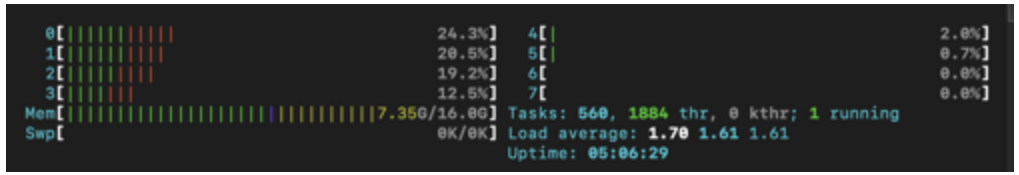
Parallele und verteilte Systeme

Einführung

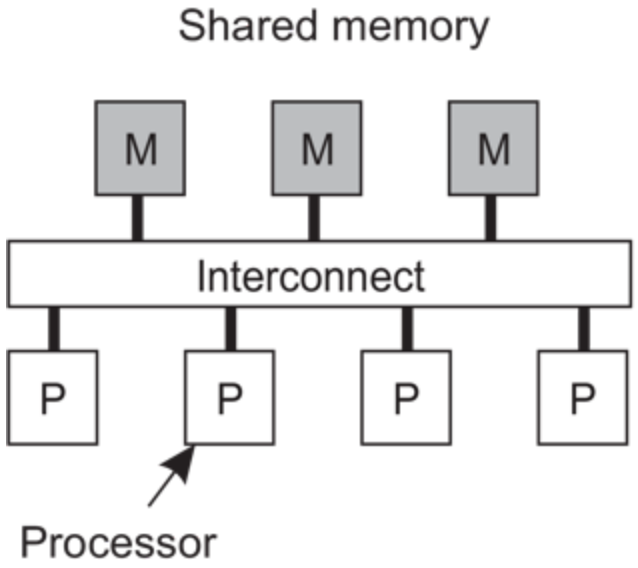
«Parallele Systeme»

- Eine single core CPU kann nur einen Prozess gleichzeitig ausführen
- Multi-core CPUs entsprechend mehrere gleichzeitig
- Ausser in sehr einfachen Embedded Systemen müssen jedoch immer sehr viele Prozesse «gleichzeitig» ausgeführt werden können z.B. auf einem Server oder auf einem Desktop Computer

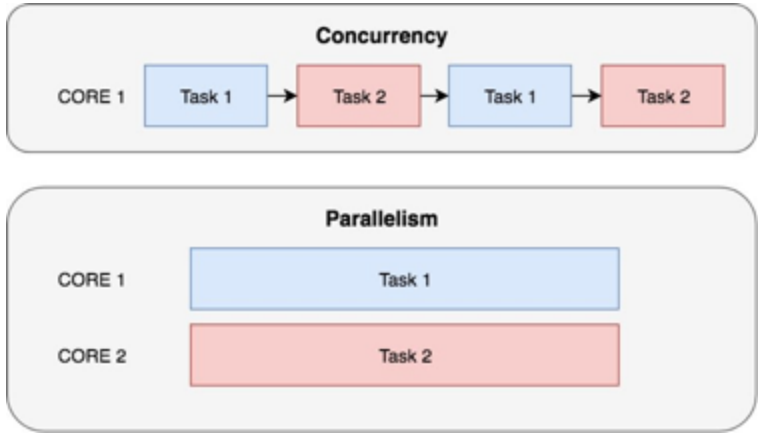


«Parallele Systeme»-

- Viele verschiedene Prozesse (tausende) werden von einem oder mehreren (bis zu dutzenden) Prozessoren ausgeführt
- Ein einzelner Prozessor kann demnach nacheinander mehrere Prozesse bearbeiten
- Die Prozessoren befinden sich auf demselben Chip oder auf dem selben Mainboard
- Sie haben geteilten sowie gemeinsamen Speicher
- Die Verbindung zwischen ihnen (Interconnect) hat geringe Latenz, hohe Bandbreite und ist zuverlässig.



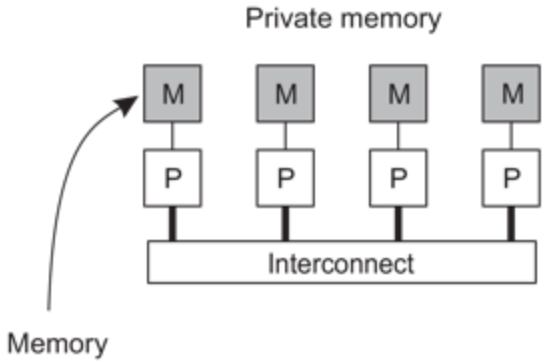
- Parallele Ausführung (parallelism): Mehr als eine Aufgabe wird gleichzeitig ausgeführt
- Nebenläufig (concurrency): Mehr als eine Aufgabe wird abgearbeitet (durch schnelles context switching)



- Eine zentrale Aufgabe von Betriebsystemen ist es, die Prozesse auf die CPUs zu verteilen.
- Dies wird «Scheduling» genannt.

Verteilte Systeme

> «A distributed system is a collection of independent computers that appears to its users as a single coherent system.»



VanSteen, 2017, S. 26

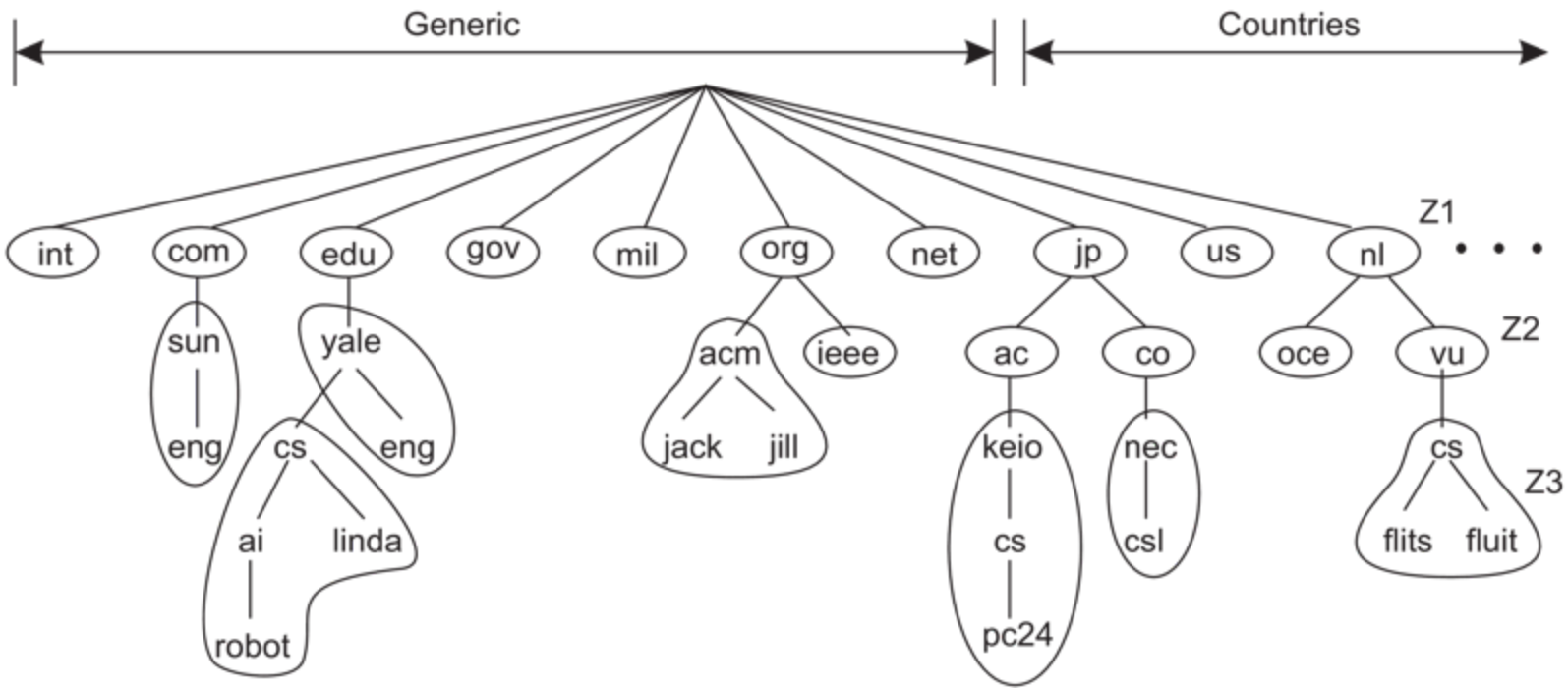
P: Prozessor,

Interconnect: Netzwerkverbindung, meistens HTTP, UDP/TCP, IP, Ethernet basiert

Resource Sharing

- Ressourcen verfügbar machen: Drucker, Computing, Storage, Daten, Netzwerk
- Teure Ressourcen können besser ausgelastet werden und müssen nicht mehrfach angeschafft werden
- Zusammenarbeit

Domain Name System



Anforderungen an moderne Software

- Hohe Verfügbarkeit
- Skalierbarkeit
- Im Katastrophenfall sollen die Systeme schnell wiederhergestellt werden können
- Soll funktionieren, auch wenn Teile des Systems Offline sind (Resilienz)
- Kostengünstig
- Einfach
- Updates müssen einfach eingespielt werden können

Skalierung

Vertikal

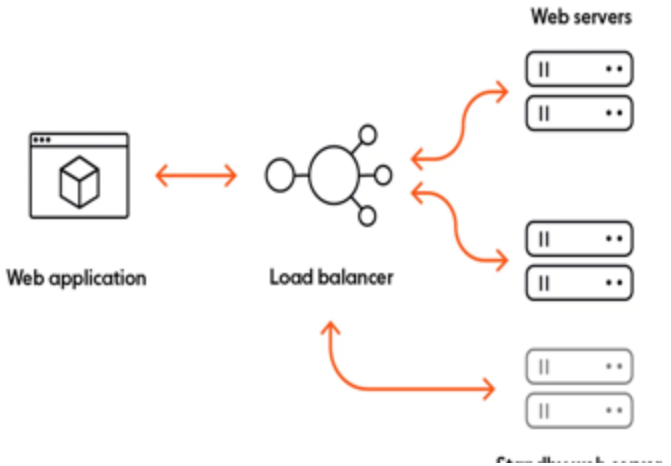
- Existierende Infrastruktur wird aufgerüstet
- Einfach
- Limitiert

Horizontal

- Hinzufügen von neuer Infrastruktur
- Aufwendig: Anwendungen müssen verteilt werden
- Kaum limitiert

Lösungsansätze

- Verfügbarkeit, Skalierbarkeit: Mehre identische Systeme müssen verfügbar sein.
- Bei Bedarf sollen weitere schnell gestartet werden können



Decentralized vs Distributed

Decentralized

- Matrix
- Mastodon
- Nextcloud
- ...

Distributed

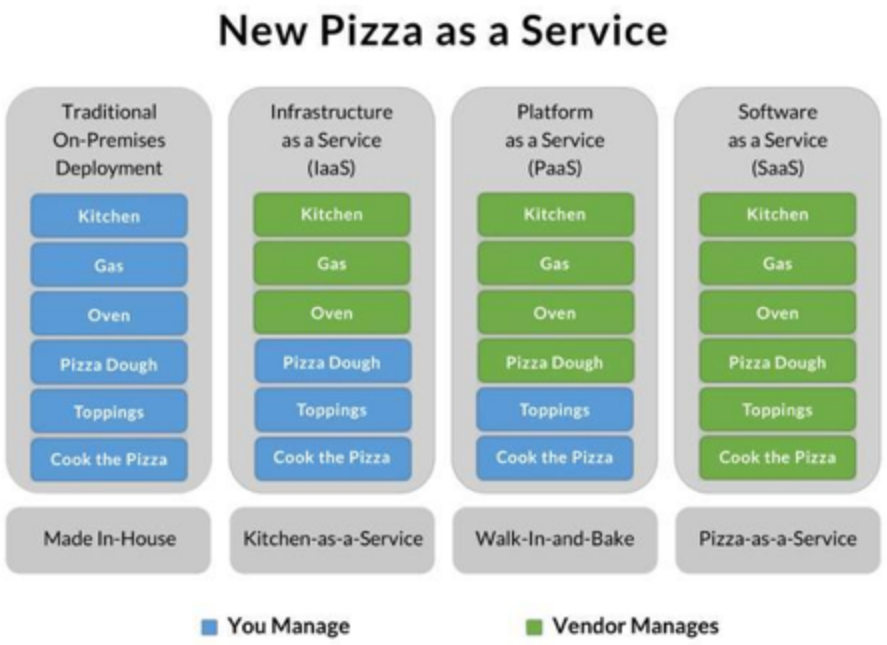
- CockroachDB
- Neon
- Ably
- ...

Cloud Computing

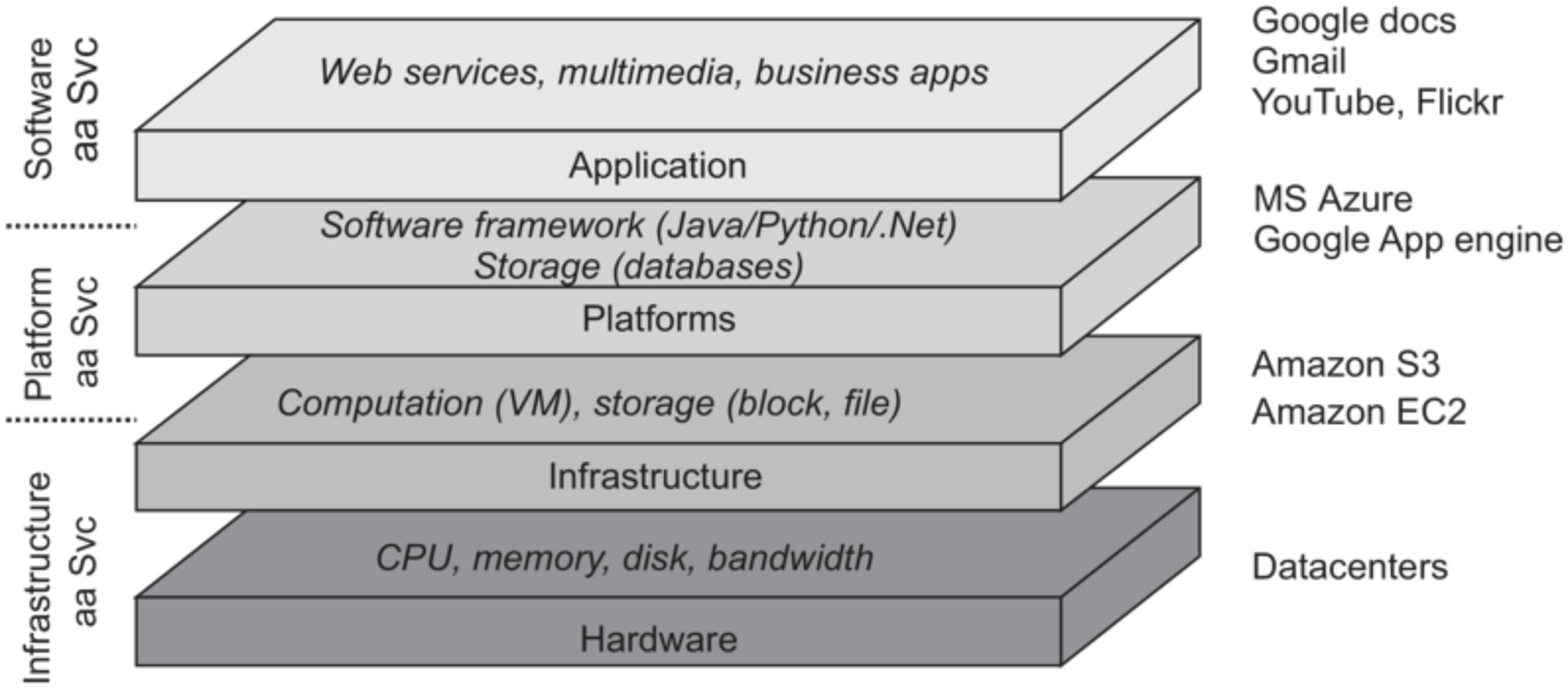
> The entire history of software engineering is that of the rise in levels of abstraction.

-- Grady Booch

Pizza as a Service

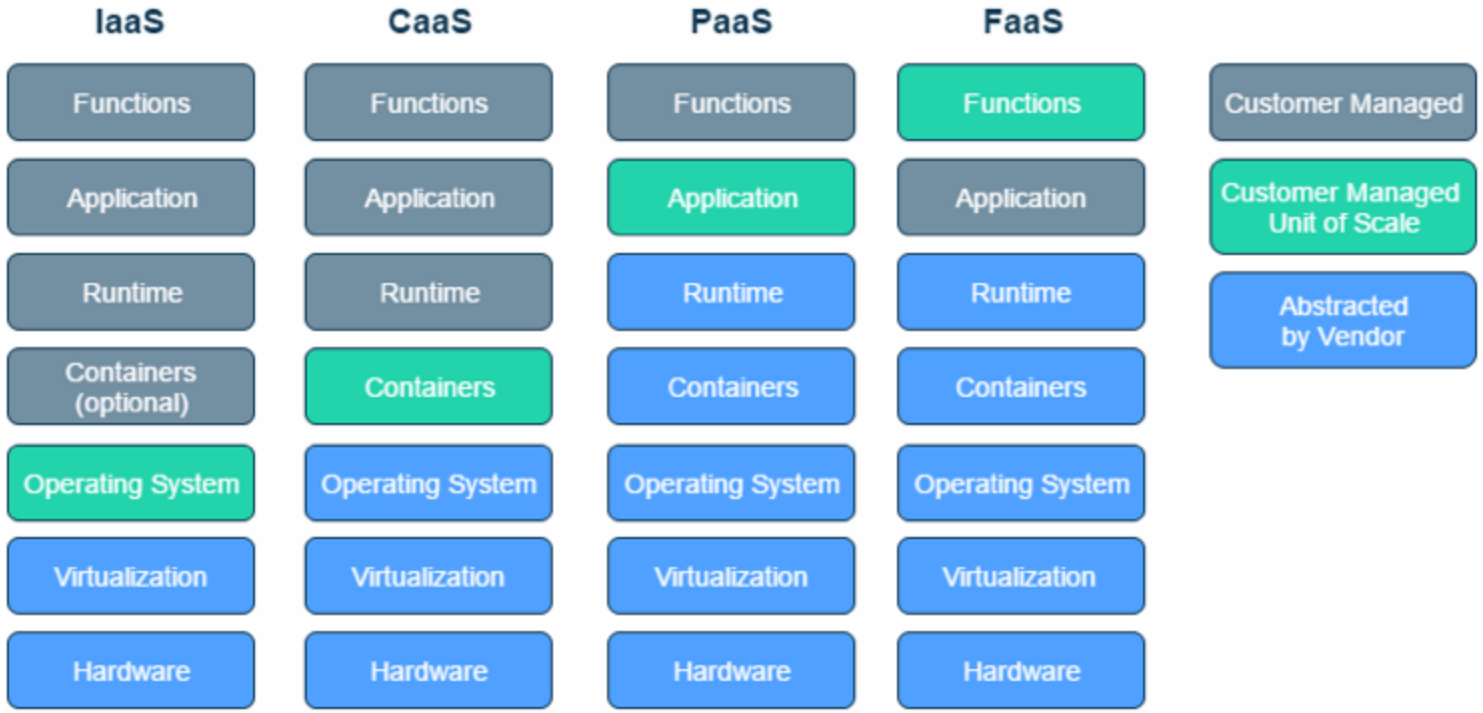


Abstractions

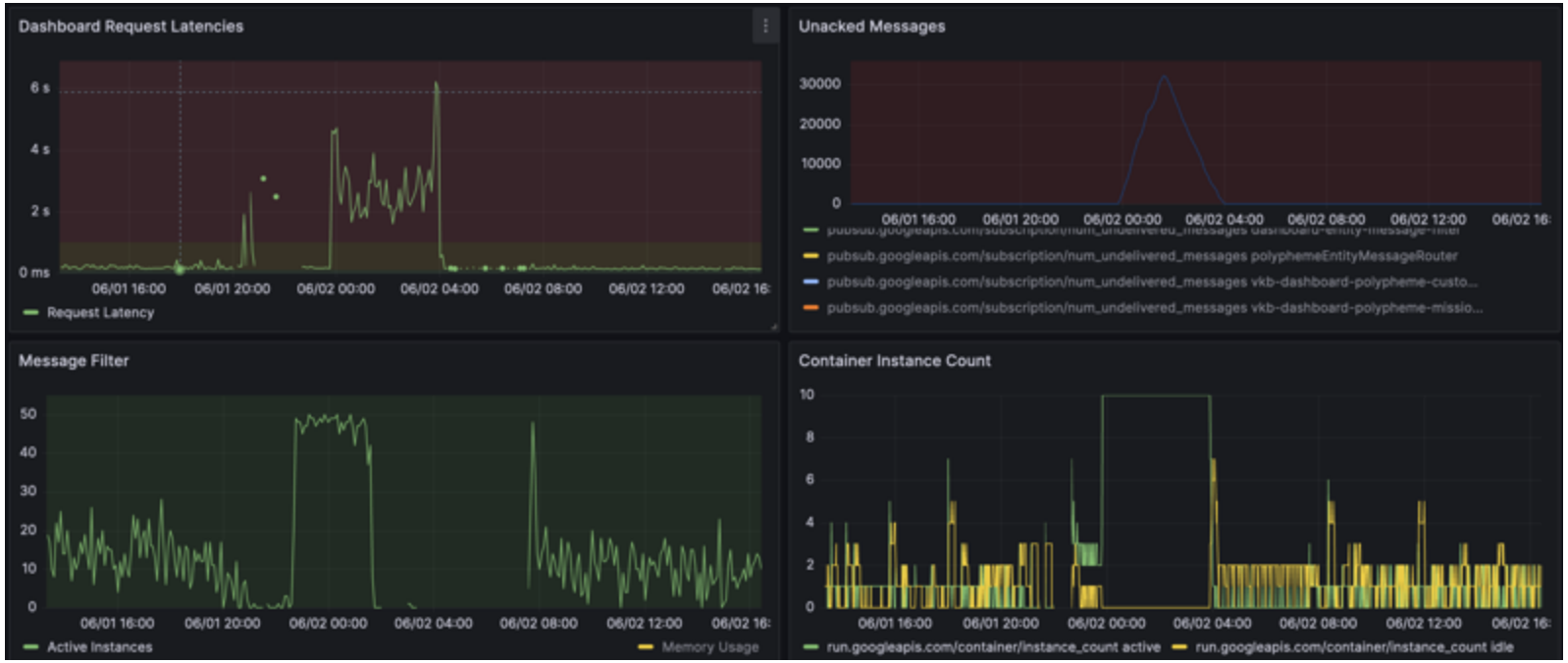


(VanSteen, 2017, S. 30)

XaaS


IaaS
Functions
Application
Runtime
Containers (optional)
Operating System
Virtualization
Hardware
CaaS
Functions
Application
Runtime
Containers
Operating System
Virtualization
Hardware
PaaS
Functions
Application
Runtime
Containers
Operating System
Virtualization
Hardware
FaaS
Functions
Application
Runtime
Containers
Operating System
Virtualization
Hardware
Customer Managed
Customer Managed Unit of Scale
Abstracted by Vendor

Fallstudie


Dashboard Request Latencies
6 s
4 s
2 s
0 ms
06/01 16:00
06/01 20:00
06/02 00:00
06/02 04:00
06/02 08:00
06/02 12:00
Request Latency
Unacked Messages
30000
20000
10000
0
pubsub.googleapis.com/subscription/num_undelivered_messages polypheme EntityMessageRouter
pubsub.googleapis.com/subscription/num_undelivered_messages vkb-dashboard-polypheme-custo...
pubsub.googleapis.com/subscription/num_undelivered_messages vkb-dashboard-polypheme-missio...
Message Filter
50
40
30
20
10
0
Active Instances
Memory Usage
Container Instance Count
10
8
6
4
2
0
run.googleapis.com/container/instance_count active
run.googleapis.com/container/instance_count idle

Fallstudie Design Goals: Cockroach DB

Why use CockroachDB?

There are many reasons to use CockroachDB, including:

- Resiliency
- Scalability
- Strong consistency
- Geo-partioning and multi-region features
- PostgreSQL-compatibility

Diskussion

Resource sharing

Ein Datenbanksystem kann als solches als gemeinsam genutzte Resource angesehen werden.
Da es meistens nicht sinnvoll ist, für jede Applikation erneut eine Persistenzschicht zu implementieren werden Daten, die persistiert werden müssen in einer Datenbank gespeichert.
Wenn sich diese Datenbank auf einem entfernten System befindet, handelt es sich um ein verteiltes System.

CockroachDB selber nutzt auch eine verteilte Architektur um verteilte Compute und Storage Ressourcen von Cloud Anbietern gewinnbringend zu nutzen.

Distribution Transparency

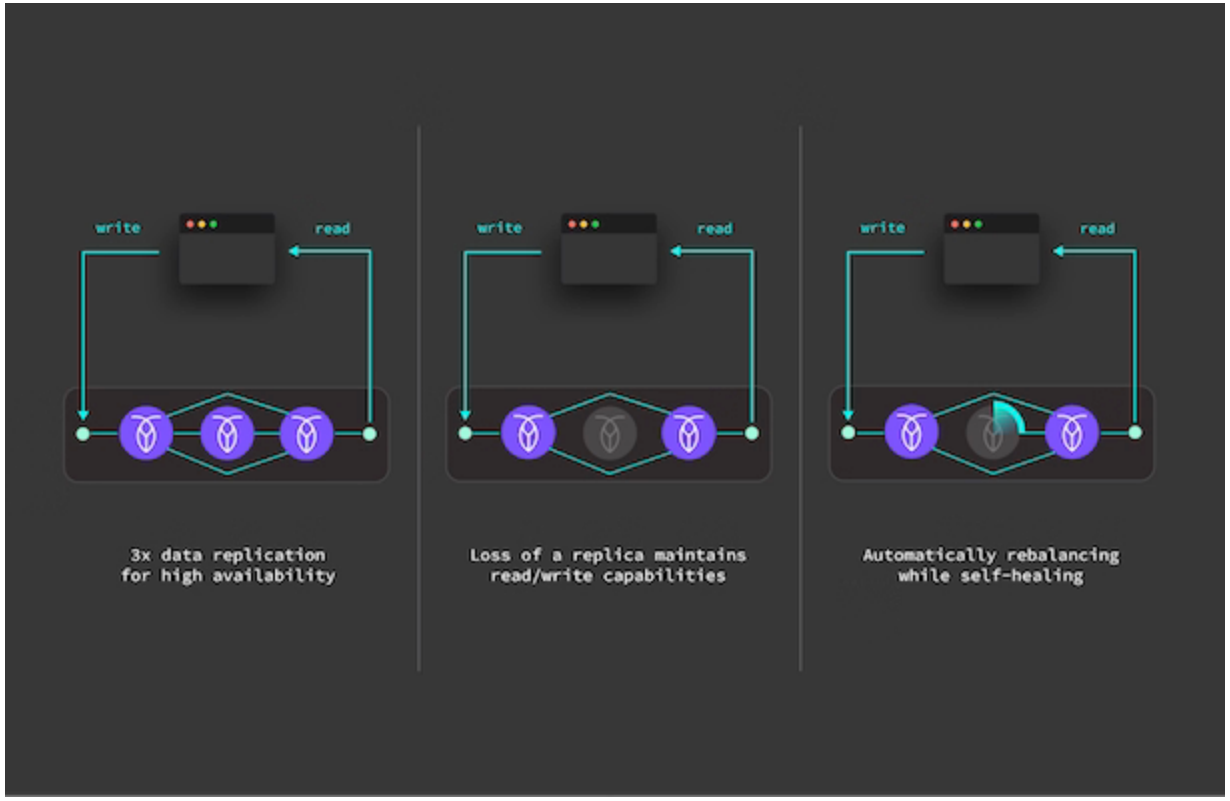
CockroachDB unternimmt einiges, um die verteilte Architektur zu verstecken.
Requests werden zum geographisch naheliegendsten Node geleitet, sowohl schreibend als auch lesend.
So bleiben die Latenzen tief, auch wenn die Nodes global verteilt sind.

Openness

CockroachDB ist zum PostgreSQL Dialekt kompatibel und ist so sehr offen.
Es können bestehende PostgreSQL Datenbanktreiber verwendet werden.
Und, im Rahmen des implementierten Sprachumfangs, kann die Datenbank mit einer anderen PostgreSQL kompatiblen Datenbank ausgetauscht werden.

Dependability

Zuverlässigkeit ist ein zentrales Feature von CockroachDB.
Die Daten werden mehrfach repliziert, bei einem Ausfall von einem Node sind die Daten weiterhin verfügbar.
Ist ein Node wieder online, werden die Daten automatisch wieder repliziert.



Security

CockroachDB unterstützt gängige security features.

Scaling

CockroachDB kann mit einer theoretisch beliebigen Anzahl an Nodes betrieben werden, auf welche die Last verteilt wird.
So kann in sehr grossem Ausmass skaliert werden.
Möglich ist dies, weil die einzelnen Nodes, anders als bei traditionelleren Datenbanksystemen, sowohl schreibend als auch lesend arbeiten können.