Databases in the Context of HPC with focus on scalability

Andreas Kergel, Daniel Reither, Manuel Taiber, Martin Till und Sophie Loibl

Introduction

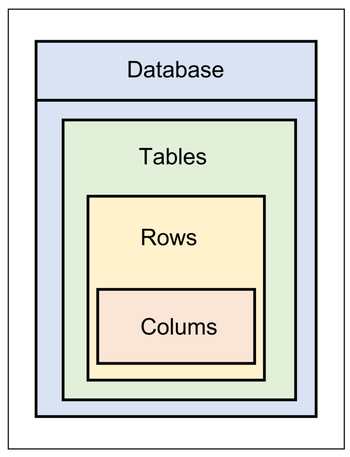
High-Performance Computing (HPC) systems are designed to process extremely large datasets and computationally intensive tasks using powerful processors and distributed clusters. In such environments, databases are not merely storage components but performance-critical infrastructure elements that must provide fast and reliable access during simulations, analytics, and parallel computations. As data volumes and workload complexity increase, the limiting factor is often no longer raw compute power but the efficiency and scalability of the data infrastructure.

Scalability describes the ability of a database system to maintain stable performance despite growing data size, user concurrency, or computational demand. Database systems implement scalability through vertical scaling (upgrading a single machine), horizontal scaling (distributing data across multiple nodes), and parallel processing (simultaneous execution of operations). Each approach introduces architectural trade-offs in terms of performance, complexity, and reliability.

Topic Overview

Database	Short Overview	Type	Scalability	HPC Suitability
MariaDB	Works well on clusters, supports plugins like Galera and ProxySQL for load balancing and replication.	Relational	Very good	Good
PostgresSQL	Supports parallel queries, table partitioning, and extensions like Citus for distributed scaling.	Relational	Very Good	Good
Oracle	Supports in-memory column store acceleration, Real Application Clusters for distributed scaling, and hybrid row/column storage for mixed workloads.	Relational (In-Memory enabled)	Very Good	Good
MongoDB	Provides sharding across nodes, distributed storage, and a flexible schema design.	NoSQL	Good	Limited
NoSQL	Fast key-value, document, or column stores optimized for distributed and high-throughput workloads.	NoSQL	Varies	Depends

Structure



PostgresSQL

Structured

Schema-based

ACID-compliant

Transactional

Table-oriented

MongoDB

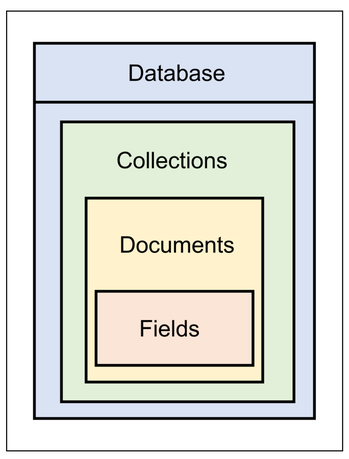
Flexible

Document-oriented

Schema-less

Decentralized

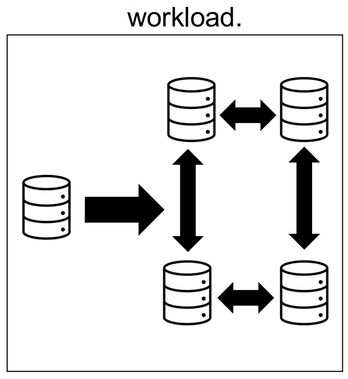
Nested



Scalability

Horizontal Scaling

Means adding more machines to share the workload.
In HPC, distributed computing spreads computations across nodes; in databases, horizontal scaling distributes both data and workload.

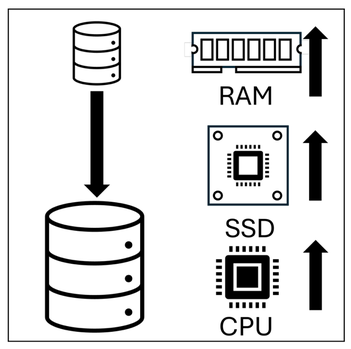


Advantages
- Highly expandable (can grow almost indefinitely)
- Fault tolerant (one node can fail)
- Ideal for large datasets and distributed systems
- Works well in cloud and clusters

Disadvantages
- Possible data consistency issues
- Higher administration and maintenance effort
- Complex architecture (networking, synchronization)

Vertical Scaling

Means adding more resources (CPU, RAM, storage) to a single machine.
In HPC context: Boosting a single node's capacity to handle larger computations without adding more nodes.

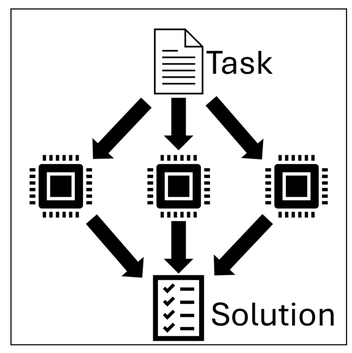


Advantages
- Easy to implement
- No data distribution required
- Strong consistency
- Often better single-query performance

Disadvantages
- Single point of failure
- Eventually cannot scale further
- Downtime during hardware upgrades

Parallelism

Means executing multiple tasks simultaneously, on oen or multible nodes.
In HPC context: Reduces computation time by splitting workloads into parallel tasks, independent of the number of machines.



Advantages
- Significantly reduces computation time
- Core principle of HPC
- Utilizes modern multi-core CPUs efficiently
- Can be combined with horizontal and vertical scaling

Disadvantages
- Difficult to program (race conditions, deadlocks)
- Not every task can be parallelized
- Debugging becomes harder

Comparison

PostgreSQL

- Primarily designed for **vertical scaling** – adding CPU/RAM to a single powerful node.
- Supports **parallel queries** on multi-core nodes.
- Horizontal scaling possible via extensions like **Citus** or table partitioning.
- Used after computation to store processed results, run analyses, and support visualization or evaluation.
- Advanced SQL support for complex joins, aggregations, and analytical queries.

➡ Useful for maintaining data integrity and consistency in structured scientific workflows.

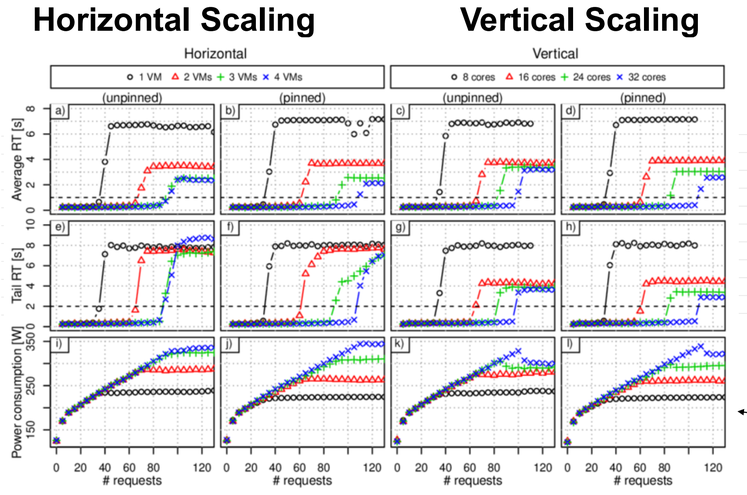
MongoDB

- Built for **horizontal scaling** – sharding across multiple nodes
- Handles **high write/read throughput** and many parallel accesses well
- Well-suited for distributed metadata, logging, and flexible datasets in HPC environments.
- Used before or during computation to collect raw data, parameters, and intermediate outputs across distributed systems.

➡ Useful for storing large, distributed, and flexible datasets (especially metadata / monitoring data) generated by HPC systems

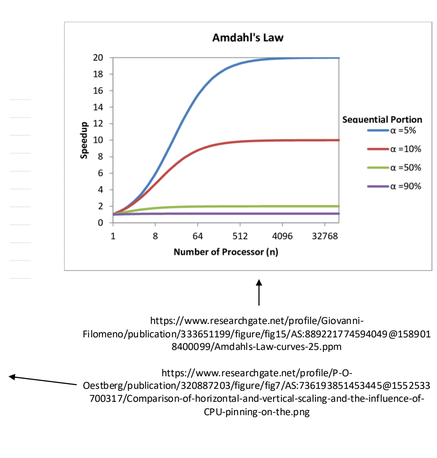
Scalability Limits

Horizontal Scaling



- Distributed coordination overhead
- Network latency between nodes
- CAP theorem trade-off

Vertical Scaling



- Hardware cost grows rapidly
- Memory bandwidth bottleneck
- Physical hardware limits

Parallelism

https://www.researchgate.net/profile/Giovanni-Filomeno/publication/333651199/figure/fig15/AS:889221774594049@1589018400099/Amdahls-Law-curves-25.ppm

https://www.researchgate.net/profile/P-O-Oestberg/publication/320887203/figure/fig7/AS:736193851453445@1552533700317/Comparison-of-horizontal-and-vertical-scaling-and-the-influence-of-CPU-pinning-on-the.png

- Thread synchronization overhead
- Memory & communication limits
- Limited by Amdahl's Law

Conclusion

KEY TAKEAWAYS:
- No single database architecture scales optimally for all HPC workloads
- **PostgreSQL-based systems** benefit from strong consistency and efficient vertical scaling on powerful nodes
- **MongoDB-like distributed systems** scale horizontally and handle large distributed datasets more naturally

However, scalability is limited by fundamental factors:
- Vertical scaling → hardware cost & memory bandwidth
- Horizontal scaling → network latency & synchronization overhead
- Parallelism → Amdahl's Law

Conclusion

Scalability in HPC environments is not a purely technical feature, but a strategic design decision that shapes the long-term efficiency and sustainability of a system. Different architectural approaches emphasize different strengths, and each comes with inherent trade-offs that cannot be fully eliminated. Rather than searching for a universally optimal database solution, system designers must determine which compromises are acceptable within their specific computational and organizational context. Ultimately, the central challenge lies not in selecting a database technology itself, but in aligning the database architecture with the specific workload characteristics and performance demands of the system.