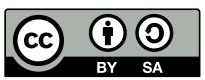
Cloud computing

HA&DR

Aleix Llusà Serra

Master's degree in Machine Learning and Cybersecurity for Internet Connected Systems
Universitat Politècnica de Catalunya
https://epsem.upc.edu

6 d'octubre de 2026



This work is licensed under a Creative Commons "Attribution-ShareAlike 4.0 International" license.

1 Introduction

2 High availability

3 Disaster recovery

4 References

Outages

- **Minor outage**: short-term interruption ⟶ **high availability**
- **Major outage**: long and impacting interruption ⟶ **disaster recovery**
  - Is data lost?
- No recovery: panic, lost backups ⟶ Overall, one must accept existing residual risks

Mitigation strategy

How to tell to business manager about the outages consequences

- Costs: directs costs + additional work hours
- Revenue: lost revenue + lost work hours

Categorisation (Schmidt 2006)

Category	Max. minor outage	Max. major outage
Continuous availability	1 min	2 h
Mission-critical	10 min	8 h
Business-important	1 h(business)	3 d
Business-foundation	1 d(business)	1 w
Business-edge	> 1 w	> 1 m

Why are there outages?

The reasons for many problems (from experience Schmidt (2006))

- Single point of failure: not enough redundancy
- System used beyond design limits
- Software error
- Human error
- Wrong design assumptions

SLA

Service Level Agreement (SLA): requirements for the delivery of IT services

1. Minor outages
   - Availability percentages
   - Maximum outage time per incident
   - Maximum frequency of incidents
2. Major outages
   - Exactly, when it is declared?
   - RTO & RPO
   - Time to full recovery
3. No recovery: remaining risks

This is a summary for HA&DR concepts (we are going to see them)

How to achieve SLA

Balance between (achieving one might compromise the other) :

- Robustness: minimise the potential for failures
  Keep It Simple and Straightforward
- Redundancy: continuity even on failures
  Virtualisation

How to achieve SLA

Layered fault protection (Schmidt 2006)

Category	Fault protection
User environment	Disaster recovery
Administration environment	Disaster recovery
Application	Distributed, failover, clustering
Middleware	Clustering
Infrastructure	Independent high availability
Operating system	Clustering
Hardware	Redundant components
Physical environment	UPS, backup data center

Fault protection sufficient (theoretically) at Application level and above when it is fault-tolerant, distributed, multiplatform, failover, and has clustering: $\longrightarrow$ Our purpose on this course, oriented to virtualisation and microservices architecture.

High availability (HA)

Definition (HA)

High availability is the characteristic of a system to protect against or recover from minor outages in a short time frame with largely automated means (Schmidt 2006)

- Automated recovery
- A user can notice the failure
- Failure of non essential component
  - Redundancy
  - Not a single point of failure

Continuous availability: subset for extreme risk applications, extreme fault tolerant

Quality of IT systems (Schmidt 2006)

- Reliability: probability that a system will be available over a time interval (without planned nor unplanned downtimes)

  Mean Time Between Failures (MTBF) (Schmidt 2006, Appendix A)

- Availability: how long a service is available for use

$$\text{availability} = \frac{\text{uptime}}{\text{uptime} + \text{downtime}}$$

  downtime: (scheduled+unscheduled)

- Serviceability: ease to repair

100% Availability

Definition

Aiming for 100% uptime is the wrong target. Pick something like 99.9% or 99.999% as a business or product trade-off, define and monitor your actual uptime, and use that “budget” as an input to how aggressively you’re willing to push risky releases. (Winters, Manshreck i Wright 2020)

Nines of Availability

90% (one) 99% (two) 99.9% (three) 99.99% (four) 99.999% (five)

Availability percentages, maximum downtime (Schmidt 2006)

	24×7		24×6		14×5	
SLA (%)	Monthly	Yearly	Monthly	Yearly	Monthly	Yearly
99.0	7.3h	3.7d	6.3h	3.1d	3.0h	1.5d
99.5	3.7h	1.8d	3.1h	1.6d	1.5h	18.3h
99.8	1.5h	17.5h	1.3h	15h	36.6min	7.3h
99.9	43.8min	8.8h	37.6min	7.5h	18.3min	3.7h
99.99	4.4min	52.6min	3.8min	45.1min	1.8min	21.9min
99.999	26.3s	5.3min	22.6s	4.5min	11s	2.2min
99.9997	7.9s	1.6min	6.8s	1.4min	3.3s	39.4s

Maximum outage time per incident or maximum frequency of incidents: perhaps sometimes it is more important than global availability?
On weekends?, on working hours?, Is it ok large downtime for maintenance on weekends?

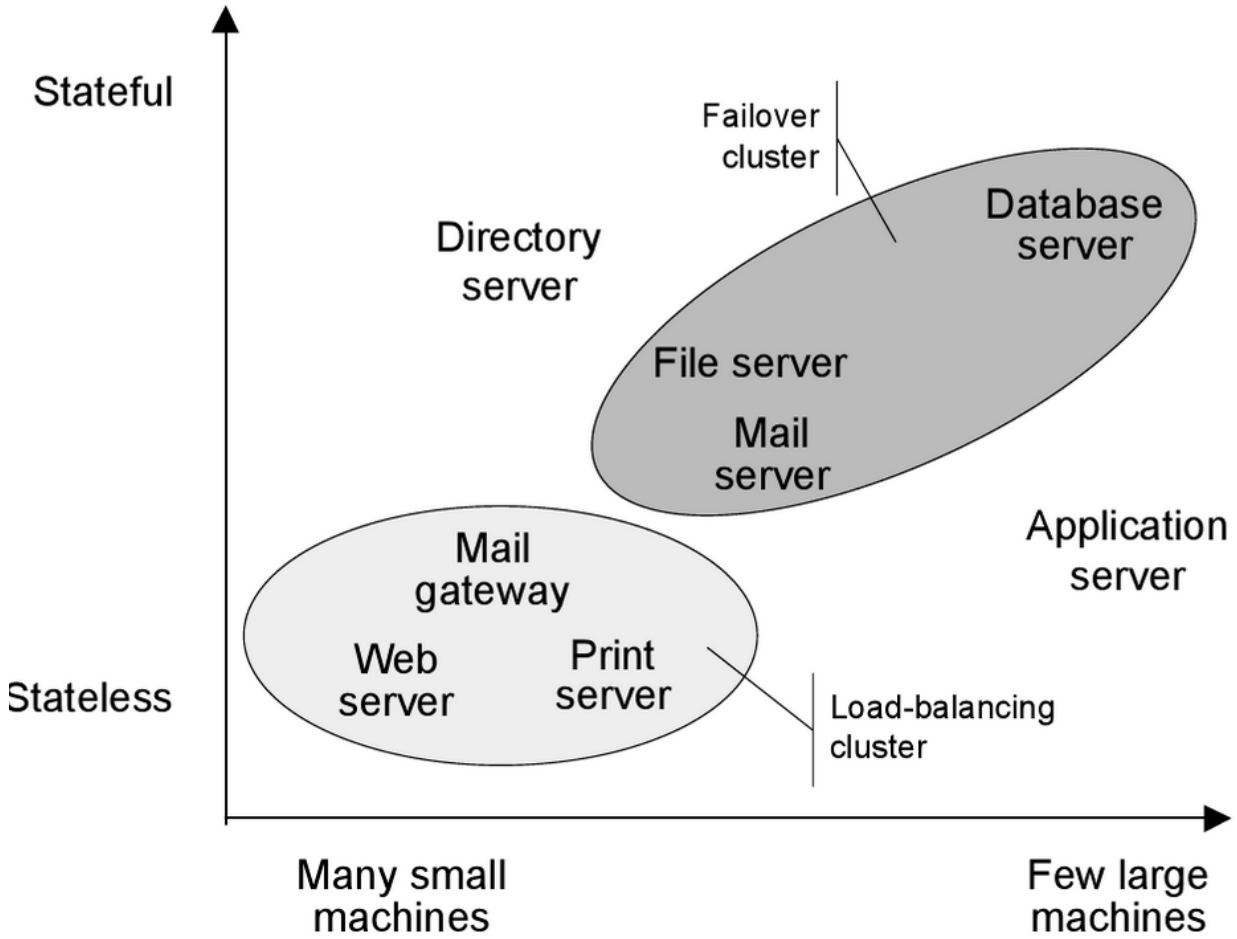
Host clustering

- Failover clusters: move from one host to another
- Load balancing clusters (server farms): provide on several hosts

Cluster usages (Schmidt 2006)

Failover cluster	Load-balancing cluster
Persistent data	Without state
Generic	Specific
Recovery from failures	Recovery from failures
No scaling	Horizontal scaling
Big servers	Small servers

Host clustering

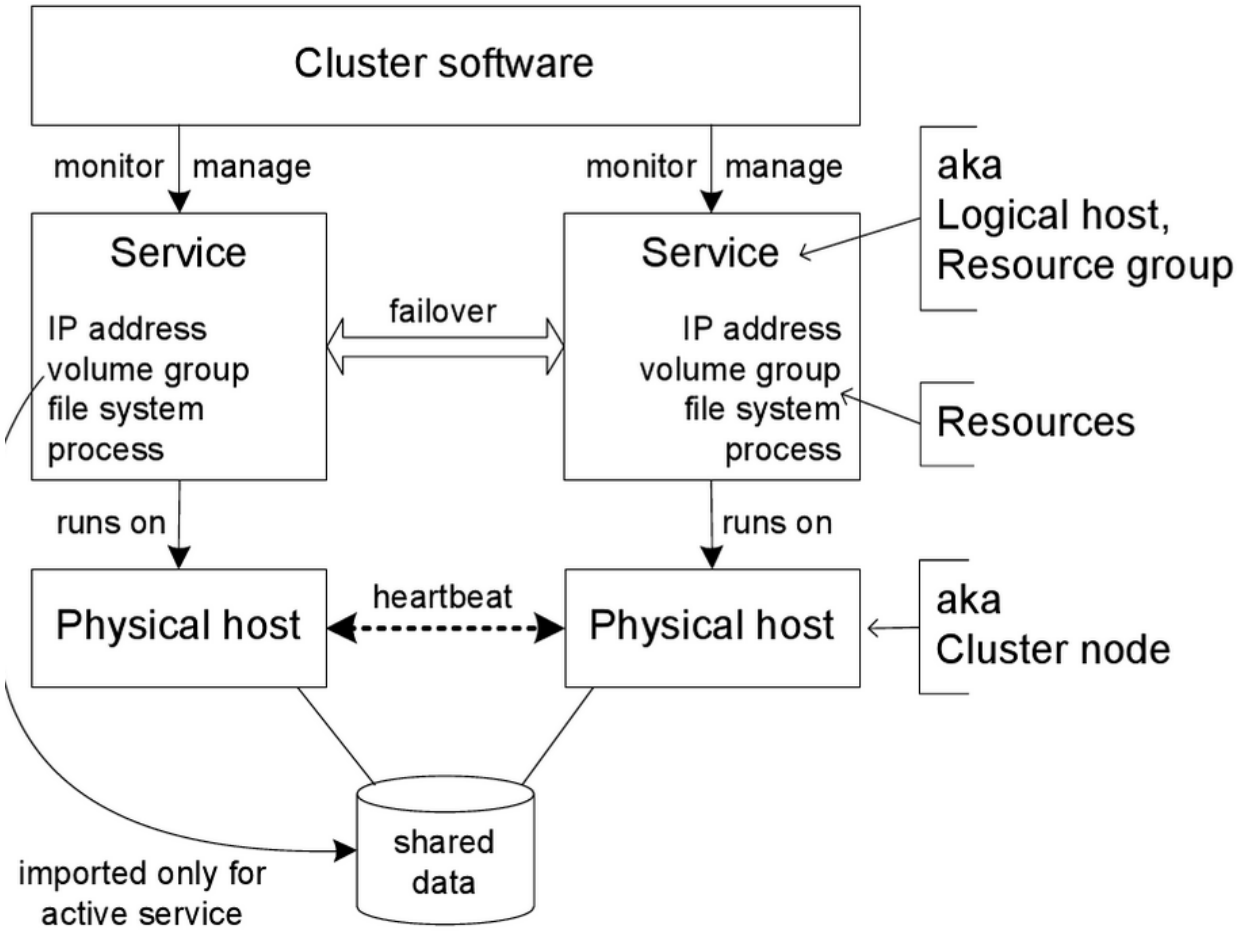


© Schmidt (2006)

Load balancing

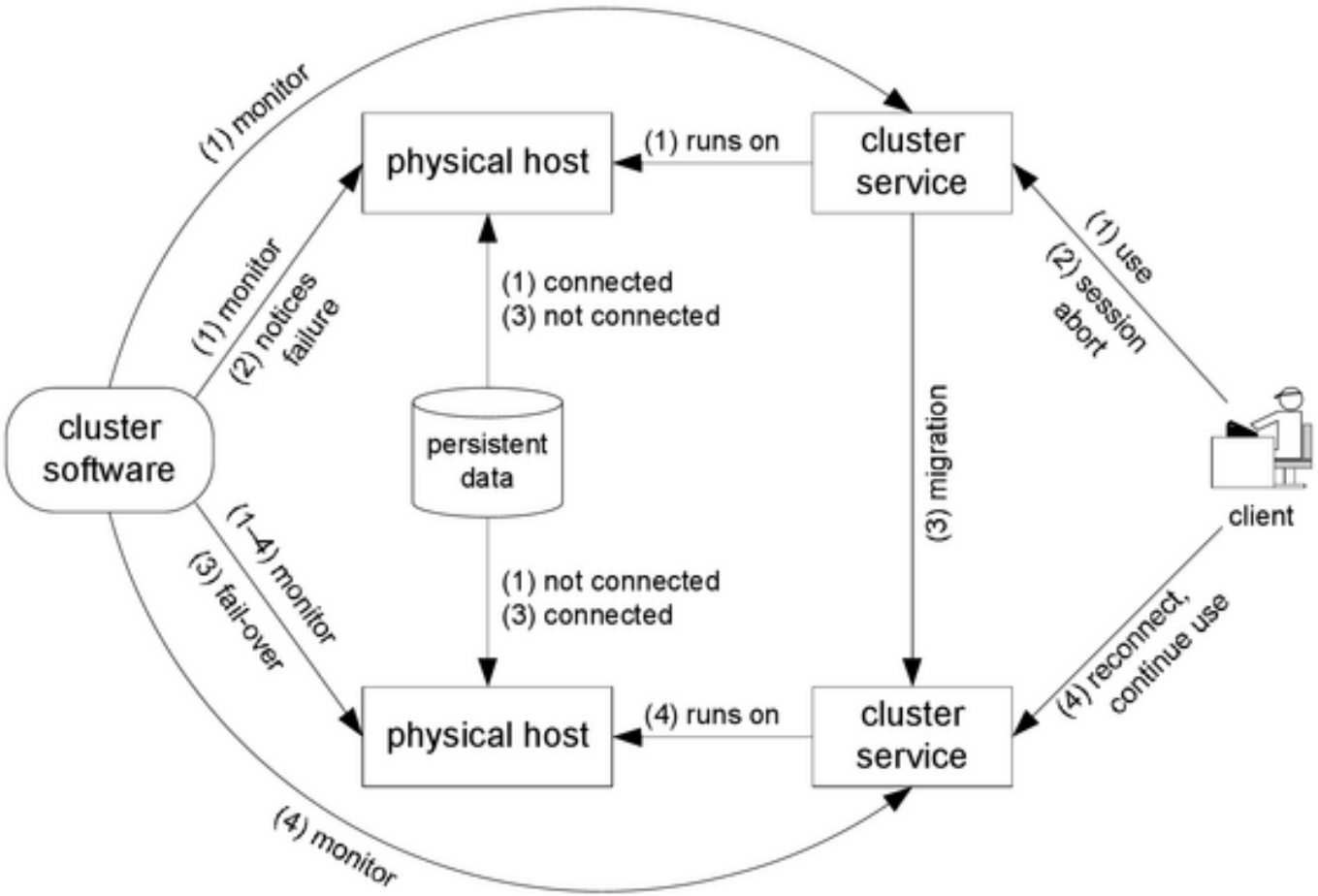
- More than one host providing the same service
- Improve performance, high efficiency
- Also called traffic management or application-level switching
- Generic load balancers may not be appropriate to DBMS
- Approaches:
  - DNS based (from load sharing without state information to CDN)
  - IP load balancers or hardware-based load balancing
  - Reverse proxies or software-based load balancing

Failover clusters



© Schmidt (2006) Failover clusters need hardware independence

Failover clusters



© Schmidt (2006)

Automating HA

Easy on stateless services (or transient state, or caching) more difficult on stateful ones (synchronisation, consensus, shard data…)

- Virtualisation as an abstraction: full or containers
- Automated orchestration: deploy on any of the available machines
- Replicated software unit: migrate all process to other machine, e.g. for maintenance purposes
- Rightsizing: on peak loads for a service scale up the service (maybe scale down others) (or multicloud for renting temporary machines)
- Load balancing between replicas and retries
- Service discovery: internal networking, not colliding ports
- Autohealing policy: kill processes failing a healthcheck
- Monitoring global stats and unified logging

More in Winters, Manshreck i Wright (2020, Chapter 25 Compute as a Service), e.g. read about the “pets versus cattle”.

Disaster recovery (DR)

Definition (DR)

Disaster recovery is the ability to continue with services in the case of major outages, often with reduced capabilities or performance (Schmidt 2006)

- Manual recovery
- Unusual circumstances
- Continuation of critical services
- New operation from a backup system

Data loss

High availability prevents and recovers from minor outages. However HA can fail:

- Redundant components failing at the same time
- Unexpected single point of failures
- Cascading effects
- Great environmental disasters: e.g. earthquakes
- Human errors
- Data loss, including inconsistent data
  - Persistent data
  - Ephemeral data: not a disaster (hope not), can be reentered
    E.g. a shopping basket you were about to checkout or the code you have produced the last hour

Then be prepared for Disaster Recovery. It isn't about preventing nor protecting: it's about restoring, in the worst case from zero.

Mitigation strategy

- Declaration of disaster: someone should decide that
- Essential/Basic services restoration
- Recovery Time Objective: time needed to recover
- Recovery Point Objective: data to be restored (some might be lost)
- Full service recovery time: all back to normal

Categorisation (Schmidt 2006)

Category	RTO max. outage	RPO max. data loss
Mission-critical	8 h	2 h
Business-important	3 d	1 d
Business-foundation	1 w	1 d
Business-edge	> 1 w	1 w

Planning

Important! Test regularly your disaster recovery planning.

Make sure your backups are being created correctly. Test from them.

Exercise

Although the cloud providers (of course) charge more than the bare cost of the metal to recoup the management expenses, they have the expertise already built up, and they are sharing it across multiple customers. (Winters, Manshreck i Wright 2020)
One significant concern when choosing a cloud provider is the fear of lock-in—the provider might suddenly increase their prices or maybe just fail, leaving an organisation in a very difficult position. One of the first serverless offering providers, Zimki, a Platform as a Service environment for running JavaScript, shut down in 2007 with three months' notice. (Winters, Manshreck i Wright 2020)
Exercise: Go to a cloud provider and estimate the price for deploying your services.

References


Schmidt, Klaus (2006). High availability and disaster recovery. Concepts, design, implementation. Berlin: Springer. ISBN: 9783540345824. URL: https://discovery.upc.edu/permalink/34CSUC_UPC/8e3cvp/alma991005248477406711.


Winters, Titus, Tom Manshreck i Hyrum Wright (2020). Software Engineering at Google. O'Reilly. ISBN: 9781492082798. URL: https://discovery.upc.edu/permalink/34CSUC_UPC/8e3cvp/alma991005250023406711.