Technische Universität Braunschweig

Institute of Operating Systems and Computer Networks
Reliable System Software

CumulusDB: Cloud-Native Databases and Unikernels

A Vision for Kernel-Integrated Application Co-Design

Christian Dietrich

March 14th, 2024

The Anabasis: Layers all the Way Down

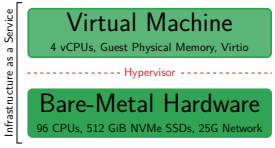
- Modern hardware is tremendously capable
  - Up to ~100 cores, 100s of GiB DRAM
  - >1M 4KiB-random reads/s, 100/25G network
  - It is hard to fully exploit it

Bare-Metal Hardware
96 CPUs, 512 GiB NVMe SSDs, 25G Network

"Anabasis: Der Zug der Zehntausen",
Xenophon, 370 BCE

The Anabasis: Layers all the Way Down

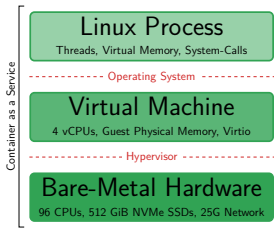
- Modern hardware is tremendously capable
  - Up to ~100 cores, 100s of GiB DRAM
  - >1M 4KiB-random reads/s, 100/25G network
  - It is hard to fully exploit it



- Cloud providers slice up these machines
  - Virtual machines isolate customers

"Anabasis: Der Zug der Zehntausen",
Xenophon, 370 BCE

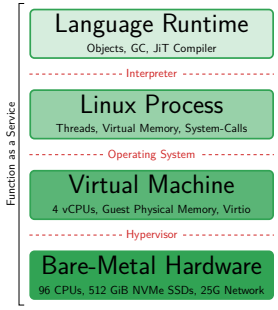
The Anabasis: Layers all the Way Down



- Modern hardware is tremendously capable
  - Up to ~100 cores, 100s of GiB DRAM
  - >1M 4KiB-random reads/s, 100/25G network
  - It is hard to fully exploit it
- Cloud providers slice up these machines
  - Virtual machines isolate customers
  - Containers isolate applications

"Anabasis: Der Zug der Zehntausen",
Xenophon, 370 BCE

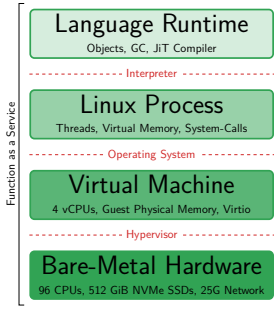
The Anabasis: Layers all the Way Down



- Modern hardware is tremendously capable
  - Up to ~100 cores, 100s of GiB DRAM
  - >1M 4KiB-random reads/s, 100/25G network
  - It is hard to fully exploit it
- Cloud providers slice up these machines
  - Virtual machines isolate customers
  - Containers isolate applications
  - Run times isolate pointers

"Anabasis: Der Zug der Zehntausen",
Xenophon, 370 BCE

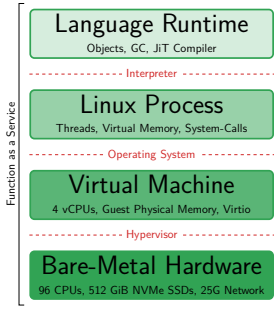
The Anabasis: Layers all the Way Down



- Modern hardware is tremendously capable
  - Up to ~100 cores, 100s of GiB DRAM
  - >1M 4KiB-random reads/s, 100/25G network
  - It is hard to fully exploit it
- Cloud providers slice up these machines
  - Virtual machines isolate customers
  - Containers isolate applications
  - Run times isolate pointers
- Benefits for average applications
  - \+ Simpler to develop and to deploy
  - \+ Cloud providers can co-locate more customers
  - – **We pay** for these abstractions

"Anabasis: Der Zug der Zehntausen", Xenophon, 370 BCE

The Anabasis: Layers all the Way Down



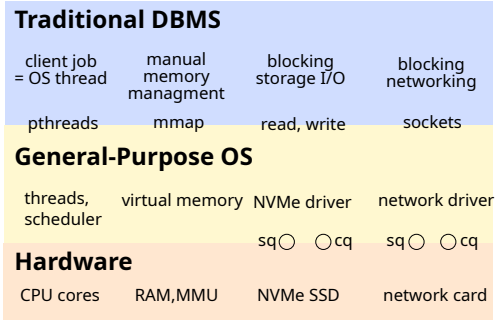
- Modern hardware is tremendously capable
  - Up to ~100 cores, 100s of GiB DRAM
  - >1M 4KiB-random reads/s, 100/25G network
  - It is hard to fully exploit it
- Cloud providers slice up these machines
  - Virtual machines isolate customers
  - Containers isolate applications
  - Run times isolate pointers
- Benefits for average applications
  - \+ Simpler to develop and to deploy
  - \+ Cloud providers can co-locate more customers
  - − **We pay** for these abstractions

⇒ **But some applications are experts!**

"Anabasis: Der Zug der Zehntausen", Xenophon, 370 BCE

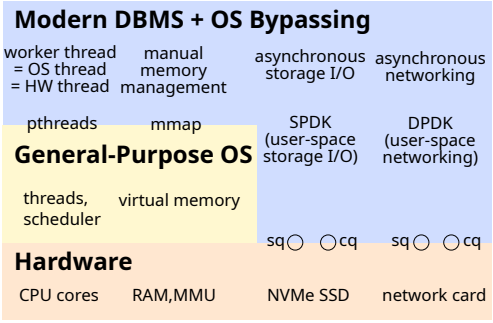
The Katabasis: From Traditional DBMS to CumulusDB

- Traditional Database Management Systems
  - Standard POSIX API tames the hardware zoo
  - Operating system manages resources
  - Fast enough for spinning disks



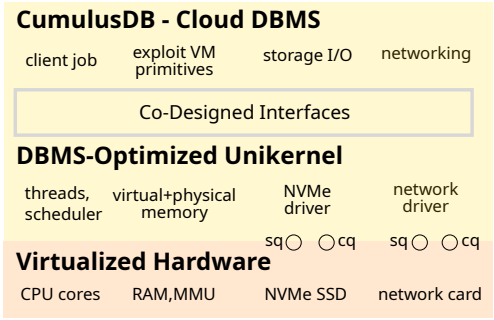
The Katabasis: From Traditional DBMS to CumulusDB

- Traditional Database Management Systems
  - Standard POSIX API tames the hardware zoo
  - Operating system manages resources
  - Fast enough for spinning disks
- Modern (High-Performance) DBMSes
  - I/O and Network became too fast for OS
  - User-Space: task queues, disk cache, memory
  - Kernel bypass required to exploit SSDs



The Katabasis: From Traditional DBMS to CumulusDB

- Traditional Database Management Systems
  - Standard POSIX API tames the hardware zoo
  - Operating system manages resources
  - Fast enough for spinning disks
- Modern (High-Performance) DBMSes
  - I/O and Network became too fast for OS
  - User-Space: task queues, disk cache, memory
  - Kernel bypass required to exploit SSDs



- **CumulusDB** – A Kernel-Integrated DBMS for the Cloud
  - Unikernel Principle: Melt OS and DBMS together, hypervisor brings isolation
  - Target Platform: Virtualized hardware is a uniform and stable ABI (x86, NVMe, virtio)
  - Kernel Integration: Vertically-integrated management of all resources

Benefits of Kernel Integration

- Kernel integration gives the DBMS access to privileged operations/interfaces

Benefits of Kernel Integration

- Kernel integration gives the DBMS access to privileged operations/interfaces

  Computation & Scheduling

  - Manipulate IRQ-vector tables
  - Block IRQs, Send IPIs, Shutdown CPUs
  - **Goal**: Query-plan-aware task/thread scheduling

Benefits of Kernel Integration

- Kernel integration gives the DBMS access to privileged operations/interfaces

Computation & Scheduling

- Manipulate IRQ-vector tables
- Block IRQs, Send IPIs, Shutdown CPUs
- **Goal**: Query-plan-aware task/thread scheduling

Memory Management

- Concurrent Lock-Free Page-Table Access
- Partially inconsistent TLB-states
- **Goal**: Virtual-memory as an active abstraction

Benefits of Kernel Integration

- Kernel integration gives the DBMS access to privileged operations/interfaces

Computation & Scheduling

- Manipulate IRQ-vector tables
- Block IRQs, Send IPIs, Shutdown CPUs
- **Goal**: Query-plan-aware task/thread scheduling

Memory Management

- Concurrent Lock-Free Page-Table Access
- Partially inconsistent TLB-states
- **Goal**: Virtual-memory as an active abstraction

Storage & Networking

- Direct access to NVMe queues
- Kernel-bypass on steroids
- **Goal**: Zero-copy from query to result stream

Benefits of Kernel Integration

- Kernel integration gives the DBMS access to privileged operations/interfaces

Computation & Scheduling

- Manipulate IRQ-vector tables
- Block IRQs, Send IPIs, Shutdown CPUs
- **Goal**: Query-plan-aware task/thread scheduling

Memory Management

- Concurrent Lock-Free Page-Table Access
- Partially inconsistent TLB-states
- **Goal**: Virtual-memory as an active abstraction

Storage & Networking

- Direct access to NVMe queues
- Kernel-bypass on steroids
- **Goal**: Zero-copy from query to result stream

Hypervisor Interaction

- Hypercalls for synchronous signals
- HV-inspected shared memory regions
- **Goal**: Elastic Resource Allocation for VMs

Benefits of Kernel Integration

- Kernel integration gives the DBMS access to privileged operations/interfaces

Computation & Scheduling

- Manipulate IRQ-vector tables
- Block IRQs, Send IPIs, Shutdown CPUs
- **Goal**: Query-plan-aware task/thread scheduling

Memory Management

- Concurrent Lock-Free Page-Table Access
- Partially inconsistent TLB-states
- **Goal**: Virtual-memory as an active abstraction

Storage & Networking

- Direct access to NVMe queues
- Kernel-bypass on steroids
- **Goal**: Zero-copy from query to result stream

Hypervisor Interaction

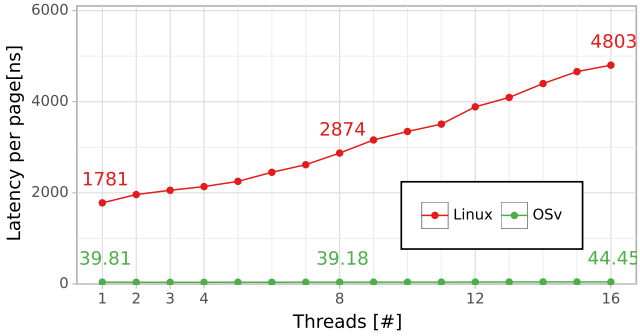
- Hypercalls for synchronous signals
- HV-inspected shared memory regions
- **Goal**: Elastic Resource Allocation for VMs

Virtual-Page–Presence Check

- Check if a virtual page is present
  - VM-based buffer managers[3]
  - OS treats VM as implementation detail
- Linux: `/proc/*/pagemap` interface
  - Accesses via `read(2)`
  - Architecture-independent format
- Unikernel: Shared MMU Structures
  - Lock-Free Page-Table Walk
  - Access to Physical Addresses[6]

Virtual-Page–Presence Check

- Check if a virtual page is present
  - VM-based buffer managers[3]
  - OS treats VM as implementation detail
- Linux: `/proc/*/pagemap` interface
  - Accesses via `read(2)`
  - Architecture-independent format
- Unikernel: Shared MMU Structures
  - Lock-Free Page-Table Walk
  - Access to Physical Addresses[6]



Host: AMD EPYC 9554P processor (64 cores, 128 HW threads, 384 GiB DRAM)
Virtual Machine: 16 cores, 12 GiB DRAM, QEMU, 4 GiB VMA
Workload: Random 4KiB Page

Use-Case Scenario

```
DB *G;   // Database

void olap(){ // 1 OLAP thread
  D = snapshot_create(global);
  res r = olap_scan(D);
  snapshot_destroy(D);
}

void oltp(){ // N OLTP threads
  while (1) {
    G[rand()]->modify();
  }
}
```

- Copy-on-Write Snapshots
  - Consistent of VM area
  - Analytics on read-only copy

Virtual-Memory-based Database Snapshots

Use-Case Scenario

```
DB *G;   // Database

void olap(){ // 1 OLAP thread
  D = snapshot_create(global);
  res r = olap_scan(D);
  snapshot_destroy(D);
}

void oltp(){ // N OLTP threads
  while (1) {
    G[rand()]->modify();
  }
}
```

- Copy-on-Write Snapshots
  - Consistent of VM area
  - Analytics on read-only copy

Linux

- Process-based snapshots [2]
  - `fork()` new process
  - Analytics in second process
  - Copy-on-Write
- During the copy
  - Single-threaded PT copy
  - OLTP threads have to block
- During the Analysis
  - TLB-shootdown storm
  - OLAP slows down OLTP

Virtual-Memory-based Database Snapshots

Use-Case Scenario

```
DB *G;    // Database

void olap(){ // 1 OLAP thread
  D = snapshot_create(global);
  res r = olap_scan(D);
  snapshot_destroy(D);
}

void oltp(){ // N OLTP threads
  while (1) {
    G[rand()]->modify();
  }
}
```

- Copy-on-Write Snapshots
  - Consistent of VM area
  - Analytics on read-only copy

Linux

- Process-based snapshots [2]
  - `fork()` new process
  - Analytics in second process
  - Copy-on-Write
- During the copy
  - Single-threaded PT copy
  - OLTP threads have to block
- During the Analysis
  - TLB-shootdown storm
  - OLAP slows down OLTP

CumulusDB

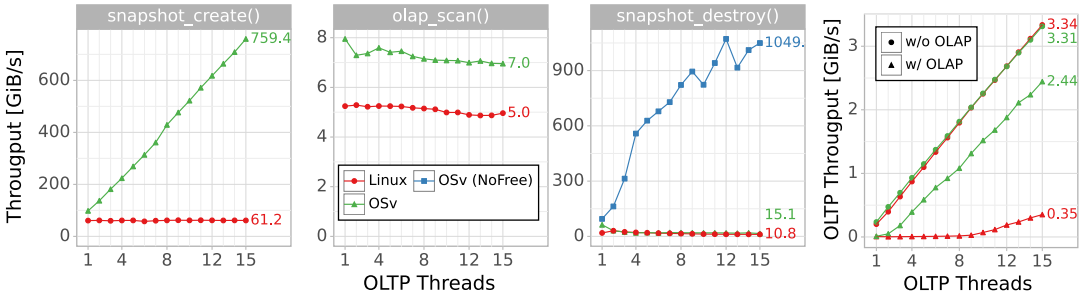
- Fine-Grained VM Snapshot [8]

```
void *snapshot_create
       (addr, length);
```

- **Ad-Hoc Parallelization**
  On CoW pagefault, the OLTP threads help an ongoing snapshot to copy page tables.
- **Reader-Side TLB invalidation**
  - No TLB Shootdown
  - CPU-local TLB-entry flush before every OLTP page access

Results: Linux vs. Modified OSv Unikernel

Host: AMD EPYC 9554P processor (64 cores, 128 HW threads, 384 GiB DRAM, 1 NUMA domain)
Virtual Machine: 16 cores, 12 GiB DRAM, QEMU, 4 GiB Snapshot Area



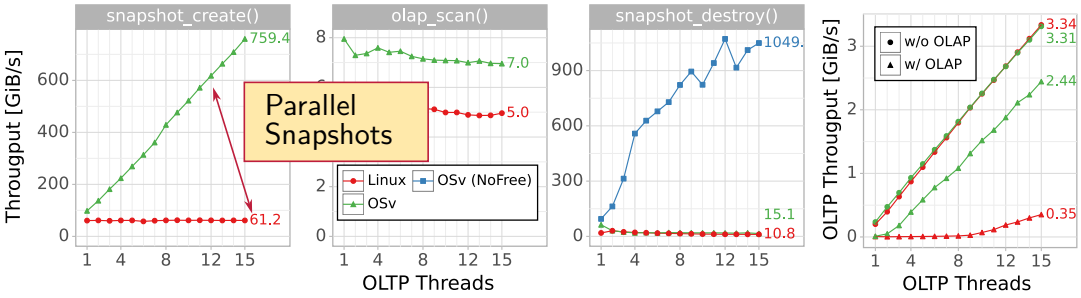
(a) Phases of an OLAP Job on an Copy-on-Write Snapshot

(b) Impact on OLTP Operations

Figure 2: Snapshot Copy-On-Write Benchmark

Results: Linux vs. Modified OSv Unikernel

Host: AMD EPYC 9554P processor (64 cores, 128 HW threads, 384 GiB DRAM, 1 NUMA domain)
Virtual Machine: 16 cores, 12 GiB DRAM, QEMU, 4 GiB Snapshot Area



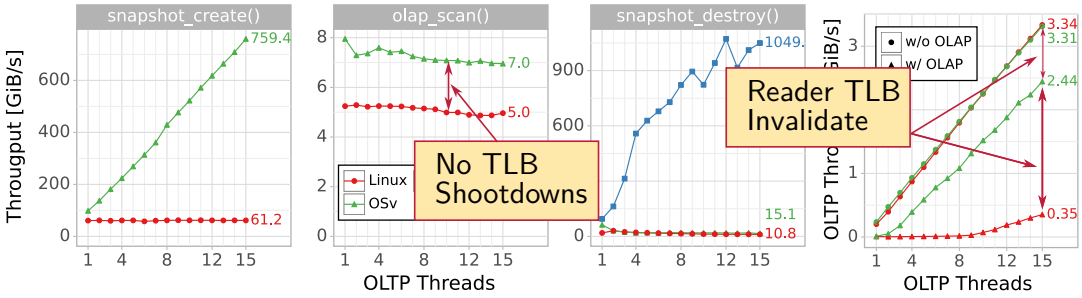
(a) Phases of an OLAP Job on an Copy-on-Write Snapshot

(b) Impact on OLTP Operations

Figure 2: Snapshot Copy-On-Write Benchmark

Results: Linux vs. Modified OSv Unikernel

Host: AMD EPYC 9554P processor (64 cores, 128 HW threads, 384 GiB DRAM, 1 NUMA domain)
Virtual Machine: 16 cores, 12 GiB DRAM, QEMU, 4 GiB Snapshot Area



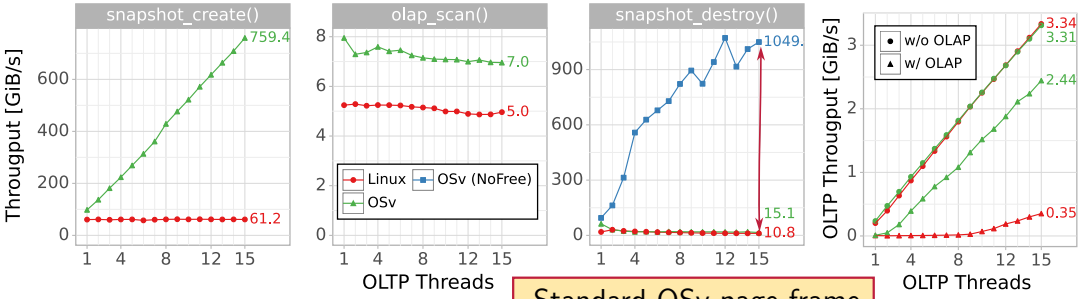
(a) Phases of an OLAP Job on an Copy-on-Write Snapshot

(b) Impact on OLTP Operations

Figure 2: Snapshot Copy-On-Write Benchmark

Results: Linux vs. Modified OSv Unikernel

Host: AMD EPYC 9554P processor (64 cores, 128 HW threads, 384 GiB DRAM, 1 NUMA domain)
Virtual Machine: 16 cores, 12 GiB DRAM, QEMU, 4 GiB Snapshot Area



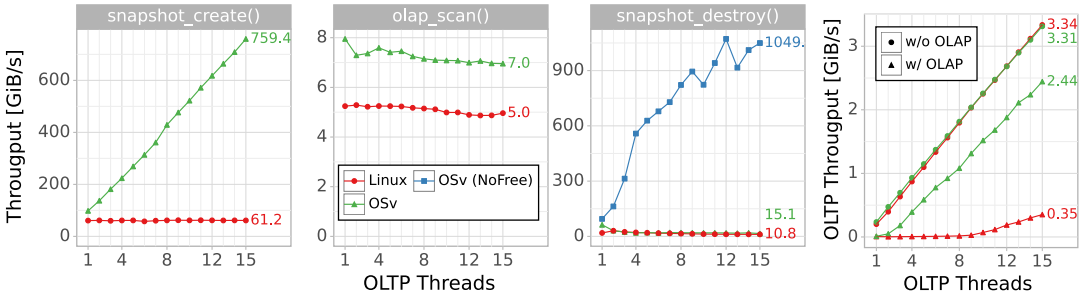
(a) Phases of an OLAP Job on an Copy-on-W

pact on OLTP Operations

Figure 2: Snapshot Copy-On-Write Benchmark

Results: Linux vs. Modified OSv Unikernel

Host: AMD EPYC 9554P processor (64 cores, 128 HW threads, 384 GiB DRAM, 1 NUMA domain)
Virtual Machine: 16 cores, 12 GiB DRAM, QEMU, 4 GiB Snapshot Area



(a) Phases of an OLAP Job on an Copy-on-Write Snapshot

(b) Impact on OLTP Operations

Figure 2: Snapshot Copy-On-Write Benchmark

More Details in our VLDB'24 Vision Paper[4]

Related DBMS–OS Co-Design Projects

- **DBOS** – A DBMS-oriented Operating System — Stanford, MIT, Google, VMware [5, 9]
  - OS components sit on top of distributed database
  - Each node runs a minimal microkernel (DBOS-brick) — still missing
  - DBOS is a cloud orchestrator. CumulusDB runs on a single node.

- **MxKernel** – Runtime System for Heterogeneous Many-Core Systems — Osnabrück, Dortmund [7]
  - Run-to-completion tasks, Dynamic system partitions (habitats, cells)
  - Kernel–application boundary, Isolation domains, RPC between components
  - MxKernel aims for heterogeneous systems, CumulusDB targets the unified cloud environment

- **COD** Open up the OS for the DBMS — Giceva et.al [1]
  - Problem: DBMS lacks knowledge that the OS already has.
  - Resource-allocation protocol between OS and DBMS
  - COD transports information, CumulusDB forwards hardware access

Summary

- The cloud attracts abstractions layers like a lamp on midsummer night
  - Good for the average application and its developer (not for his purse)
  - Database management systems are not the average application!

- **CumulusDB**: A Cloud-Native DBMS based on Kernel-Integration
  - Combine a Unikernel and a DBMS into a single VM image without isolation
  - Vertical-integrated resource management and privileged hardware access
  - The virtualized hardware is a portable machine model

- New Primitives for the DBMS
  - Page-Table Walk — Directly access MMU data for in-core check
  - Ad-hoc parallelization: — Instead of blocking, others help completing the snapshot
  - Joint TLB management: — Distribute the chores of TLB management between reader and writer

Advisement Block

We're hiring!
If you're interested in doing an PhD on CumulusDB, get in touch!

Institute of Operating Systems and Computer Networks
Reliable Systemsoftware

References I

[1] Jana Giceva, Tudor-Ioan Salomie, Adrian Schüpbach, et al. “COD: Database / Operating System Co-Design”. In: CIDR. 2013.

[2] Alfons Kemper and Thomas Neumann. “HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots”. In: ICDE. 2011. DOI: 10.1109/ICDE.2011.5767867.

[3] Viktor Leis, Adnan Alhomssi, Tobias Ziegler, et al. “Virtual-Memory Assisted Buffer Management”. In: Proceedings of the ACM SIGMOD/PODS International Conference on Management of Data. Seattle, WA, USA: ACM, June 2023. DOI: 10.1145/3588687.

[4] Viktor Leis and Christian Dietrich. “Cloud-Native Database Systems and Unikernels: Reimagining OS Abstractions for Modern Hardware [Vision]”. In: Proceedings of the 50th International Conference on Very Large Data Bases. Vision Paper, Accepted with availability check. Guangzhou, China: VLDB Endowment, Aug. 2024.

[5] Qian Li, Peter Kraft, Kostis Kaffes, et al. “A Progress Report on DBOS: A Database-oriented Operating System”. In: CIDR. 2022.

[6] Yannick Loeck and Christian Dietrich. “Evaluation and Refinement of an Explicit Virtual-Memory Primitive”. In: IEEE Access 11 (Dec. 2023), pp. 136855–136868. DOI: 10.1109/ACCESS.2023.3338149.

References II

[7] Jan Mühlig, Michael Müller, Olaf Spinczyk, et al. “mxkernel: A Novel System Software Stack for Data Processing on Modern Hardware”. In: Datenbank-Spektrum 20.3 (2020). DOI: 10.1007/s13222-020-00357-5.

[8] Ankur Sharma, Felix Martin Schuhknecht, and Jens Dittrich. “Accelerating Analytical Processing in MVCC using Fine-Granular High-Frequency Virtual Snapshotting”. In: SIGMOD. 2018. DOI: 10.1145/3183713.3196904.

[9] Athinagoras Skiadopoulos, Qian Li, Peter Kraft, et al. “DBOS: A DBMS-oriented Operating System”. In: PVLDB 15.1 (2021). DOI: 10.14778/3485450.3485454.