

**HPS**

<https://hps.vi4io.org>

Julian Kunkel

## Performance Estimation

# Learning Objectives

- Describing relevant performance factors for systems
- Listing peak performance of relevant components
- Assessing/Judging observed performance

# Outline

- 1 Introduction
- 2 System Characteristics
- 3 Assessing Performance
- 4 Example: Parallel FS
- 5 Summary

# Motivation

- Admins must know basic performance aspects to design suitable systems
  - ▶ Capacity planning (how many servers are needed)
  - ▶ Optimizing systems (higher efficiency)
- Goal (system perspective):
  - ▶ Efficiency: Good utilization of (hardware) resources means less hardware
  - ▶ Cheap hardware, i.e., less performance
  - ▶ (Simple deployment and easy management)
  - ▶ (Security + Privacy + Compliance with laws)
- User perspective: Minimal time to solution, easy to use

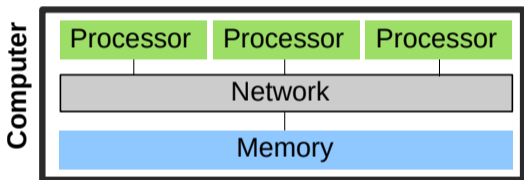
# Outline

- 1 Introduction
- 2 System Characteristics**
  - HPC Clusters
  - Software
- 3 Assessing Performance
- 4 Example: Parallel FS
- 5 Summary

# Parallel & Distributed Architectures

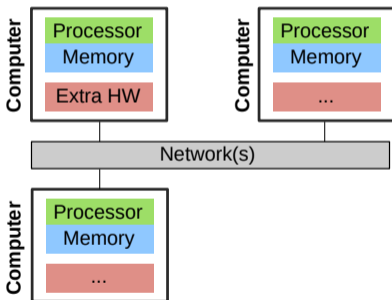
In practice, systems are a mix of two paradigms:

## Shared memory



- Processors access joint memory
  - ▶ Communication/coordination
- Cannot be scaled up to any size
- Expensive to build big system

## Distributed memory systems



- Processor see only own memory
- Performance of the network is key

## Example: Characteristics of an HPC Cluster

- High-end components
- Extra fast interconnect, global/shared storage with dedicated servers
- Network provides high (near-full) bisection bandwidth.

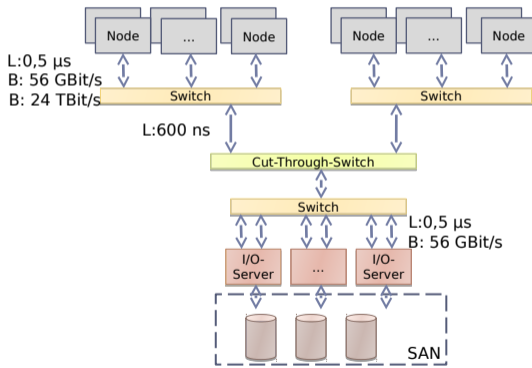


Figure: Architecture of a typical HPC cluster (here fat-tree network topology)

# Hardware Performance

## Computation

- CPU performance (frequency  $\times$  cores  $\times$  sockets)
  - ▶ E.g.: 2.5 GHz  $\times$  12 cores  $\times$  2 sockets = 60 Gcycles/s
  - ▶ The number of cycles per operation depend on the instruction stream
- Memory (throughput  $\times$  channels)
  - ▶ E.g.: 51.2 GiB/s per DDR5 module  $\times$  8 channels (AMD Epyc) = 400 GiB/s

## Communication via the network

- Throughput, e.g., 1250 MiB/s with 10 GbE Ethernet
- Latency, e.g., 0.1 ms with Gigabit Ethernet

## Input/output devices

- Access data consecutively and not randomly
- Performance depends on the I/O granularity
  - ▶ E.g.: HDDs 150 MiB/s with 10 MiB blocks, even Flash suffers by small access



# Influence of Software on Performance

- Allow monitoring of components to detect overloaded services
  - ▶ For instance, using Grafana, Prometheus, ...
- Java: 1.2x - 2x of cycles compared to C<sup>1</sup>
- Balance and distribute workload among all available servers/services
  - ▶ Linear scalability of the solution is important
  - ▶ Add 10x servers, achieve 10x performance (or process 10x data)
- Avoid I/O, if possible and keep data in memory
- Host depending services locally

---

<sup>1</sup> This does not matter much compared to the other factors. But vectorization matters.

# Outline

- 1 Introduction
- 2 System Characteristics
- 3 Assessing Performance**
  - Basic Approach
- 4 Example: Parallel FS
- 5 Summary

# Strategy

## Guiding question

Is the observed performance *acceptable*?

- My observation: often a simple approximative model is sufficient
  - ▶ Knowing that something is 100x slower than it should be...
- You must understand the basic architecture of the software system
- You must understand most important hardware characteristics
- Advice
  - ▶ Start with simple models for workload and hardware performance
  - ▶ Refine the model as needed, e.g., include details about intermediate steps

## Approximation – Simple Example on Computation

Example: Summing up data in an array of 10M ints

- Workload: 10M integers
- System: 3.7 GHz PC
- Python (for loop): 0.39s = 98 MB/s, 144 cycles per op  
 $(10 \cdot 1000 \cdot 1000) \cdot 4 \text{ bytes} / 0.39\text{s} = 98\text{MiB/s}$   
 $3700 \cdot 1000 \cdot 1000\text{cycles} \cdot 0.39\text{s} / (10 \cdot 1000 \cdot 1000\text{op}) = 144 \text{ cycles/op}$
- Numpy: 0.0055s, 7000 MB/s, 2 cycles per op
- Python (sum up numbers): 0.14s, 272 MB/s, 52 cycles per op
- One line to measure the performance in Python using Numpy:

```

1  timeit.timeit(stmt="np.sum(d)", setup="import numpy as np; d =
    ↪ np.array(range(1,10*1000*1000))", number=1)
2  # Just sum up numbers: sum(range(1,10*1000*1000))

```

# Methodology

- 1 Measure time for the execution of your workload
- 2 Quantify the workload with some metrics
  - ▶ E.g., amount of tuples or data processed, computational operations needed
  - ▶ E.g., you may use the statistics output for each Hadoop job
- 3 Compute  $W$ , the workload you process per time
- 4 Compute expected performance  $P$  based on system's hardware characteristics
- 5 Compare  $W$  with  $P$ , the efficiency is  $E = \frac{W}{P}$ 
  - ▶ If  $E \ll 1$ , e.g., 0.01, you are using only 1% of the potential!

## Example: Object Storage

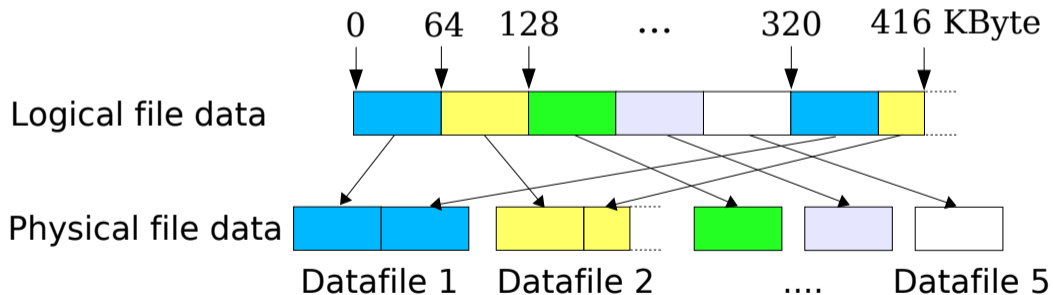
### Scenario: Accessing data on object storage

- 1 Time: 0.1s (3x measured, between 0.09 and 0.11s)
- 2 Workload: 100 MiB of data fetched from object storage
- 3  $W = 100\text{MiB}/0.1\text{s} = 1000\text{MiB}/\text{s}$
- 4 System: Client and server are interconnected via a 100 GbE network  
Characteristics:  $P = 12,500\text{GiB}/\text{s}$  throughput  
Latency doesn't matter for large files
- 5 Efficiency:  $E = 1,000/12,500 = 8\%$

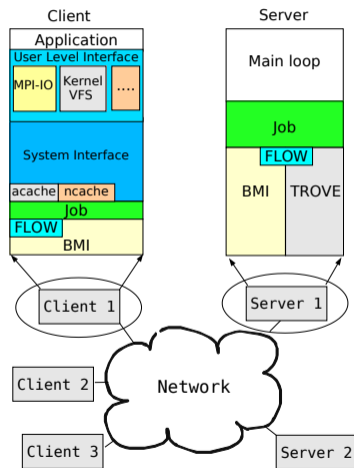
For a 10 GbE interconnect, 80% efficiency would have been achieved!

## Example: Parallel File System

- Workload: Reading/writing X amount of data from a parallel file system
- One file is distributed across multiple datafiles and servers



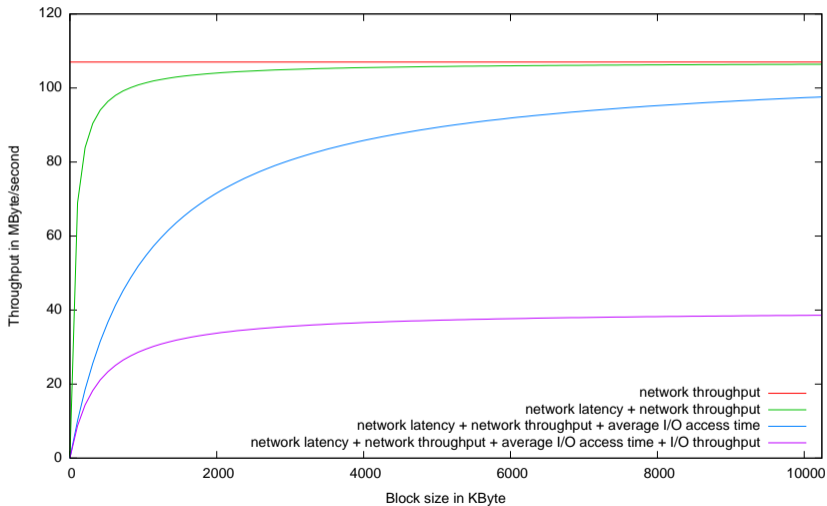
# Parallel File System Architecture: Here PVFS2



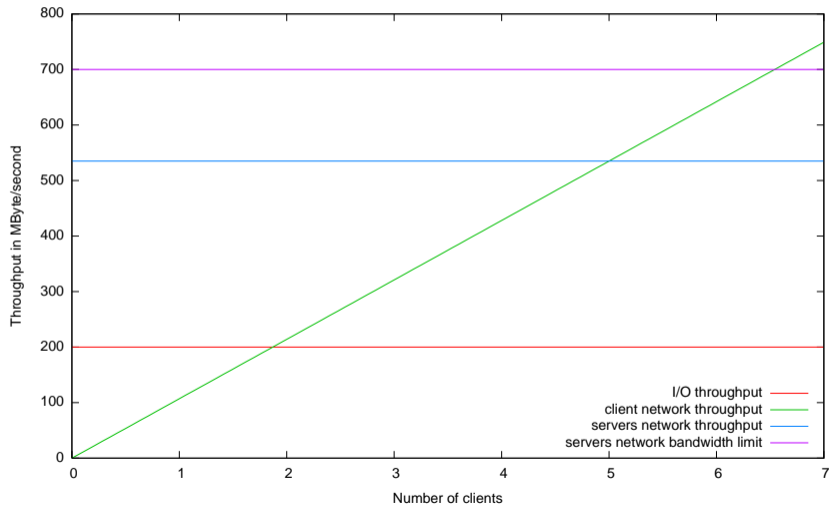
- We can ignore the layers
- C clients connect to S servers
- Clients may access the same file Concurrently - at the same time
- System: GbE Ethernet, HDDs with 40 MiB/s
- Let's build performance models! Start from a simple model and refine



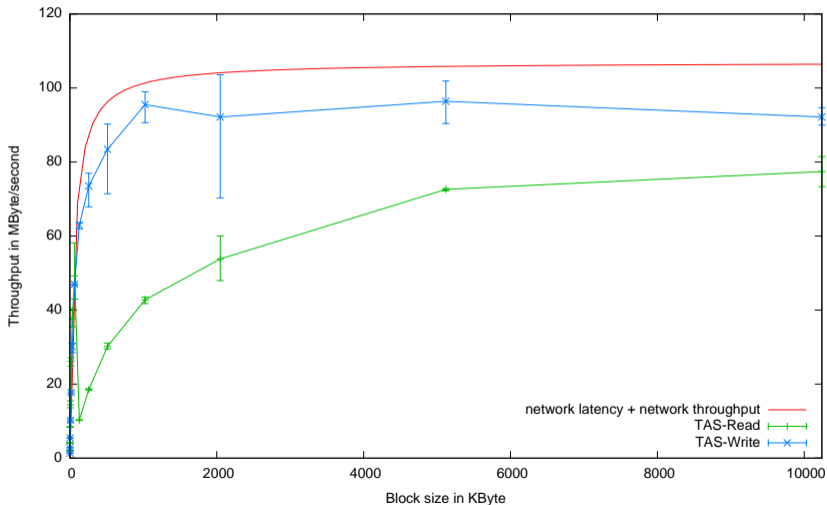
## Small I/O Access (Single Client)



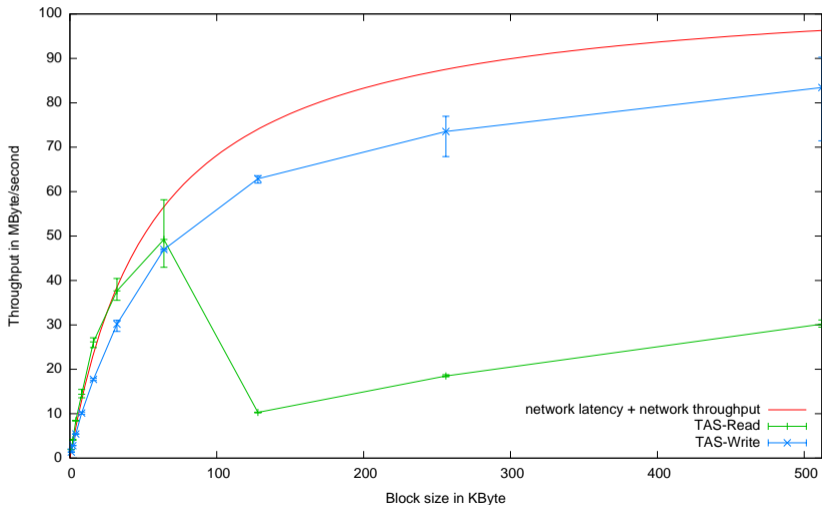
# Large Access (Multiple Clients)



# Actual Measured Performance (Single Client)



# Actual Measured Performance (Single Client) - Small Block Sizes



# Summary

- Understanding hardware characteristics helps to assess performance
- Basic performance analysis
  - 1 Estimate the workload
  - 2 Compute the workload throughput per node
  - 3 Compare with hardware capabilities
- Exercise: You'll do an own performance estimation!