Technology Facilities Council

# Satellite storage




+ It's a cache
+ allows to use a smaller SSD capacity as a high-refresh rate storage buffer to absorb vector read traffic
+ see Brian's talk to see how Pelican uses this architecture

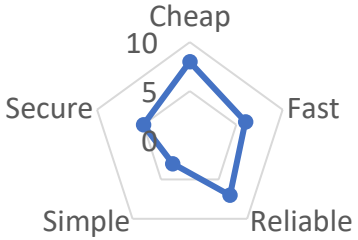
- Writes still need to go to the origin or need variant implementations (e.g. Pelican) which may risk being out of sync with the origin
- Issues on the origin storage will affect all satellite sites when requesting non-cached files

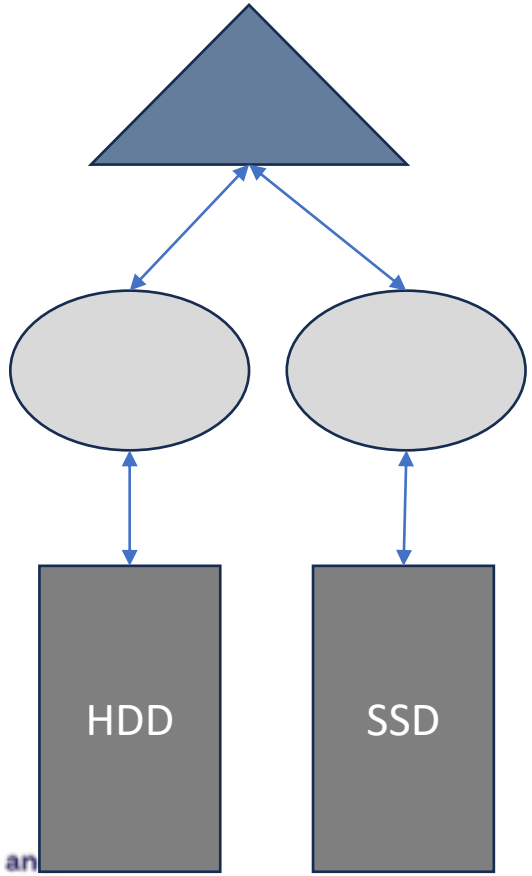
Science and Technology Facilities Council

# And now some wild ideas…

Science and Technology Facilities Council

# Tiered storage

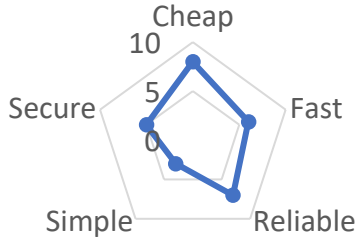


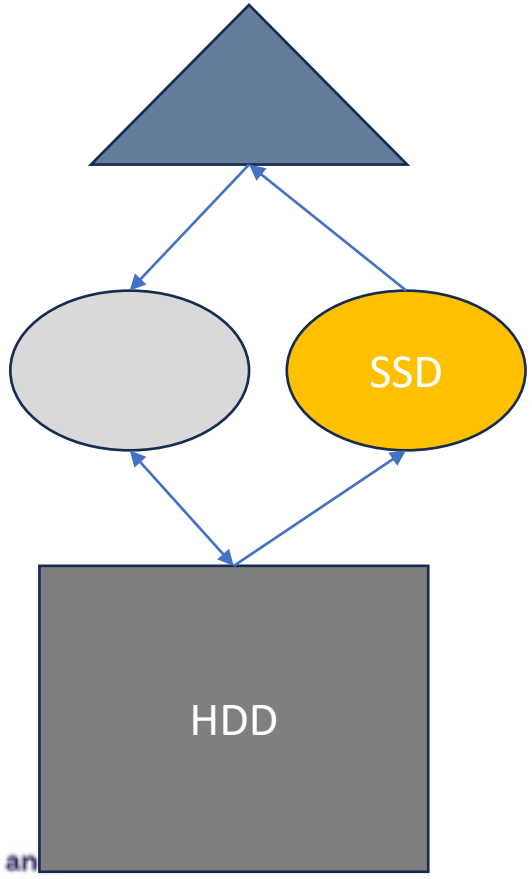




+ small cache for HDD-based storages
+ some level of internal IOPs insurance from vector reads
- Slower read throughput for un-cached reads

UKRI Science and Technology Facilities Council

# Cached reads




High Priority: CPU, disk performance, memory

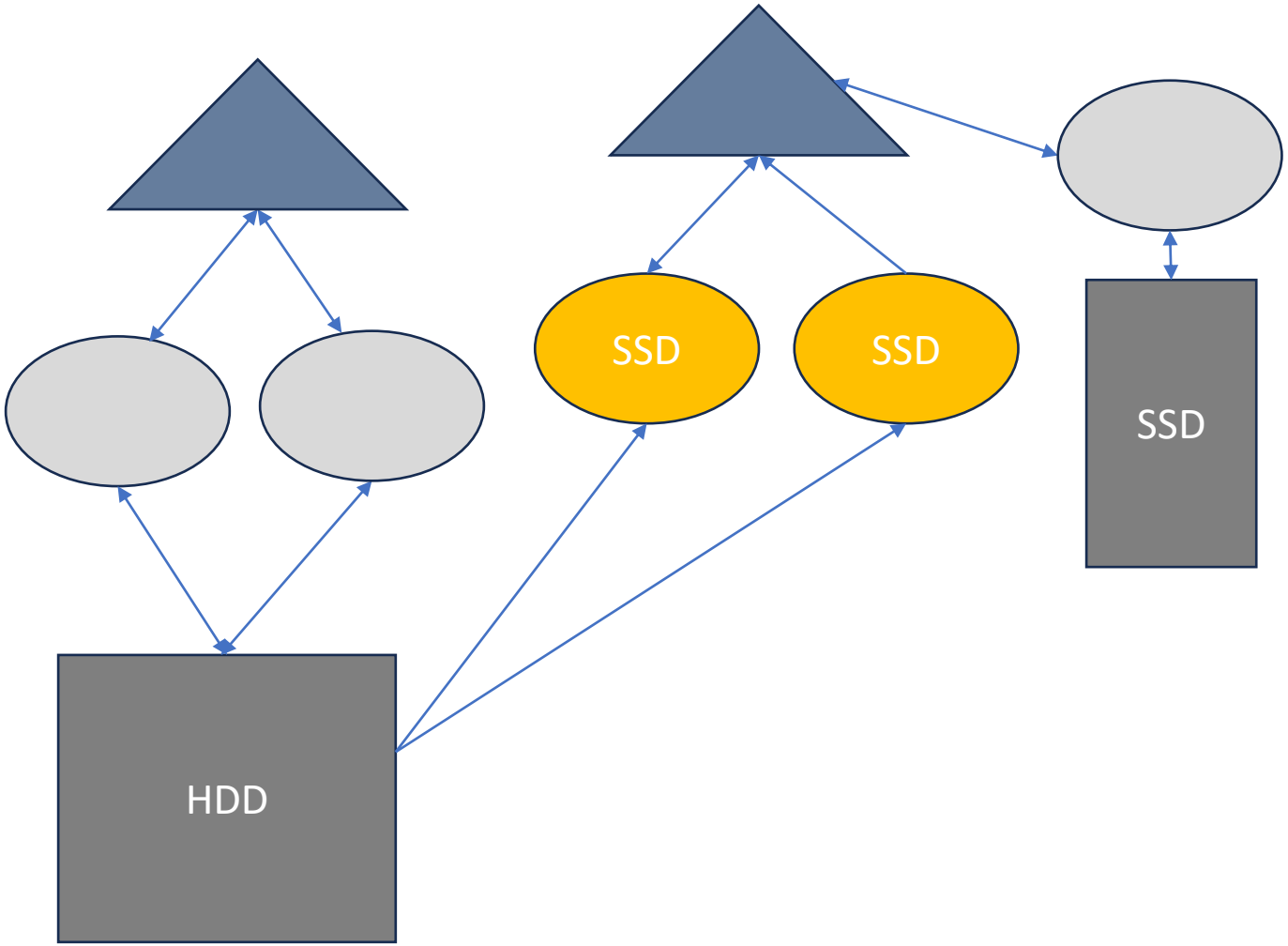
Medium Priority: disk capacity

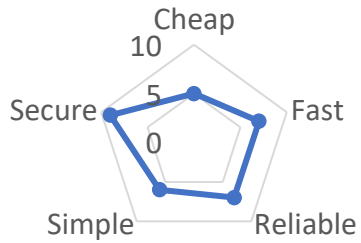
Low priority: Network

+ small cache for HDD-based storages
+ some level of internal IOPs insurance from vector reads
- Slower read throughput for un-cached reads

Science and Technology Facilities Council

# Satellite storage with scratch




+ read-only raw data access from origin, local R/W scratch space for analysis
- Satellite sites may have a strong requirement on having SSDs instead of HDD due to the cache and scratchdisk access patterns

Science and Technology Facilities Council

# Cache Redirect Auth-less Basics

- Maximize use of Caches, especially for reads
- Redirect traffic to balance the load/storage across multiple servers instead of having a monolithic storage and to enable differing read/write flows
- Reduce Auth/TLS loads as much as possible
- Stick to the Basics – keep each site's config as simple as possible

# Future possibilities

- HDD/SSD price ratio
- Minimum storage price changes
- Network/Storage price ratio changes

# Future possibilities

- HDD/SSD price ratio
  - SSD ↑ - smaller caches with higher refresh rates, more intermediary HDD origins
  - HDD ↑ - more cached
- Minimum storage price changes
  - Minimum price ↑ - total storage capacity available will be reduced. Hardware will be kept running for longer. Error resilience and correction at the source become higher priority
  - Minimum price ↓ - better storage lifecycle and capacity growth will be maintained. No significant changes in priority
- Network/Storage price ratio changes
  - Network ↑ - larger caches and smarter routing (SENSE, Pelican) become a priority
  - Storage ↑ - smaller sites, faster data movement between them; data integrity and resilience in flight become a priority

# Conclusions

- XRootD storage architectures are very flexible
- They’re capable of adapting and optimizing on changing hardware types and availability
- Storage requirement priorities on XRootD improvements may change depend on future hardware economy