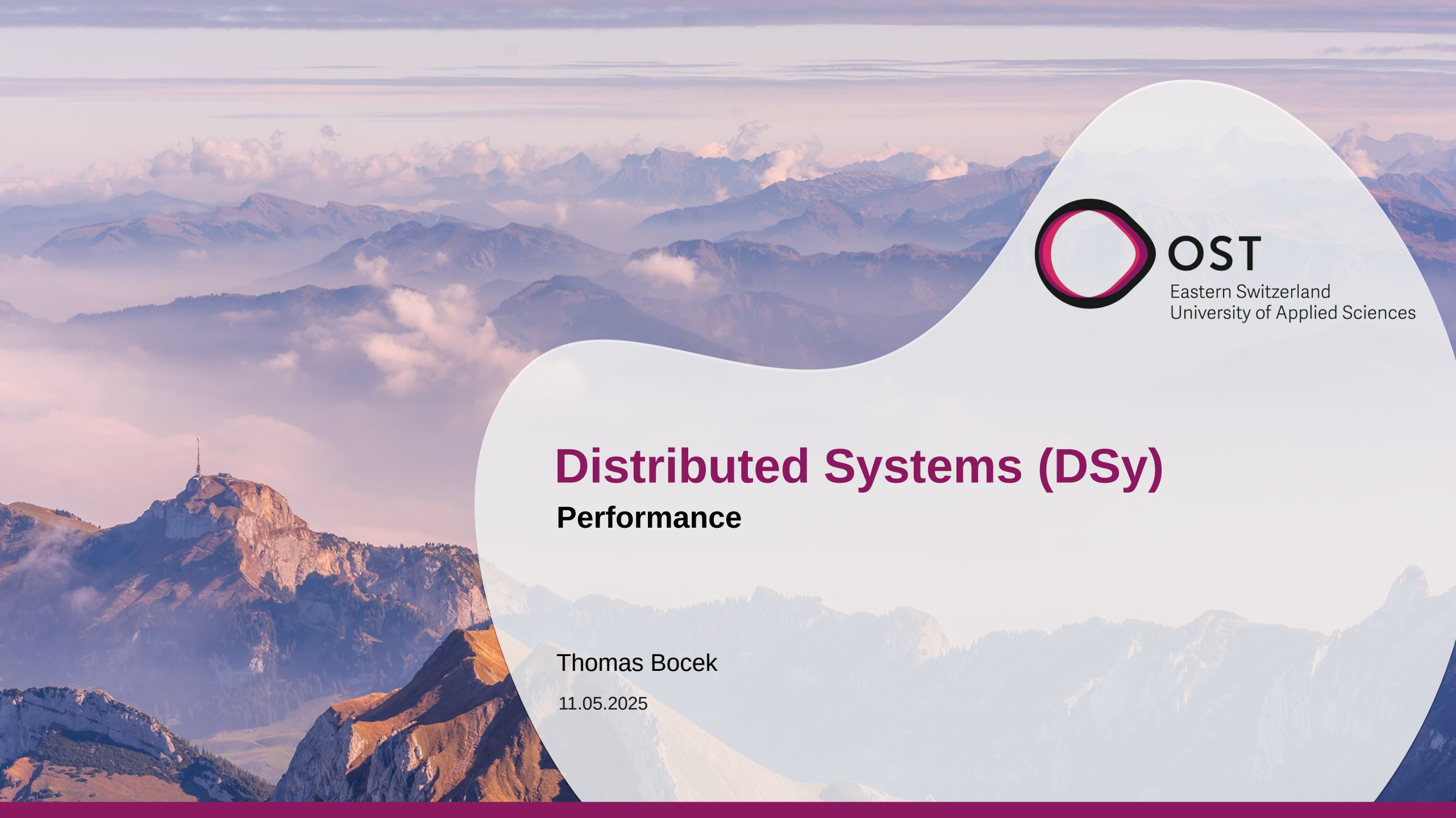

OST
Eastern Switzerland
University of Applied Sciences

Distributed Systems (DSy)

Performance

Thomas Bocek

11.05.2025

Learning Goals

- Lecture 12 (Performance)
  - Get a feeling how fast it runs

Latency Numbers Every Programmer Should Know

- Interactive [link] from 1990 - 2020
  - Network stays ~ 150ms
  - L1: 1ns / branch miss 3ns – example
- HDD / SSD / NVMe (Non-Volatile Memory Express) - comparison, 2

NVMe Latency vs. Other Storage Protocols

Feature	NVMe	SATA SSD	HDD
Read Latency	~20 µs	~100 µs	2-5 ms
Write Latency	~30 µs	~200 µs	5-10 ms
Queue Depth	64K queues × 64K commands	32 commands	1 command
Bandwidth	Up to **16GB/s (PCIe 4.0)**	600MB/s	~150MB/s

- Latency and throughput important
- Napkin Math [link]
  - Cost

Operation	Latency	Throughput	1 MiB	1 GiB
Sequential Memory R/W (64 bytes)	0.5 ns			
├ Single Thread, No SIMD		10 GiB/s	100 µs	100 ms
├ Single Thread, SIMD		20 GiB/s	50 µs	50 ms
├ Threaded, No SIMD		30 GiB/s	35 µs	35 ms
├ Threaded, SIMD		35 GiB/s	30 µs	30 ms
Network Same-Zone		10 GiB/s	100 µs	100 ms
├ Inside VPC		10 GiB/s	100 µs	100 ms
├ Outside VPC		3 GiB/s	300 µs	300 ms
Hashing, not crypto-safe (64 bytes)	25 ns	2 GiB/s	500 µs	500 ms
Random Memory R/W (64 bytes)	50 ns	1 GiB/s	1 ms	1s
Fast Serialization [8] [9] †	N/A	1 GiB/s	1 ms	1s
Fast Deserialization [8] [9] †	N/A	1 GiB/s	1 ms	1s
System Call	500 ns	N/A	N/A	N/A
Hashing, crypto-safe (64 bytes)	100 ns	1 GiB/s	1 ms	1s
Sequential SSD read (8 KiB)	1 µs	4 GiB/s	200 µs	200 ms
Context Switch [1] [2]	10 µs	N/A	N/A	N/A
Sequential SSD write, -fsync (8KiB)	10 µs	1 GiB/s	1 ms	1s
TCP Echo Server (32 KiB)	10 µs	4 GiB/s	200 µs	200 ms
Decompression [11]	N/A	1 GiB/s	1 ms	1s
Compression [11]	N/A	500 MiB/s	2 ms	2s
Sequential SSD write, +fsync (8KiB)	1 ms	10 MiB/s	100 ms	2 min
Sorting (64-bit integers)	N/A	200 MiB/s	5 ms	5s
Sequential HDD Read (8 KiB)	10 ms	250 MiB/s	2 ms	2s

OST

Latency Numbers Every Programmer Should Know

- Compression Ratios (from napkin-math)
  - HTML: 2-3x
  - Source Code: 2-4x
- Compression benchmarks [link]
  - enwik8: 100MB, enwik9: 1GB
- Compression speed vs. ratio, read vs. write
- HTTP: gzip, brotli, zstd [link]
  - Brotli: optimized for web content
    - HTML/XML elements, CSS properties, JavaScript keywords, HTTP headers, URI/URL, JSON/XML, common words
    - Dictionary in Brotli algorithm, gzip, zstd not
  - zstd: General purpose, better than gzip, faster and better ratio
- Best practice: enable compression, default ration (only tune after measurement)
  - Or you have a static site, e.g., PrevelteKit → maxed out (zopfli)
- Image compression, choose
  - Jpeg (guetzli, mozjpeg), jpeg xl, avif, webp, png, gif…
  - Image as SVG → brotli, Google in 2015: logo in 305bytes – in 146 bytes
  - Comparison

Latency Numbers Every Programmer Should Know

- How many requests / sec can your site handle?
  - Artillery [link] - Looks impressive, but renders page on client, client may become bottleneck
  - Benchmarks via Wifi, testing the network bottleneck
  - Benchmark via new connections, testing the slow start
  - Benchmark the login page. Testing the computational complexity of bcrypt
- Example repository, simple page, golang, PostgreSQL DB
- Apache Benchmark
  `ab -n 10000 -c 50 -k https://dsl.i.ost.ch/`
  - localhost – fast machine 13k req/s, slow machine 2.8k req/s
    - Get your baseline
  - dsl.i.ost.ch – testing my instance, no DB, 6k req/s
  - If you don’t get 1k req/s, something is wrong
- Best practice
  - Make it work, make it correct, make it fast [link]
  - Premature optimization is the root of all evil [link]
  - Only measure and optimize your use-case