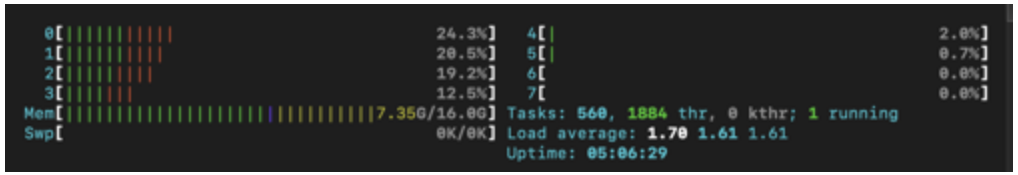
Einführung

«Parallele Systeme»

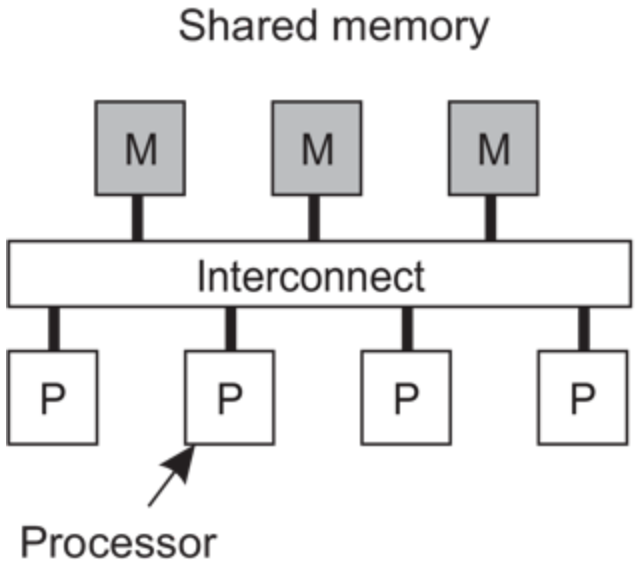
- Eine single core CPU kann nur einen Prozess gleichzeitig ausführen
- Multi-core CPUs entsprechend mehrere gleichzeitig
- Ausser in sehr einfachen Embedded Systemen müssen jedoch immer sehr viele Prozesse «gleichzeitig» ausgeführt werden

  können z.B. auf einem Server oder auf einem Desktop Computer

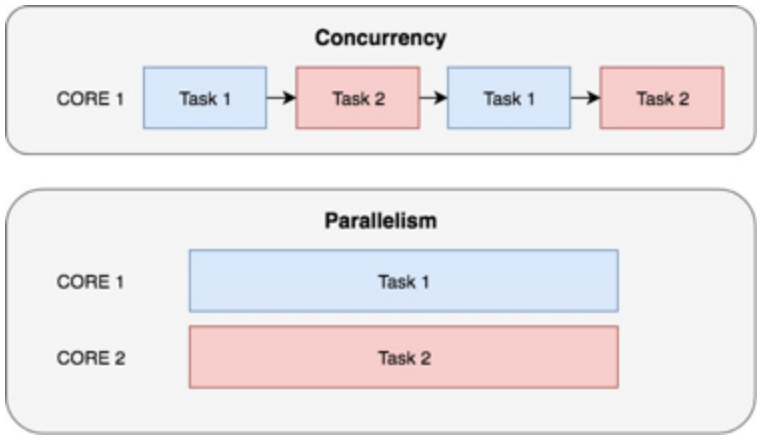


«Parallele Systeme»

- 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.



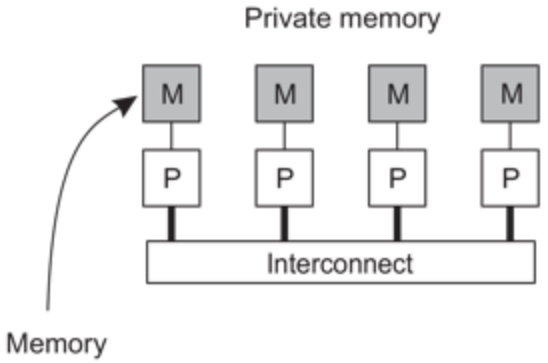
- Parallele Ausführung (parallelism): Mehr als eine Aufgabe wird gleichzeitig ausgeführt
- Nebenläufig (concurrency): Mehr als eine Aufgabe wird abgearbeitet (durch schnelles context switching)



- 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.»



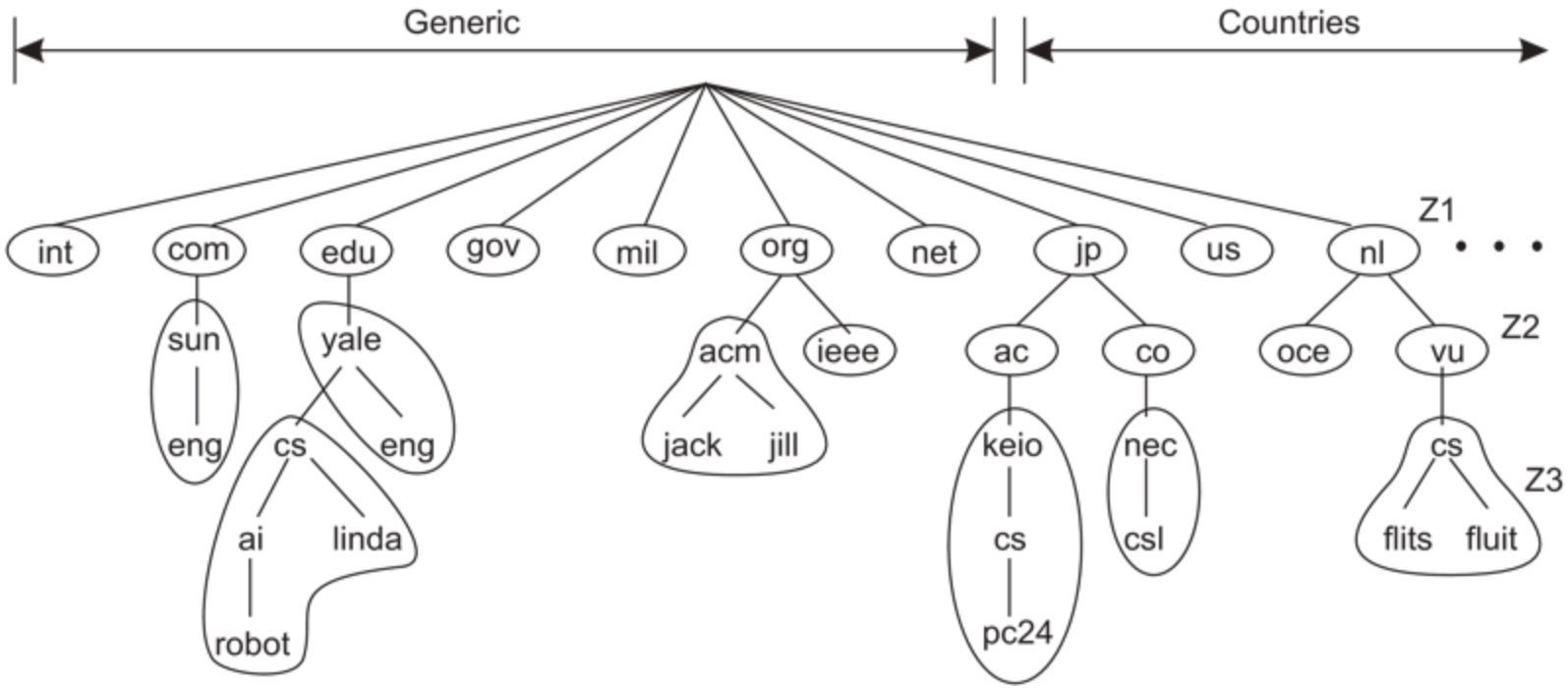
VanSteen, 2017, S. 26

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



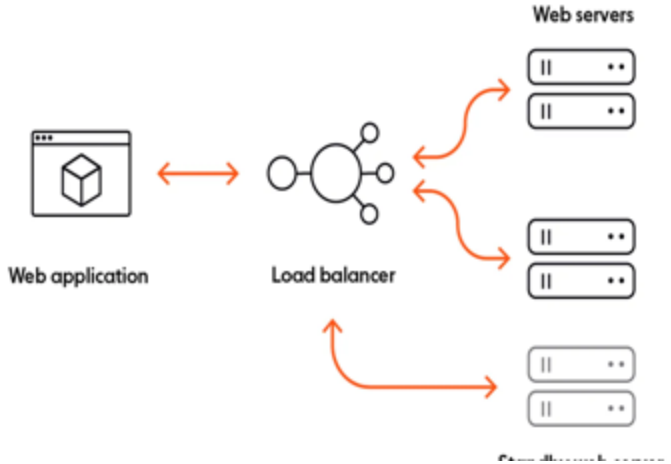

When the cloud leaves the datacenter
Meme-Generator.com

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

Lösungsansätze

- Verfügbarkeit, Skalierbarkeit: Mehre identische Systeme müssen verfügbar sein und bei Bedarf sollen weitere schnell
  gestartet werden können
- Tradeoff: Kostengünstig, Einfach



Decentralized vs Distributed

Decentralized

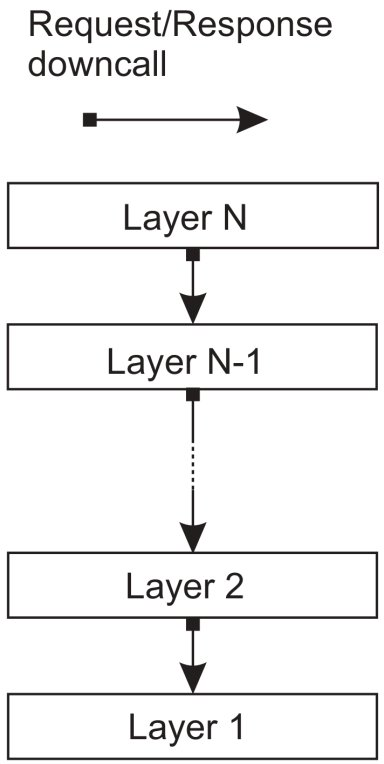
- Matrix
- Mastodon
- Nextcloud
- ...

Distributed

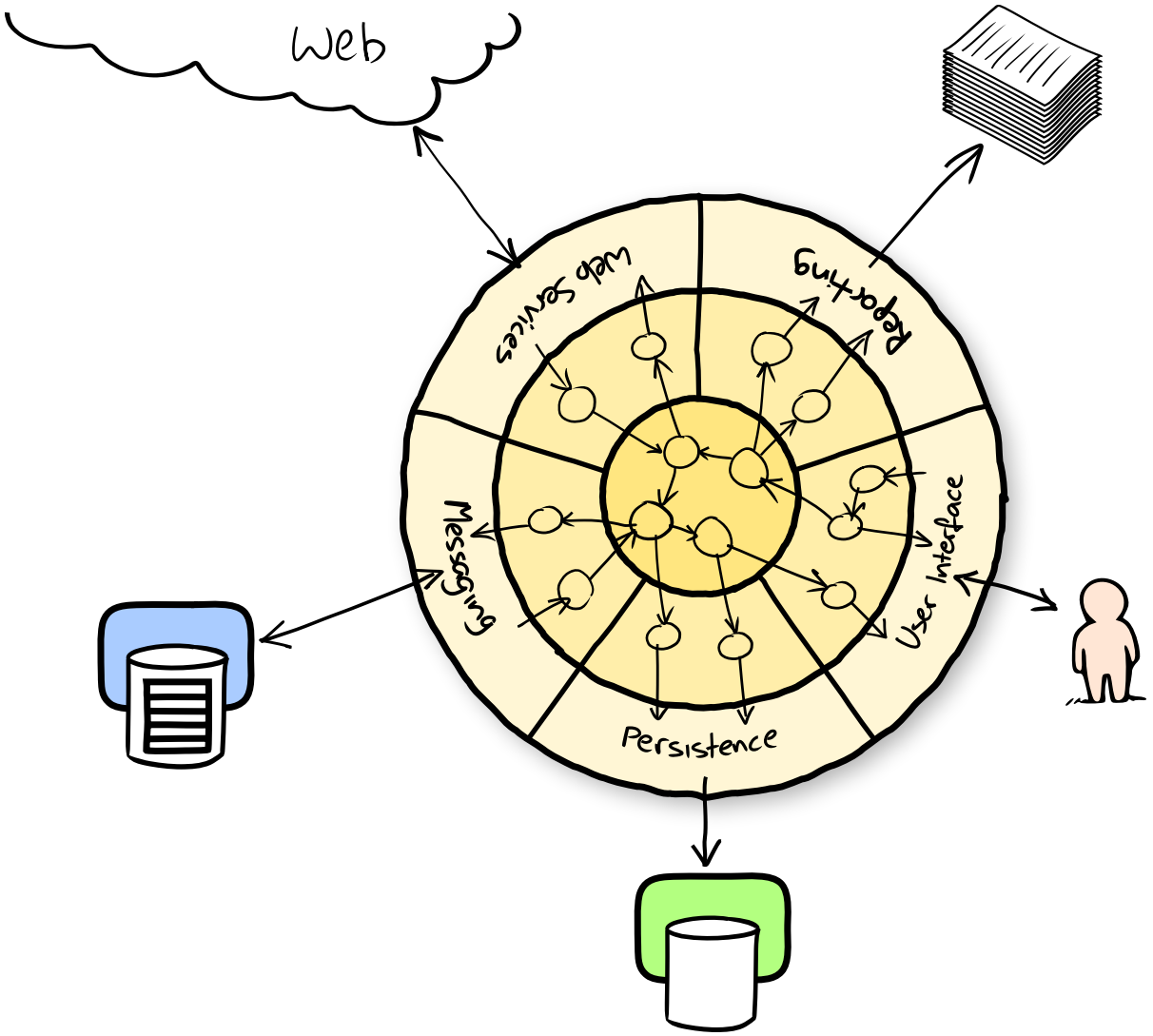
- CockroachDB
- Neon
- Ably
- ...

Architekturen

Schichtenarchitekturen

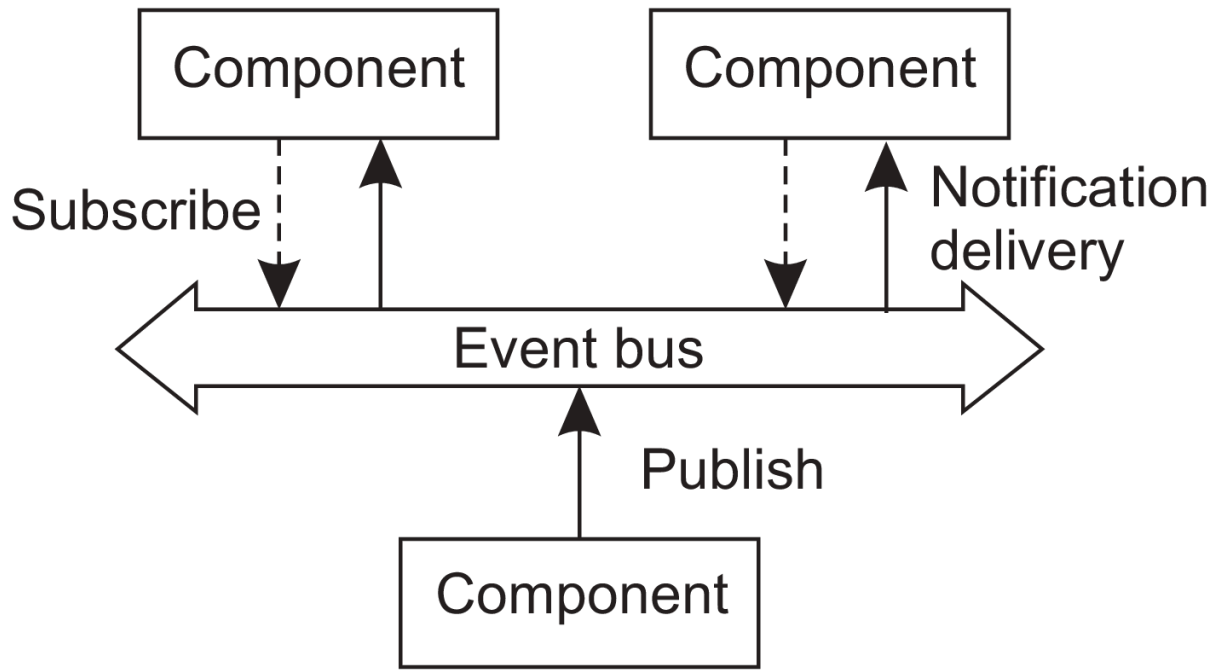


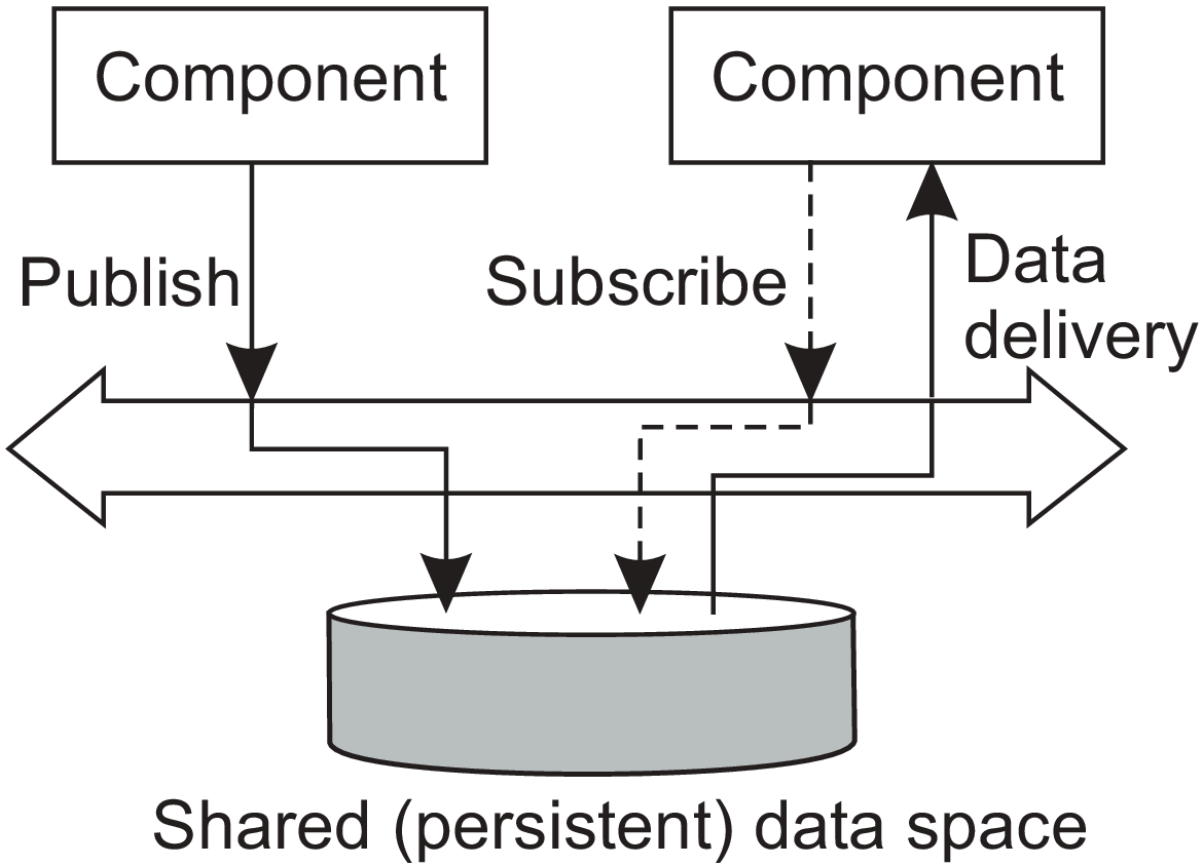
Ports and Adaptors architecture

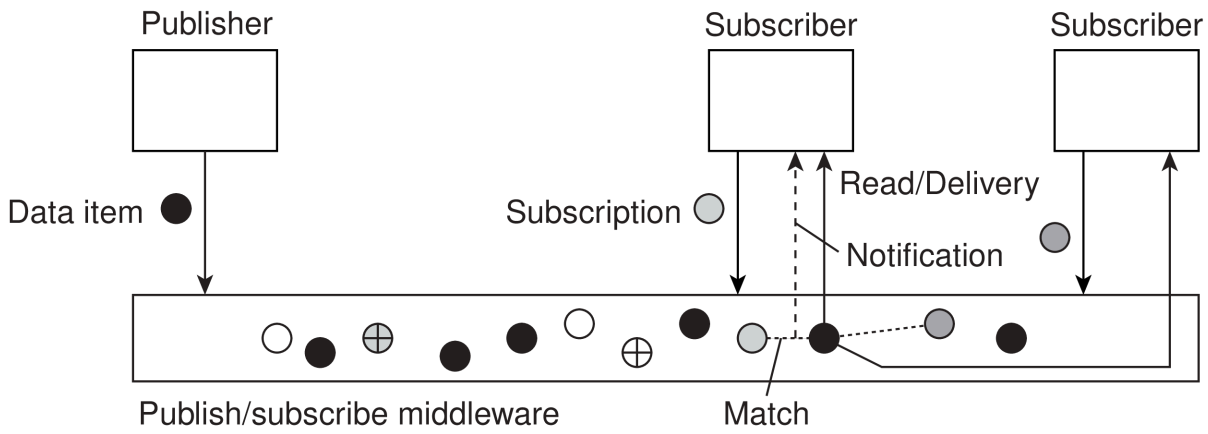


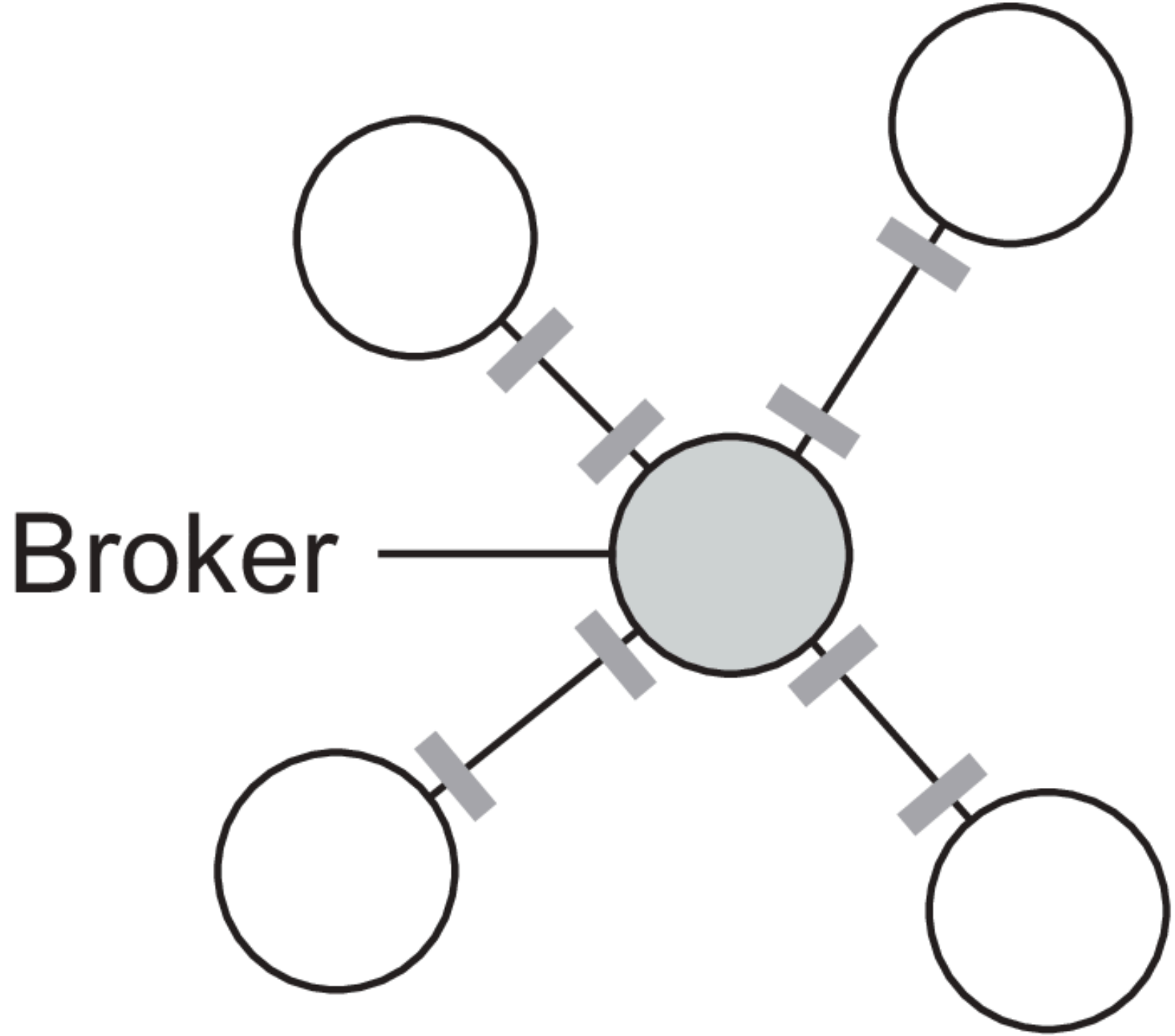
Publish-subscribe Architekturen

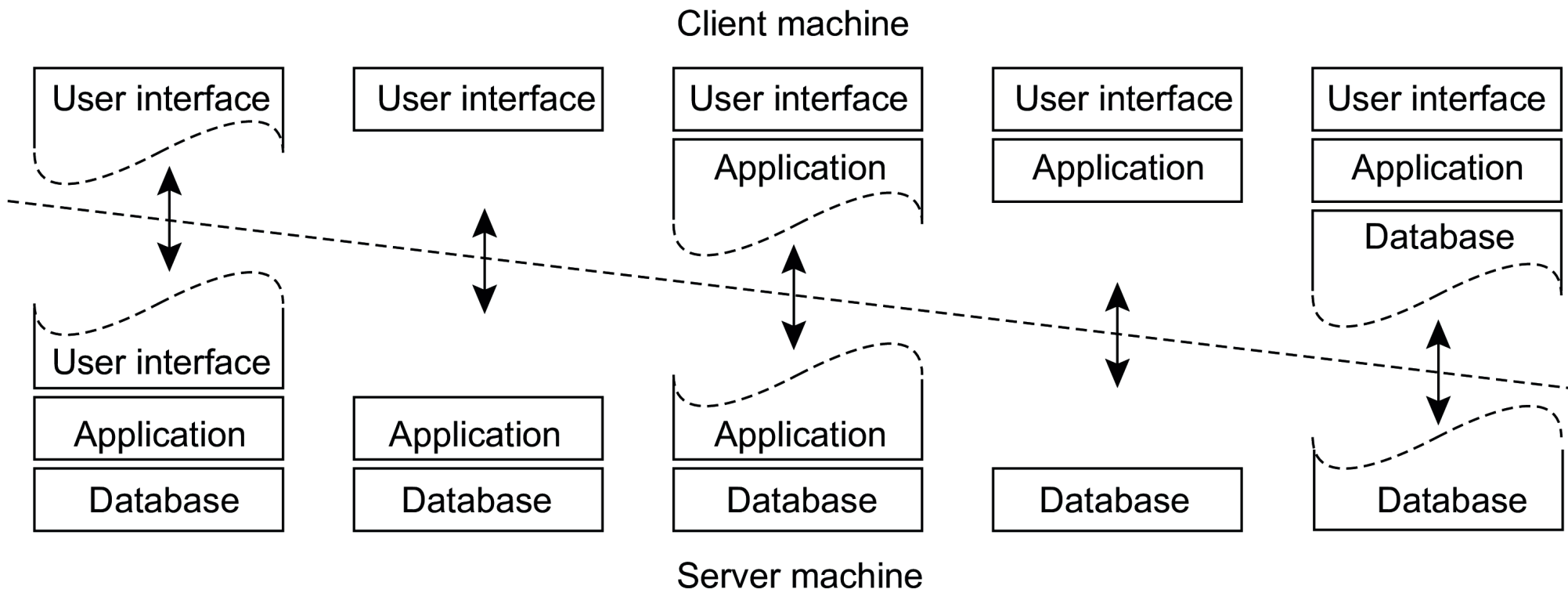
	Temporally coupled	**Temporally decoupled**
Referentially coupled	Direct	Mailbox
Referentially decoupled	Event-based	Shared data space


Component
Component
Subscribe
Notification
delivery
Event bus
Publish
Component


Component
Component
Publish
Subscribe
Data
delivery
Shared (persistent) data space


Publisher
Subscriber
Subscriber
Data item
Subscription
Read/Delivery
Notification
Publish/subscribe middleware
Match


Broker

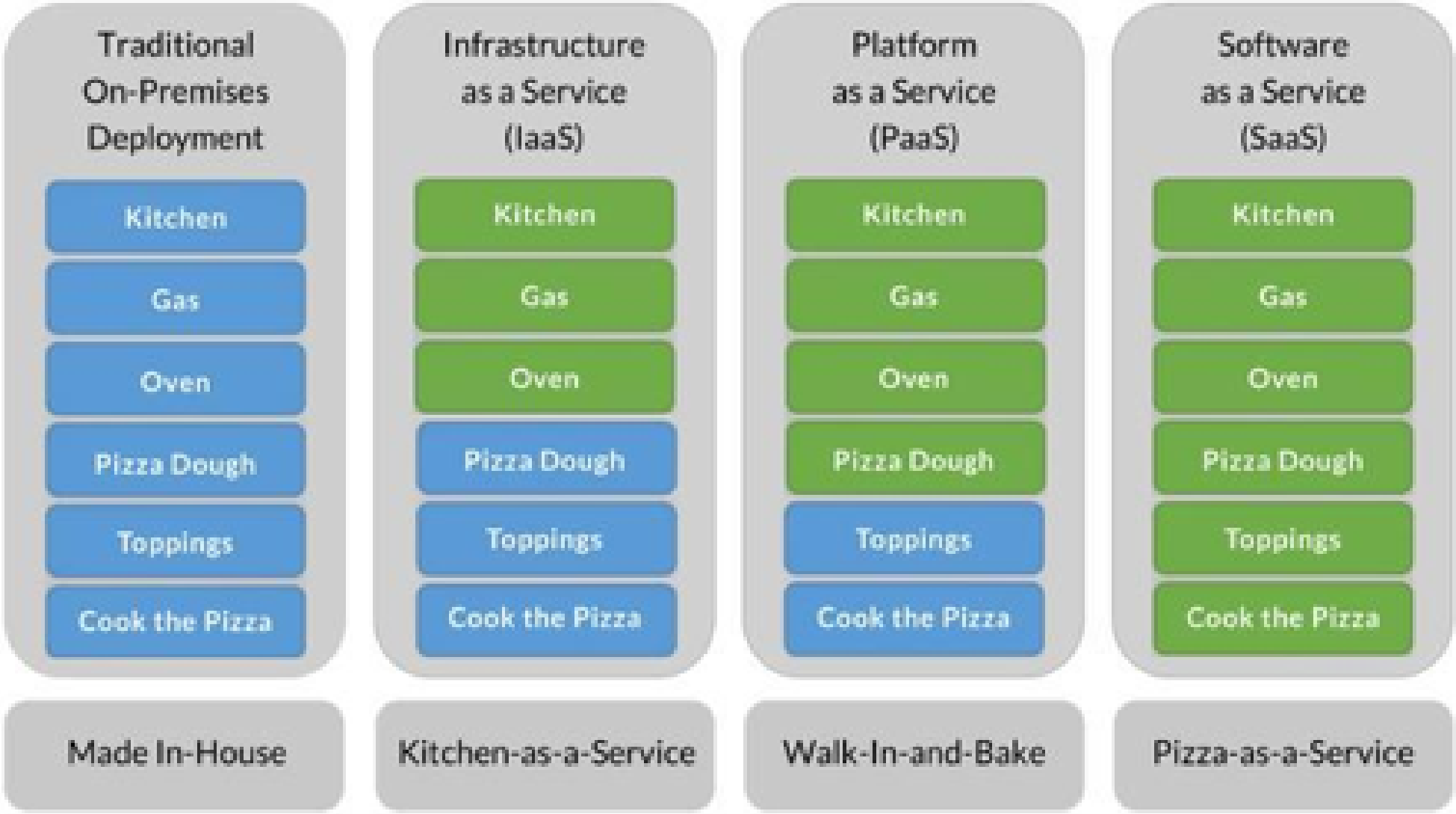

Client machine
User interface
User interface
User interface
User interface
User interface
Application
Application
Application
Database
User interface
Application
Application
Application
Database
Database
Database
Database
Database
Server machine

Cloud Computing

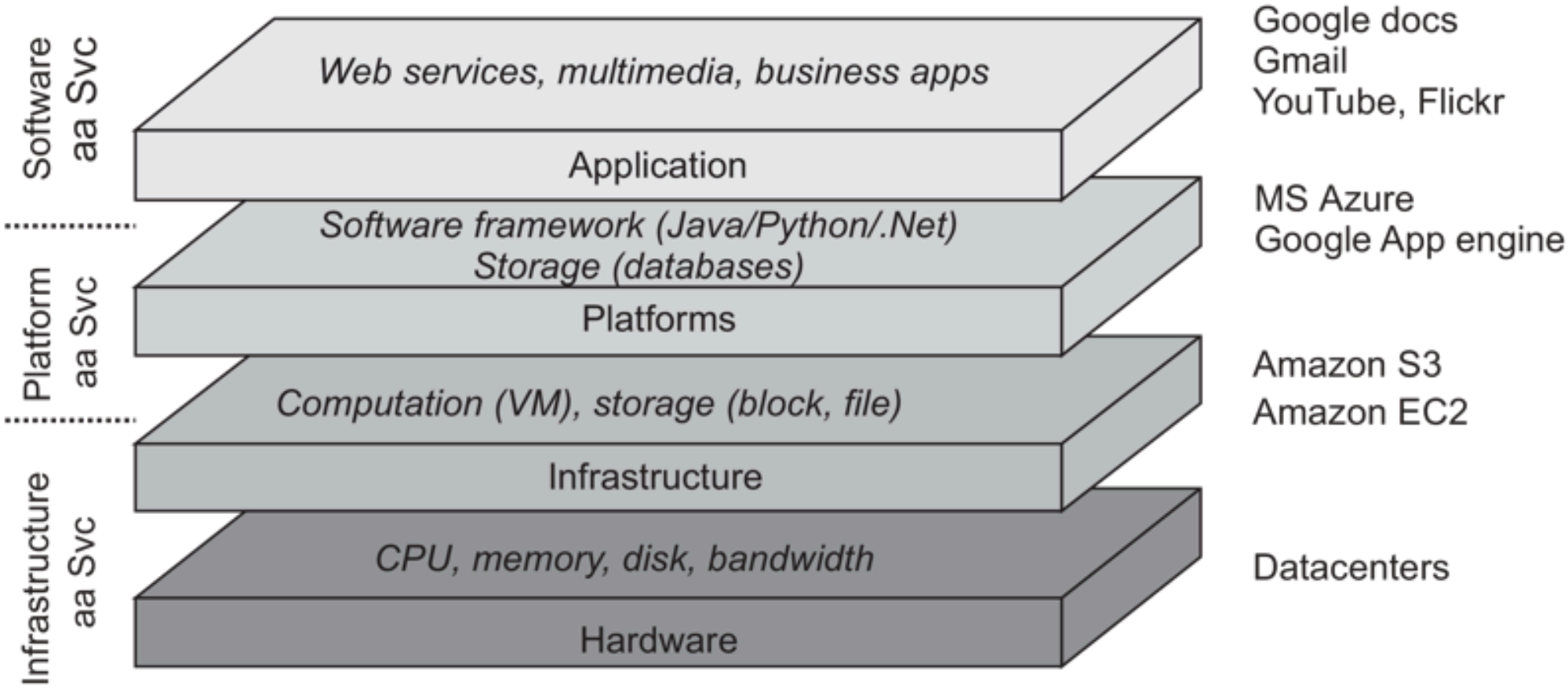
> The entire history of software engineering is that of the rise in levels of abstraction.

-- Grady Booch

New Pizza as a Service

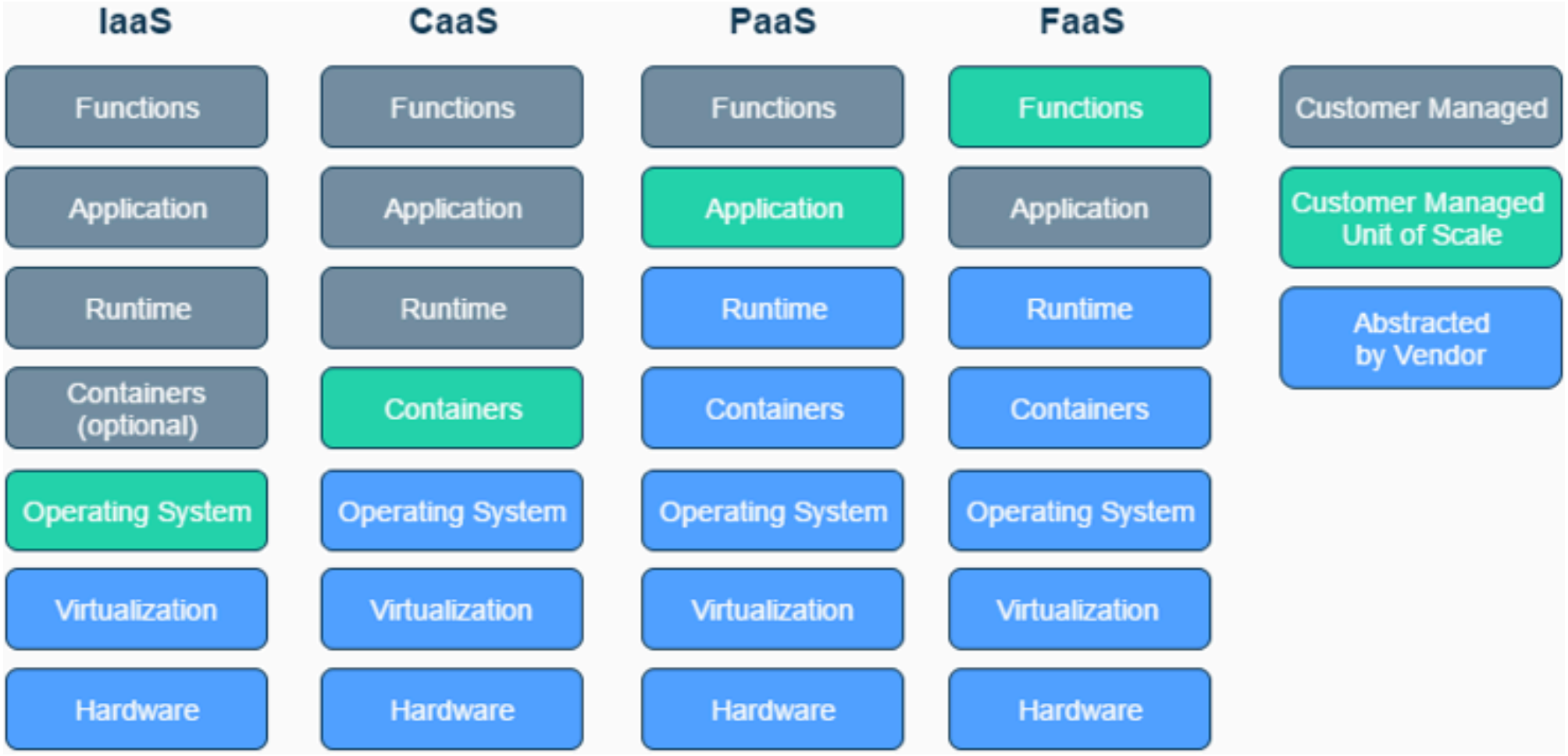


Abstractions

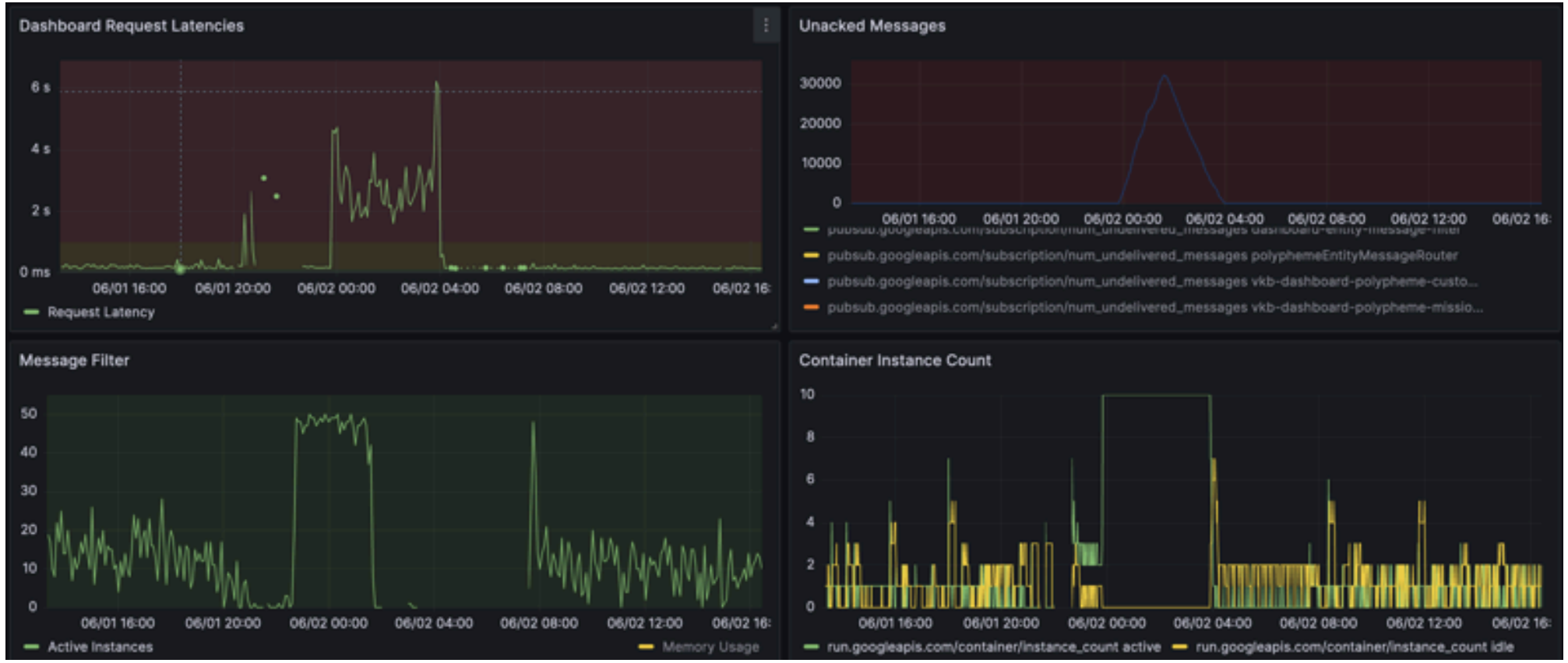


(VanSteen, 2017, S. 30)

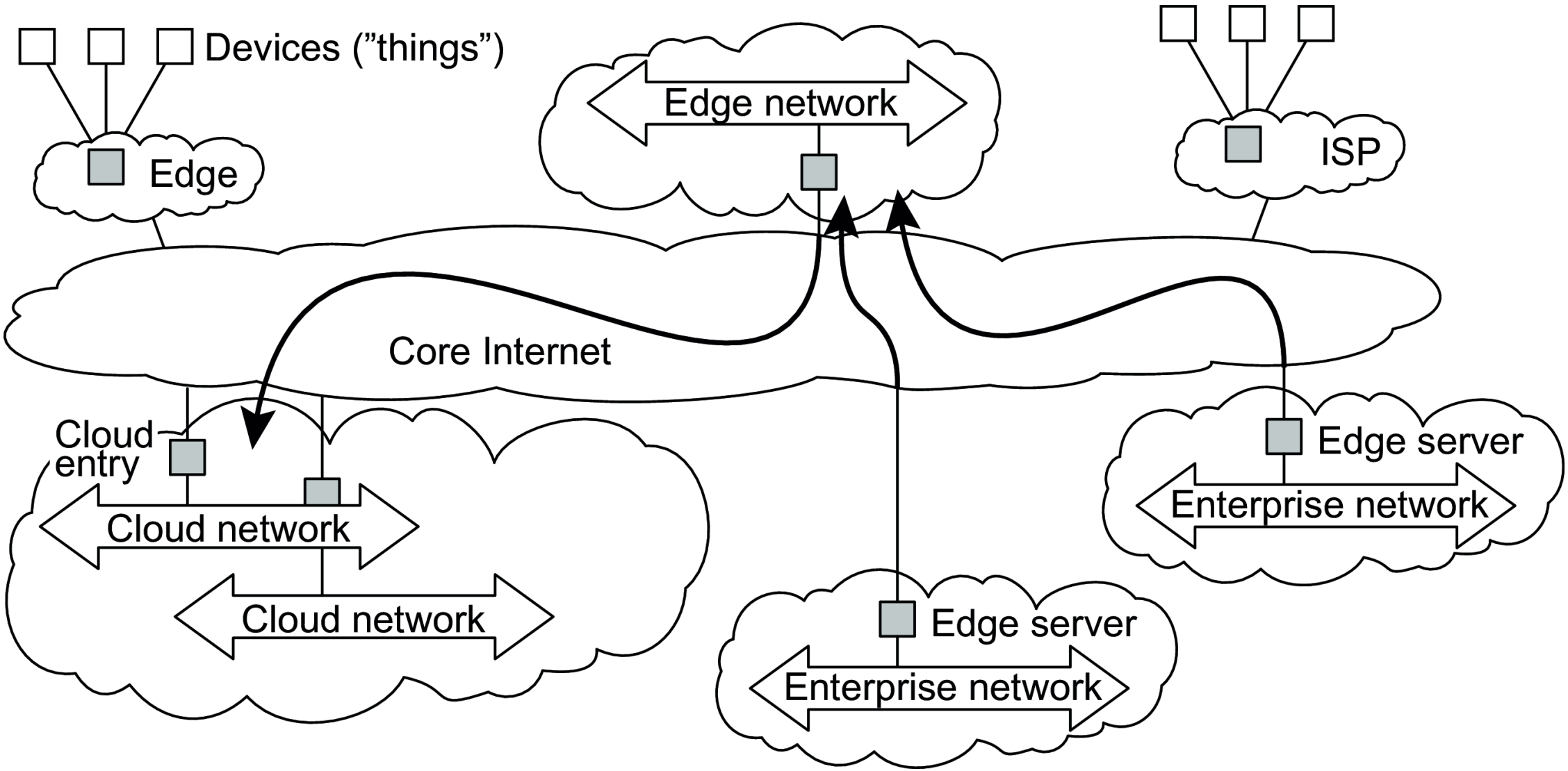
XaaS


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
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

Edge Computing



Koordination

- Tasks können gleichzeitig ausgeführt werden
- Gleichzeitiger Zugriff auf gemeinsame Daten kann in inkonsistenten Daten resultieren

Mutex

- MUTual EXclusion: wechselseitiger Ausschluss
- Einfachste Möglichkeit, Ressourcen für alle anderen zu blockieren
- Critical Section wird mit acquire() und release() umschlossen
- acquire() und release() müssen atomare Operationen sein (Hardwareunterstützung)

Mutex

```
acquire() {
  while (!available)
    /* busy wait */
  available = false;;
}

release() {
  available = true;
}
```

```
do {
    acquire lock
       critical section
    release lock
      remainder section
 } while (true);
```

Semaphore

- Mehr Möglichkeiten als Mutex
- Schützt gemeinsame Ressourcen
- Counting semaphore: Mehrere Ressourcen
- Binary semaphore: Nur eine Ressource
- Ein Zugriff auf eine gemeinsame Ressource wird mit dem Nehmen und Geben umschlossen

Beispiel

```php
$semaphore = $this->createSemaphore($id);

sem_acquire($semaphore);
try {
    $entityPublishTime = $this->getEntityPublishTime($model, $id);

    if ($entityPublishTime < $messagePublishTime) {
        $returnCode = $this->saveStateToModel($model, $state, $data->timestamp);
    } else {
        $returnCode = 3;
    }
} finally {     // make sure to always release semaphore
    sem_release($semaphore);
}
```