Data Infrastructure & Hardware

Fundamentals of Data Management

Dr Rob Baxter
Software Development Group Manager, EPCC
r.baxter@epcc.ed.ac.uk
+44 131 651 3579 | +44 7971 437749

Course outline

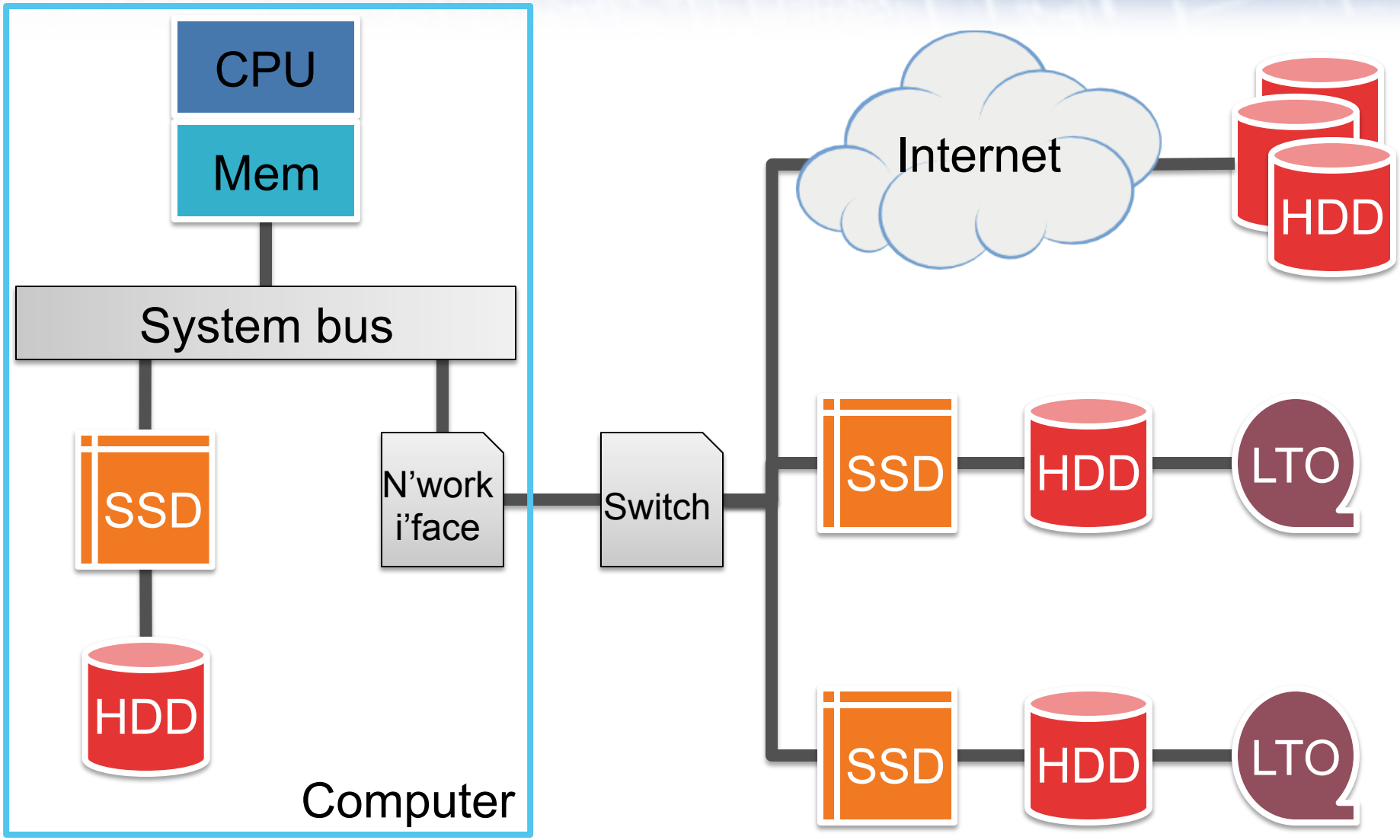
- How is storage hardware keeping up?
- How do modern HPC systems deal with ever-larger data volumes?

- After completing this lesson, you should be able to:
  - Understand the performance challenges of data-driven science
  - Explain what hierarchical storage is and what HSM systems do
  - Understand what Amdahl-balanced machines are and why they have evolved

Archiving vs darkiving

- When it comes to storing data for longer terms (= years), what's most important is *how often is it going to be used?*
- Data stored just for the record may rarely be read again
  - Dark-archiving or "darkiving"
  - Write Once Read Never!
- Other data need to be stored for an ongoing programme of analysis
  - space data, astronomical data, climate data, cultural heritage data, high-energy physics data
  - What we might call "active data"
- "Active data" need very different patterns of storage

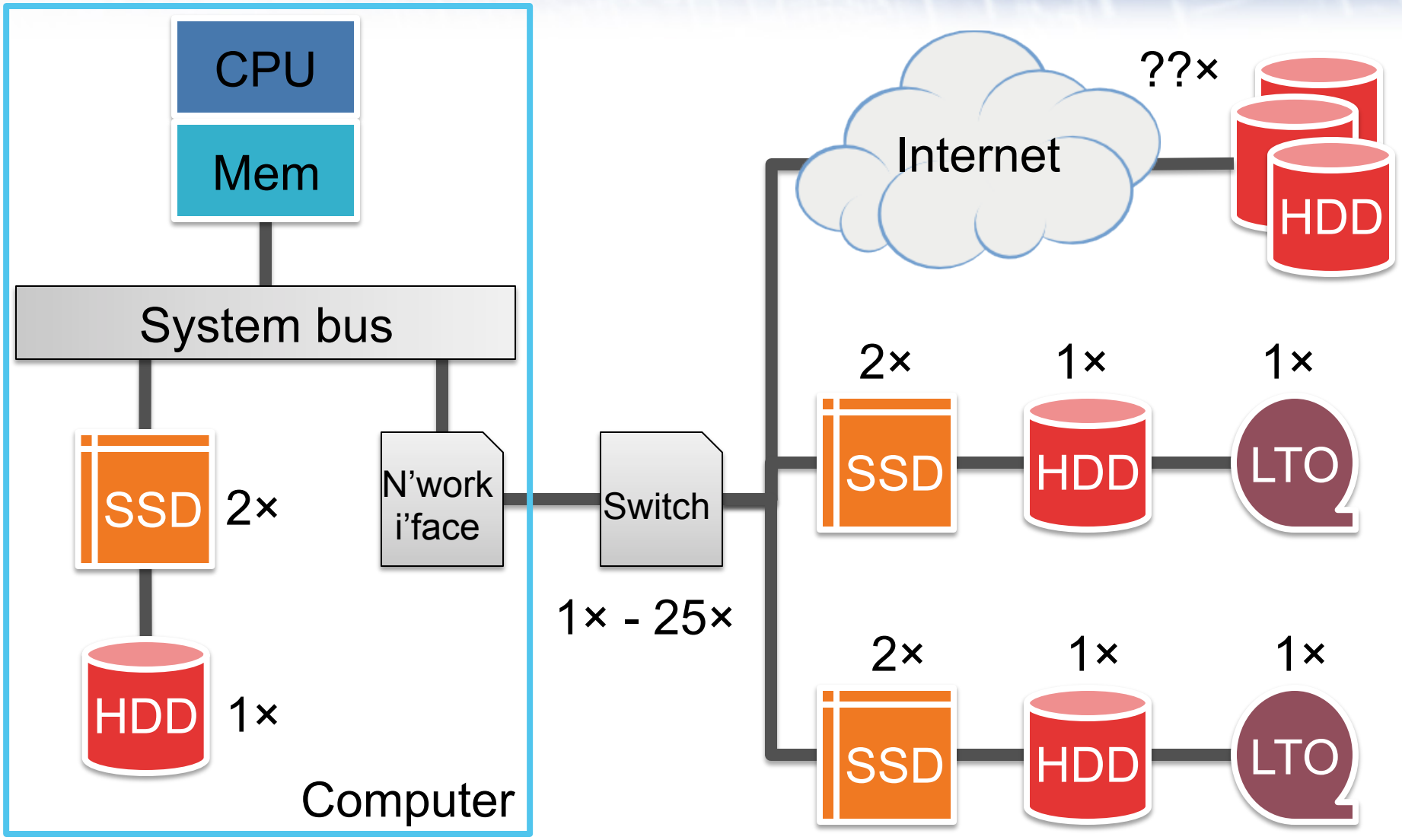
Storage hardware patterns



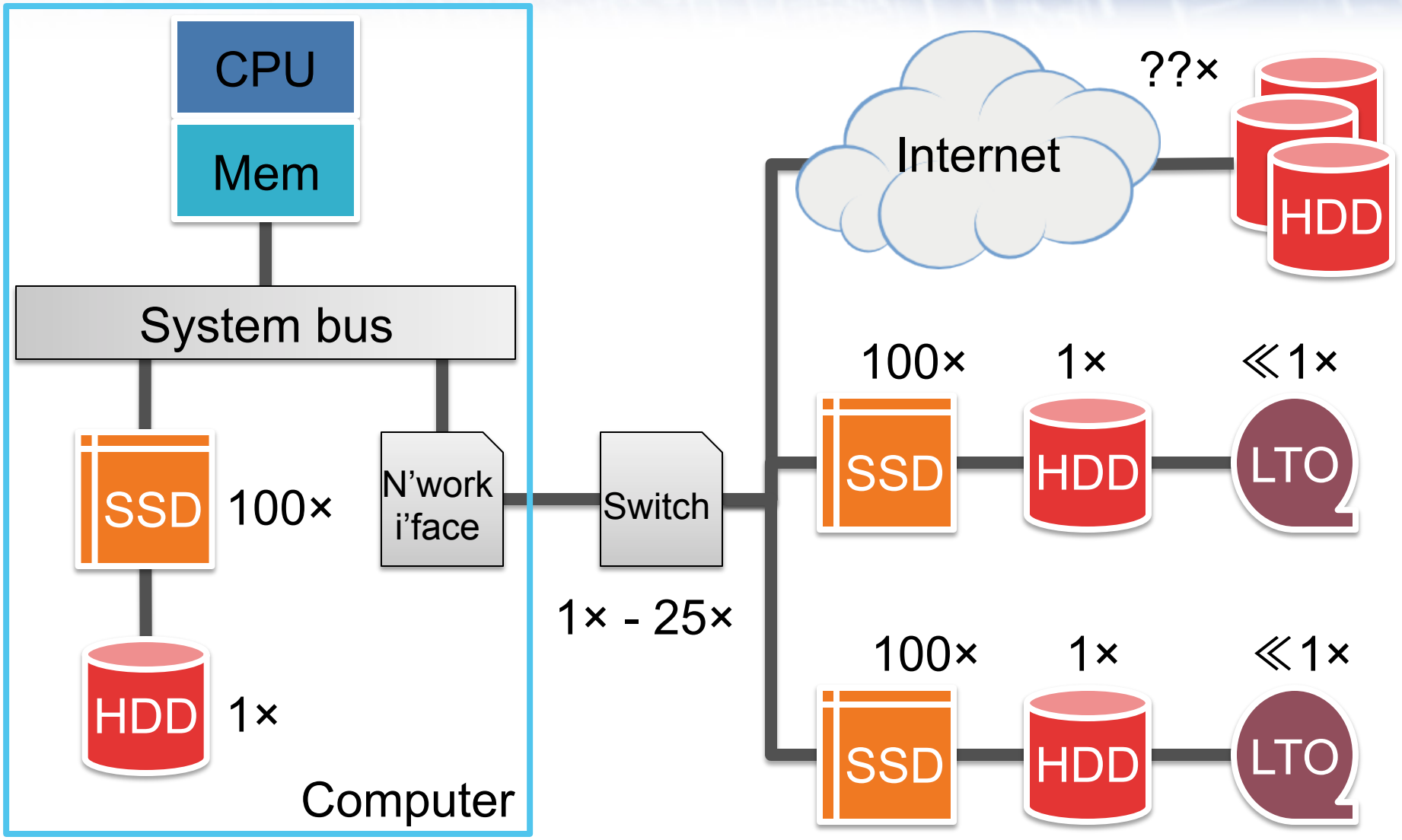
I/o elements of modern systems

- Large-scale computer systems are thus a mix of i/o hardware
- HDD: hard disk drives
  - Spinning, mechanical; access times governed by head seek times
  - Performance is partly a function of the *controller* interface
    - USB, IDE, SCSI/SAS, SATA
  - Frequently connected in *arrays* for resilience & redundancy (RAID)
- SDD: solid state disks
  - Non-volatile flash memory arrays
  - Write lifetime limited; capacity degrades with each overwrite
  - Performance from 2× (sequential read) to 50× (random read) HDD
- LTO: linear tape – open
  - Most common (open) format for modern tape systems
  - Performance comparable with HDD for sequential access (only!)

Sequential performance



Random access performance

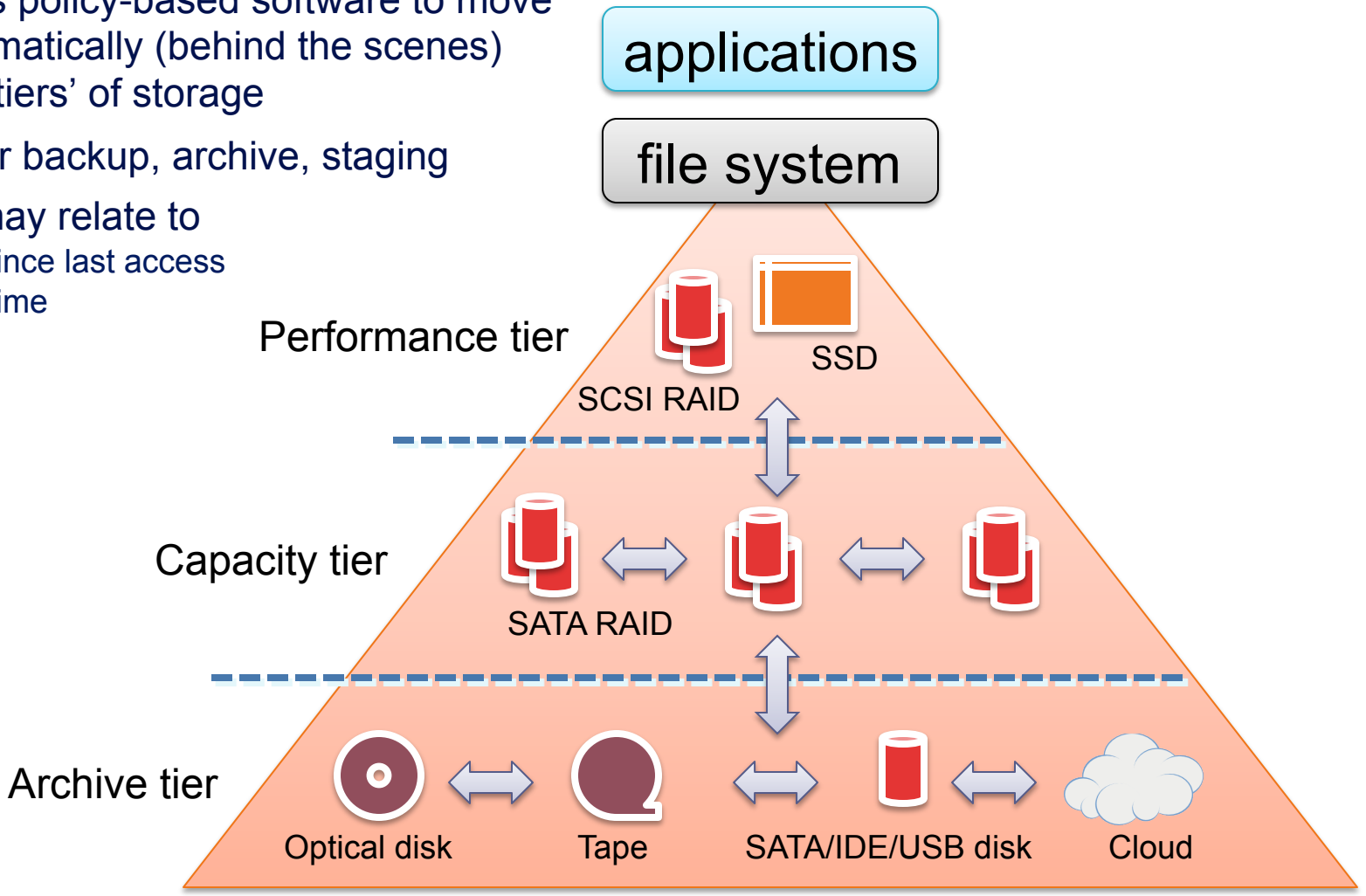


Sidebar: SDN

- With many computers accessing many disks, network bandwidth & switching becomes a real issue
- Big data centres (Amazon, Google, Microsoft) use software defined networking (SDN) to manage this
- Separate network control and routing (control plane) from traffic (data plane)
  - Control plane can then be configured through software, possibly dynamically
  - OpenFlow (en.wikipedia.org/wiki/OpenFlow) is one tech for this
  - Infiniband (high-perf interconnect) uses its own OpenSM SDN tech

Hierarchical storage management

- HSM uses policy-based software to move data automatically (behind the scenes) between 'tiers' of storage
- May be for backup, archive, staging
- Policies may relate to
  - Time since last access
  - Fixed time
  - Events



High-performance file systems

- GPFS – IBM's General Parallel File System
  - Software defined storage layer
  - Single file system namespace across (widely) distributed storage
  - Can use (almost) any storage hardware underneath (flash, disk, tape)
  - Supports flash-based storage of file system metadata
  - Parallel file access well-suited to HPC
- Lustre – open source parallel file system (lustre.opensfs.org)
  - Similar (in many ways) to GPFS
  - "a bit faster but a bit less reliable"
- HDFS – Hadoop file system (hadoop.apache.org/hdfs)
  - A Java-based middleware layer for distributing data in chunks across cluster storage
  - Underpins Hadoop/MapReduce type computational patterns
  - Combines distributed file access with redundancy

Performance just got harder…

- Need to understand the performance of your computations *and your data transfers*
- Do you know how fast your program runs?
- Do you know if it's spending all of its time on compute or if it's spending its time waiting for data?
- Where is your data bottleneck?
- *Benchmarking is key*
  - IOZone (iozone.org) – good for single client I/O throughput
  - IOR (github.com/chaos/ior) – good for parallel scalability
- but your own application will always give you the best measure

Data Intensive Computing

Computing applications which devote most of their execution time to computational requirements are deemed compute-intensive and typically require small volumes of data, whereas computing applications which require large volumes of data and devote most of their processing time to I/O and manipulation of data are deemed data-intensive. – **Wikipedia**

- Our working definition:
  - *I/O-bound computations*
- Data are (generally) too big to fit in memory
  - Efficient disk access is required to get the data to the CPU on time
  - Having the data in the right place at the right time is *vital*

Amdahl's *other laws*

- Gene Amdahl's quantification of the balance required for data-intensive applications (based on empirical observation):
- **Amdahl Number**: IO bandwidth (bits/s) / CPU clock rate = 1
  - Or, one bit of disk IO per compute cycle
- **Amdahl Memory Ratio**: Memory size (bytes) / instructions per second = 1
  - Assumes memory bandwidth > IO rate (generally true)
- **Amdahl IOPS Ratio**: IO bandwidth × 50,000 / CPU clock rate = 1
  - Or, one IO-Op per 50,000 instructions

See Amdahl, G. (2007). Computer Architecture and Amdahl's Law.

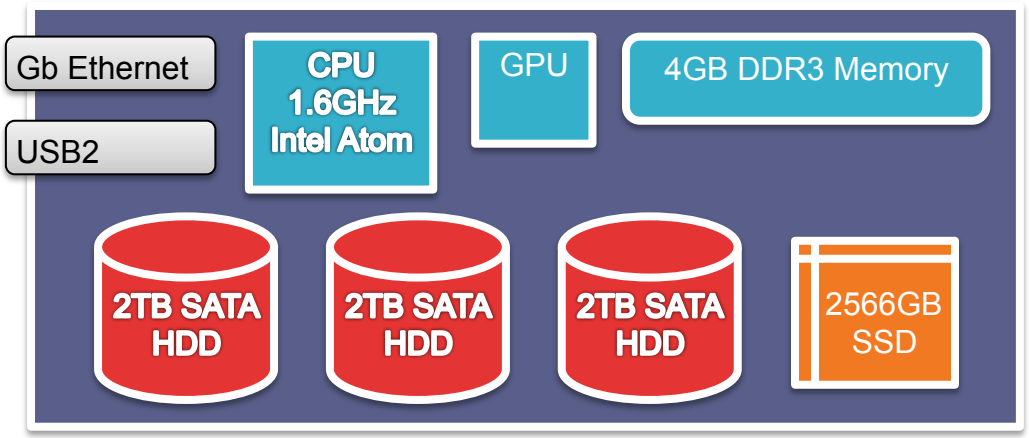
IEEE Solid State Circuits Society News, pp 4-9

http://ewh.ieee.org/soc/sscs/images/newsletter_archive/sscs_newsletter_200707.pdf

Building an Amdahl-balanced system

1. Scale-up the IO throughput to match the CPU clock
   a. Add high-performance SAN to fast compute nodes
      – Expensive, in £ and Watts!
      – Quite soon hit a network bandwidth bottlenecks
   b. Add disk arrays & high-end IO controllers
      – Expensive, in £ and Watts!
      – Quite soon hit IO bus bandwidth bottlenecks (eg. PCIe)
2. Power down the processors to match the IO throughput
   – Jim Gray's original idea: the "data brick"
   – Use slower processors, commodity kit
   – Aim for an Amdahl Number of 1

A Data Intensive Computer



× 120

Aggregate thru'put (MB/s)	Aggregate IOPS (/s)	Amdahl Number	Amdahl Memory Ratio	Amdahl IOPS Ratio
506.25	60642	1.33	1.28	0.95

Amdahl numbers – examples

CPU (GHz)	Mem (GB)	seqIO (GB/s)	randIO (kIOPS)	Disk (TB)	Amdahl Number	Amdahl Mem Ratio	Amdahl IOPS Ratio	
3.200	4.000	0.500	10.400	0.500	1.250	1.250	0.163	SATA SSD, 2 core Intel Atom
3.200	4.000	1.500	104.000	0.500	3.750	1.250	1.625	PCIe SSD, 2 core Intel Atom
3.200	4.000	0.500	0.140	0.500	1.250	1.250	0.002	SATA 10k HDD
64.800	64.000	2.400	-	-	0.296	0.988	0.000	Cray XC30 node, 24 core Intel
4.800	8.000	1.500	100.000	0.250	2.500	1.667	1.042	MacBook Pro, PCIe SSD, 2 core Intel i5

Making best use of a machine designed for data-intensive computing

- Work on streams of data, not files
  - Not (so easily) searchable
  - Not (so easily) sortable
  - Not all programs can benefit from this approach
    - and those that can, might require work
- Use multiple threads and asynchronous I/O
- If you're using files, use a library that does some of the hard work for you, e.g. MPI-IO

Data Intensive Computation as a new way to do Research

Datascope

- Look for patterns in large amounts of data
- Data mining
- You don't know what you're looking for when you collect the data
  - Or you're using others' data in novel ways

The role of data infrastructures in data intensive computing

- Traditionally, we bring the data to the compute
- In the future, we'll want to bring the compute to the data
  - So where is the data?
  - More than likely it's in a repository…
  - Maybe an "archive"…

How might you bring compute to data?

- As (relational) database queries (SQL)
- As queries against an RDF store (SPARQL)
- As VMs which can mount local disks
- As scripts or executables that you allow a user to run
- As services that you as a data service offer with some kind of API (e.g. as a web service)

Compute + data

- These are the approaches that will need to be offered by “repositories” holding large amounts of “live” data.
  - Many will probably also be relevant to archives
- “Data infrastructures” are combining large data with large compute
- How can a user get the information back out of the archive?
  - As complete files?
  - Over the Internet (and your network!)?

Summary

- Hardware, infrastructure for data-driven computing adds extra dimensions to program design, performance
- Need to think about new ways of doing computing
  - It's usually parallel computing, but not "traditional HPC"
- Data-Intensive is a new(ish) kind of computing
  - necessitated by the huge amounts of data
  - and offering new opportunities
- Matters for data preservation. Either:
  - you're preserving huge amounts of data that need to be easily reused
  - you need to process large amounts of data to do a meaningful reduction so that the stored data retains its value

Acknowledgements

- Includes material created originally for the EUDAT project
  - Data Intensive Computing
    - Adam Carter, EUDAT. Retrieved August, 2014.
    - http://www.eudat.eu/training-materials-downloads