Cloud computing

From the resource infinity paradigm to the fluctuation paradigm

Hélène Coullon

Associate professor (HDR), IMT Atlantique

Source of inspiration

“Antidote to the cult of performance. Robustness from nature.”

— Olivier Hamant, 2024

Outline

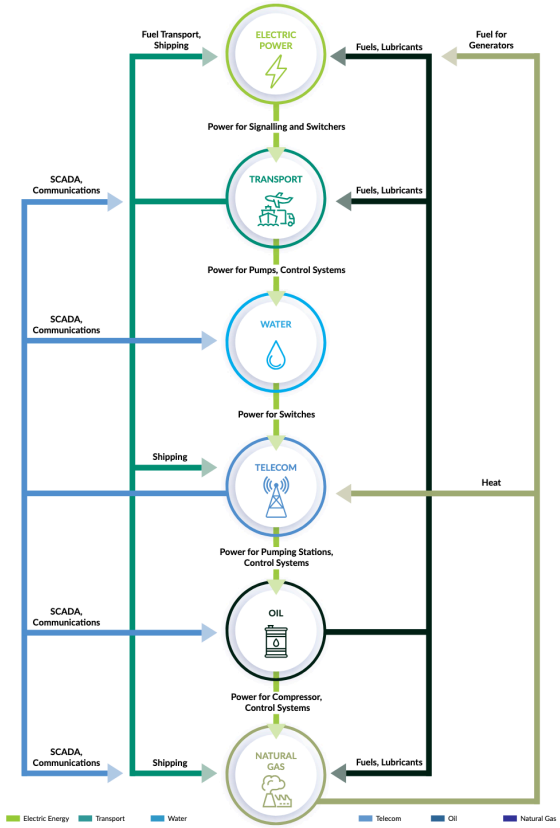
1. Cloud computing and resilience

1.1 Why this topic?

Societies rely on digital infrastructures

1.1 Why this topic?

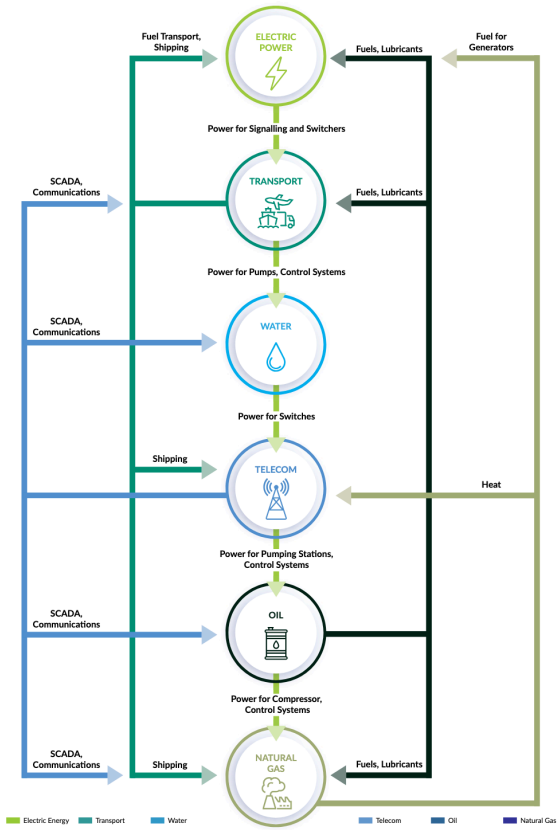
Dependencies between critical infrastructures

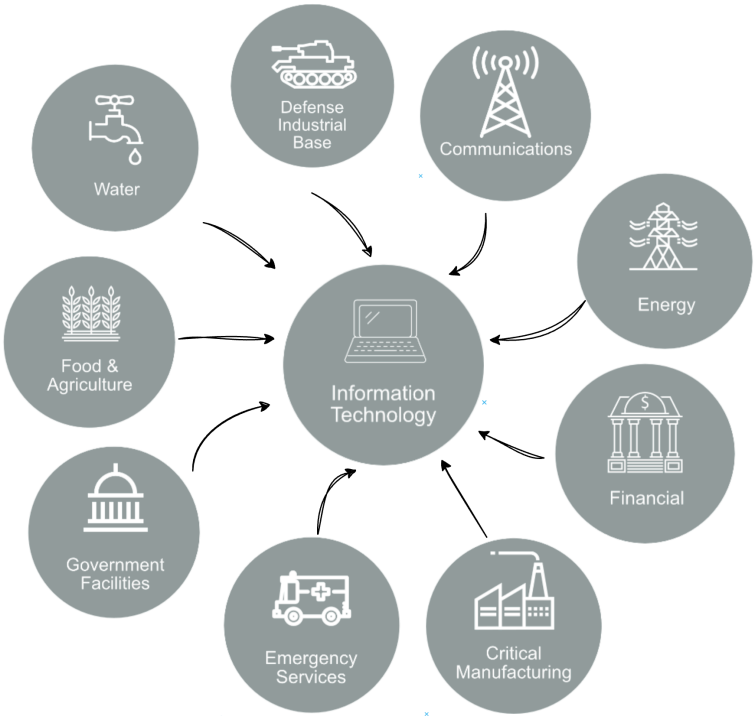


Source: National Strategy on the Resilience of Critical Entities 2026-2029

1.1 Why this topic?

Dependencies between critical infrastructures

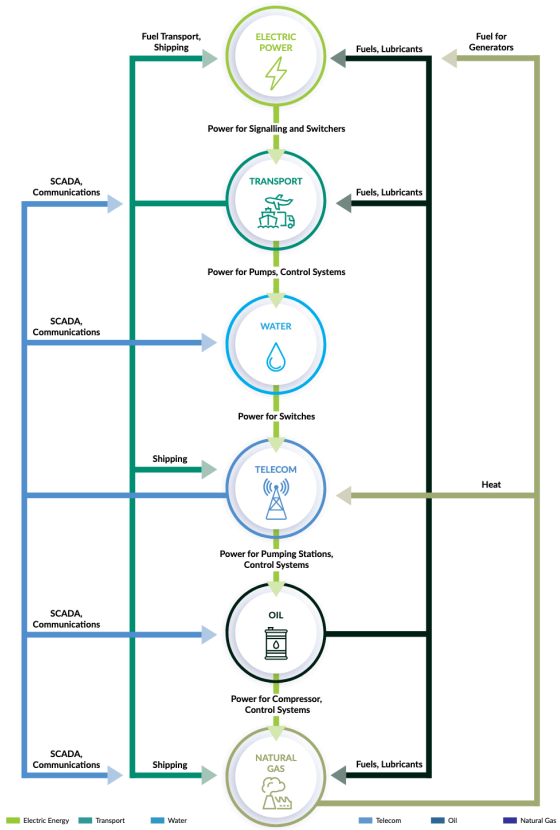




Source: National Strategy on the Resilience of Critical Entities 2026-2029

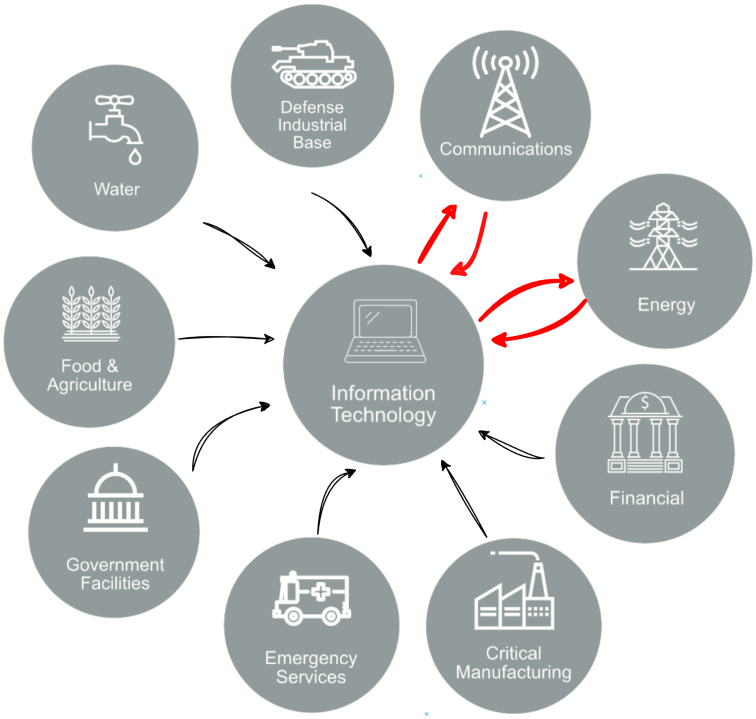
1.1 Why this topic?

Dependencies between critical infrastructures



Source: National Strategy on the Resilience of Critical Entities 2026-2029

⚠ Loops



1.2 Utility computing and cloud

"[...] computing may someday be organized as a public utility just as the telephone system is a public utility."

— John McCarthy, speaking at the MIT Centennial in 1961

1.2 Utility computing and cloud

"[...] computing may someday be organized as a public utility just as the telephone system is a public utility."

— John McCarthy, speaking at the MIT Centennial in 1961

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

— NIST, 2011

1.2 Utility computing and cloud

"[...] computing may someday be organized as a public utility just as the telephone system is a public utility."

— John McCarthy, speaking at the MIT Centennial in 1961

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

— NIST, 2011

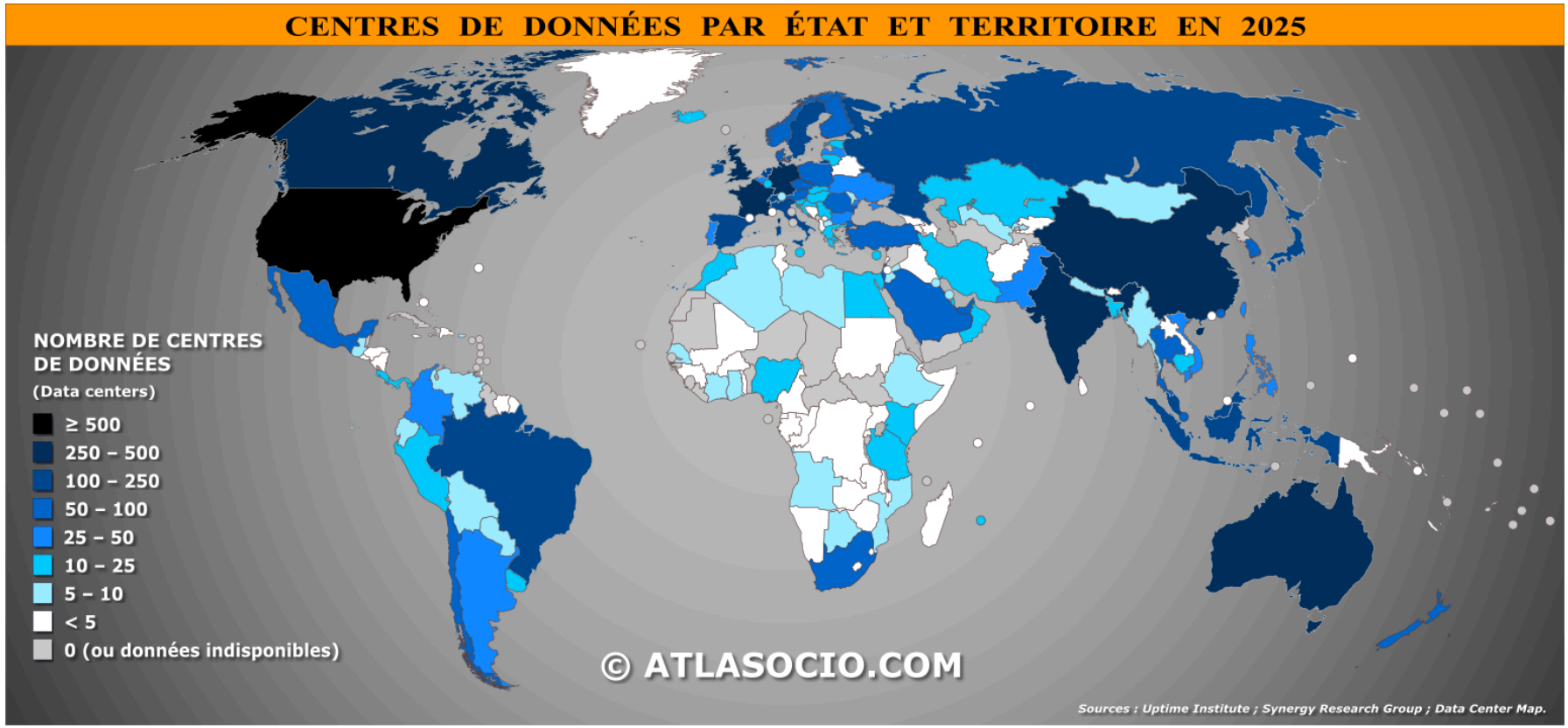
Cloud computing is more a private utility...

1.3 Ground computing

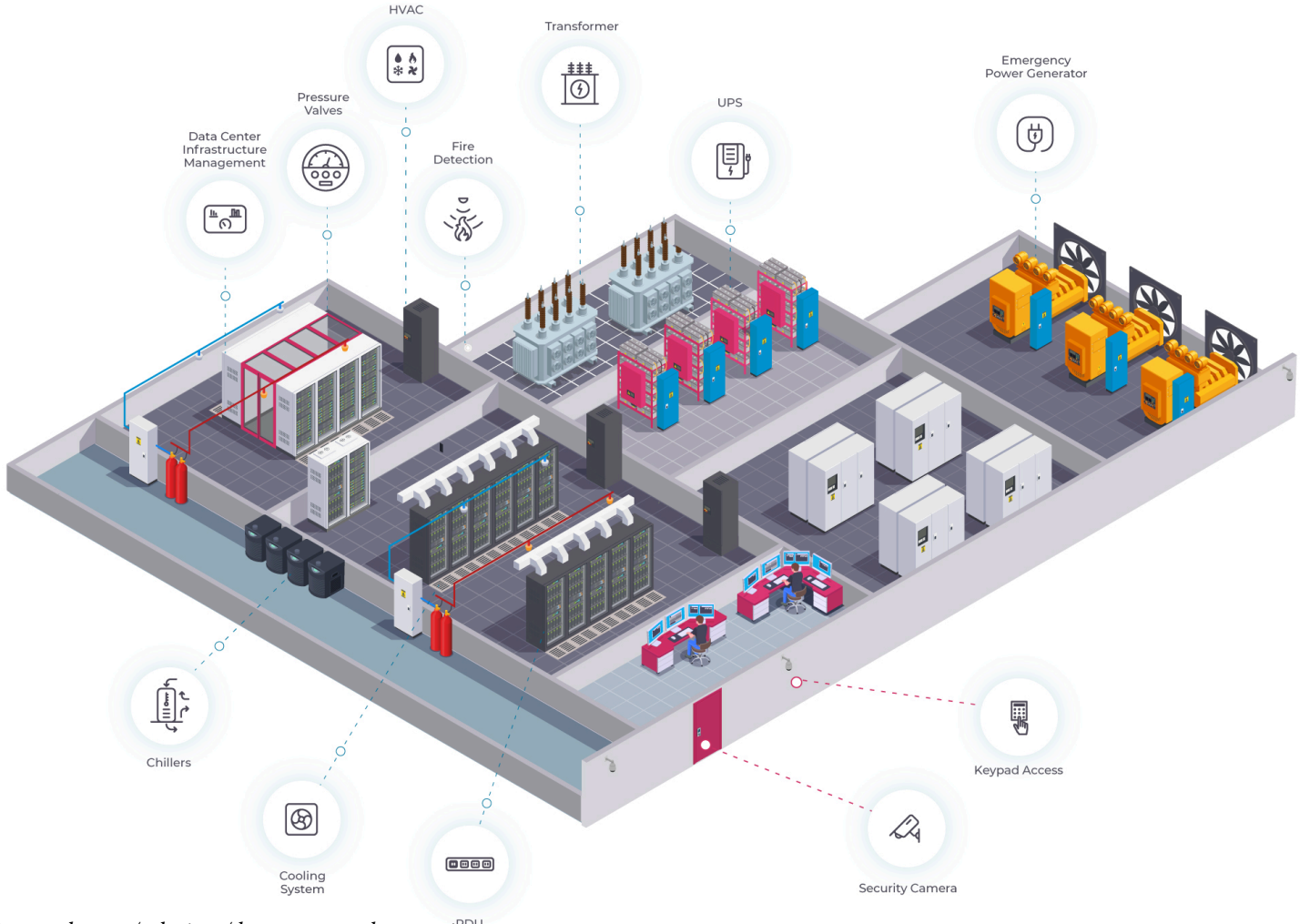
Cloud computing is more ground computing...

1.3 Ground computing

Cloud computing is more ground computing...



1.4 Materiality

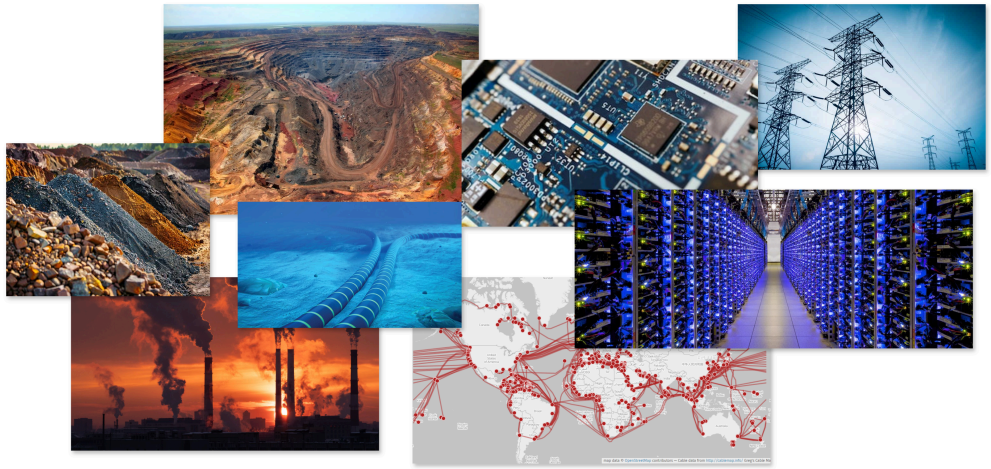


Source: https://www.nozominetworks.com/solutions/data-center-cybersecurity

1.4 Materiality

Source: https://blogs.microsoft.com/blog/2025/09/18/inside-the-worlds-most-powerful-ai-datacenter/
Microsoft's new AI datacenter campus in Mt Pleasant, Wisconsin, 2025

1.4 Materiality



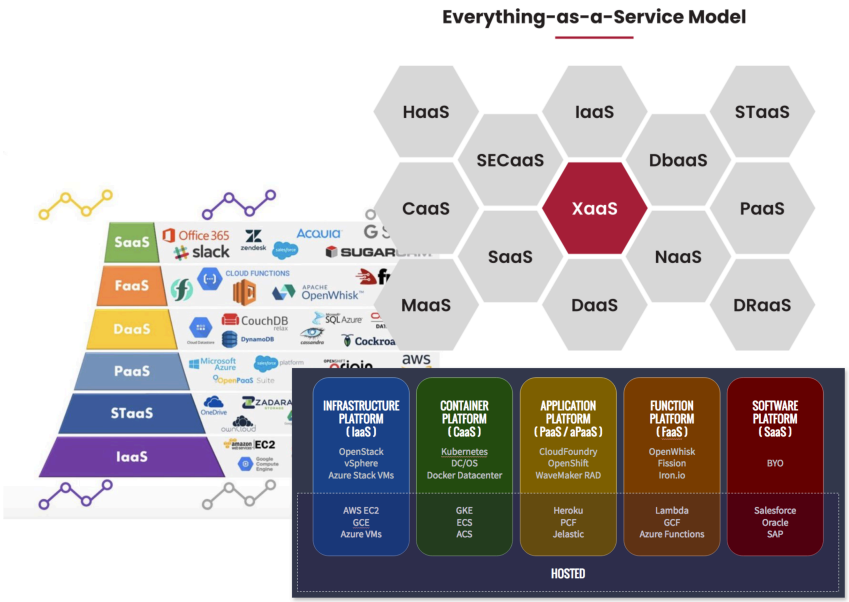
Important ecological impacts...

- ICT part in the global carbon impact: 2.1 to 3.9% in 2021 (+6% to 10% /year)
- ICT part in the electricity consumption of France: 11% in 2020 (52 TWh) (2050, +79%)
  - 65 TWh in 2020 including abroad data centers

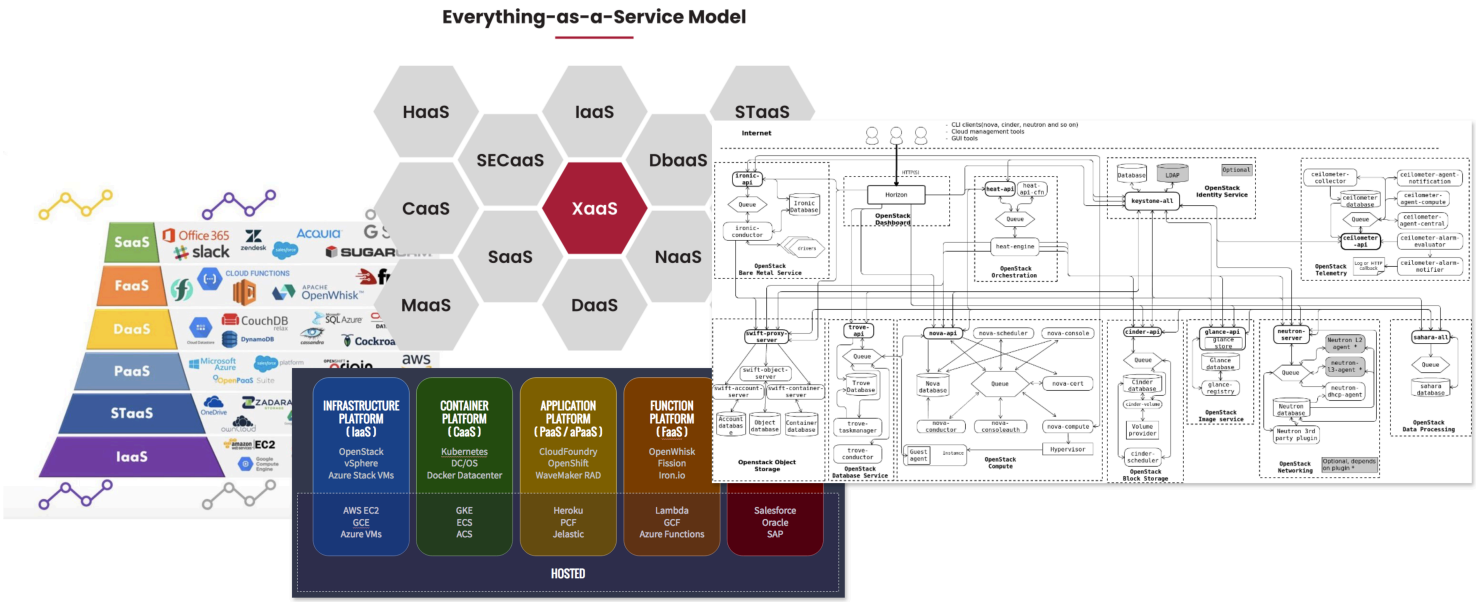
...but also numerous external dependencies inducing fragilities!

Source: Keynote of Anne-Cécile Orgerie at INFRASTRUCTURE 2025

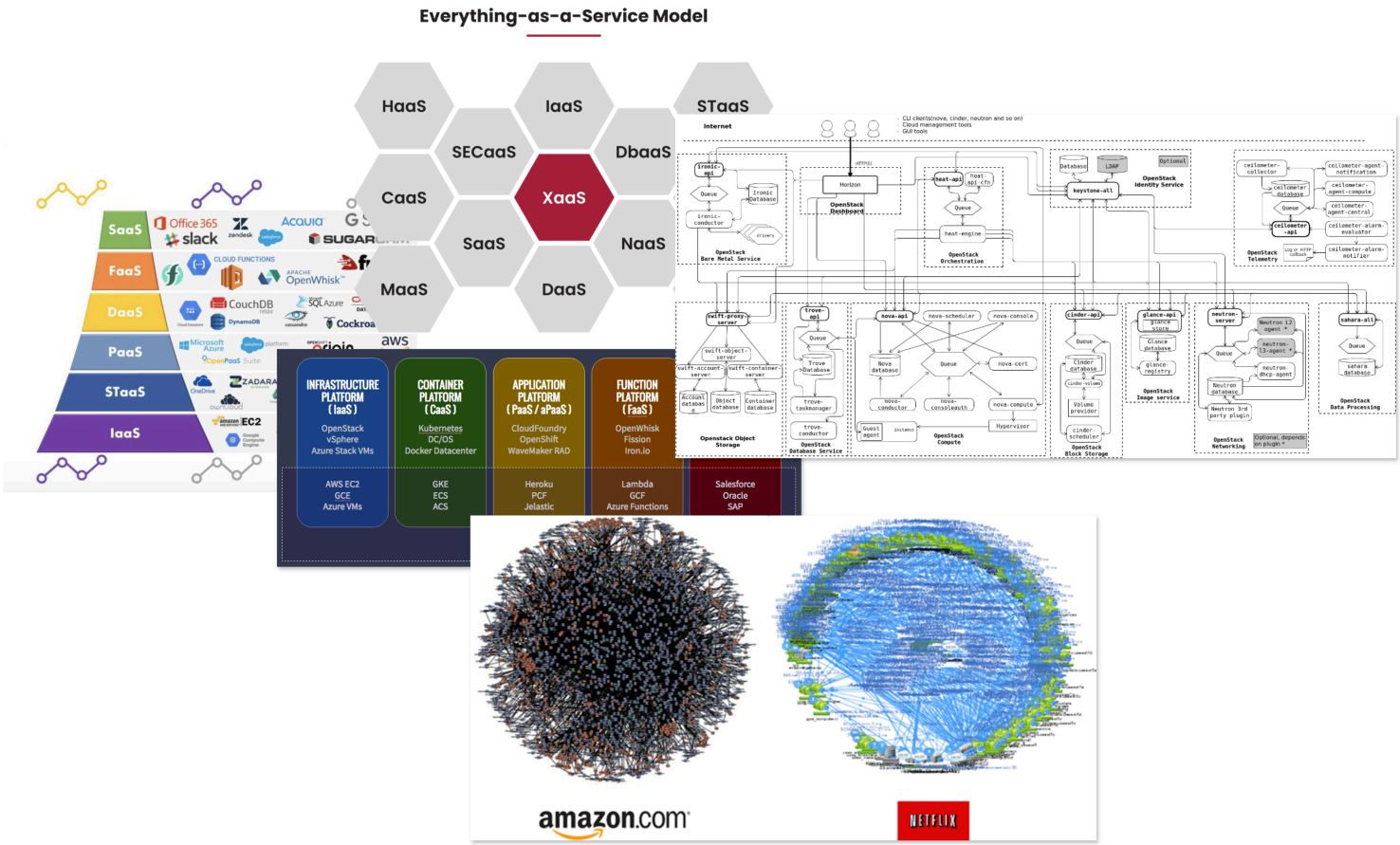
1.5 What about Software?



1.5 What about Software?



1.5 What about Software?



⇨ **Huge Graph of interdependent internal and external services**

1.6 Fragility

Complex evolving Systems with many dependencies

⇨ **Internal and worldwide cascading outages**

1.6 Fragility

Complex evolving Systems with many dependencies

⇨ **Internal and worldwide cascading outages**

1. Network config *2. Undersea cable cuts* *3. Incompatible metadata in CDN*	*Bad automated update quota check system*	*Race condition in Amazon's DynamoDB's DNS*	*Larger than expected file replication*
			
Jan. 2025, 50h Sept. 2025 Oct. 2025, , 10h	June 2025, 3h	Oct. 2025, 15h	Nov. 2025, 8h

1.7 Resilience

Resilience

Resilience is used as an integrative concept that encompasses system robustness, survivability, recovery, and adaptability

Resilience

Resilience is used as an integrative concept that encompasses system robustness, survivability, recovery, and adaptability

SODIs are generally **resilient** (i.e., able to recover after incidents)

⇨ stability/survivability of control services and critical dependencies

- high replication and redundancy
- inifinte resources (computing, energy, water, batteries, etc.)
- stability of external services

2. Fluctuation paradigm

2.1 Polycrises

Amongst others

- climate change
- natural resource scarcity
- geopolitical instability

2.1 Polycrises

Amongst others

- climate change
- natural resource scarcity
- geopolitical instability

Dependability, fault tolerance, reliability

⇨ yes, but ⚠ **new challenges are raised!**

2.1 Polycrises

Amongst others

- climate change
- natural resource scarcity
- geopolitical instability

Dependability, fault tolerance, reliability

⇨ yes, but ⚠ **new challenges are raised!**

Fluctuation

An **irregular** rising and falling in number or amount

2.2 The fluctuation paradigm

- Electrical fluctuations
- Fluctuations in access to water
- Temperature fluctuations
- Fluctuations on hardware supply
- Fluctuation in network quality or connectivity
- Fluctuations to access data, software, services, applications
- etc.

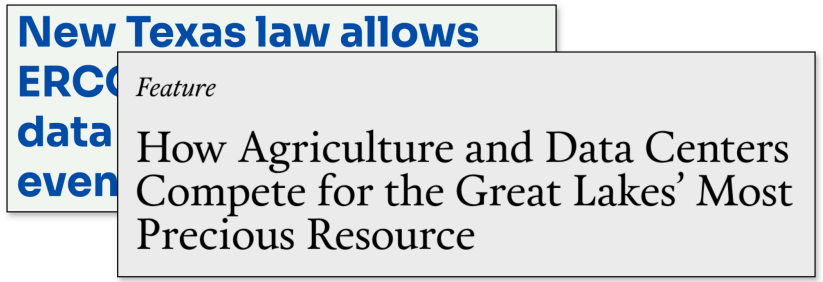
2.2 The fluctuation paradigm

- Electrical fluctuations
- Fluctuations in access to water
- Temperature fluctuations
- Fluctuations on hardware supply
- Fluctuation in network quality or connectivity
- Fluctuations to access data, software, services, applications
- etc.

New Texas law allows ERCOT to disconnect data centers during peak events

2.2 The fluctuation paradigm

- Electrical fluctuations
- Fluctuations in access to water
- Temperature fluctuations
- Fluctuations on hardware supply
- Fluctuation in network quality or connectivity
- Fluctuations to access data, software, services, applications
- etc.



2.2 The fluctuation paradigm

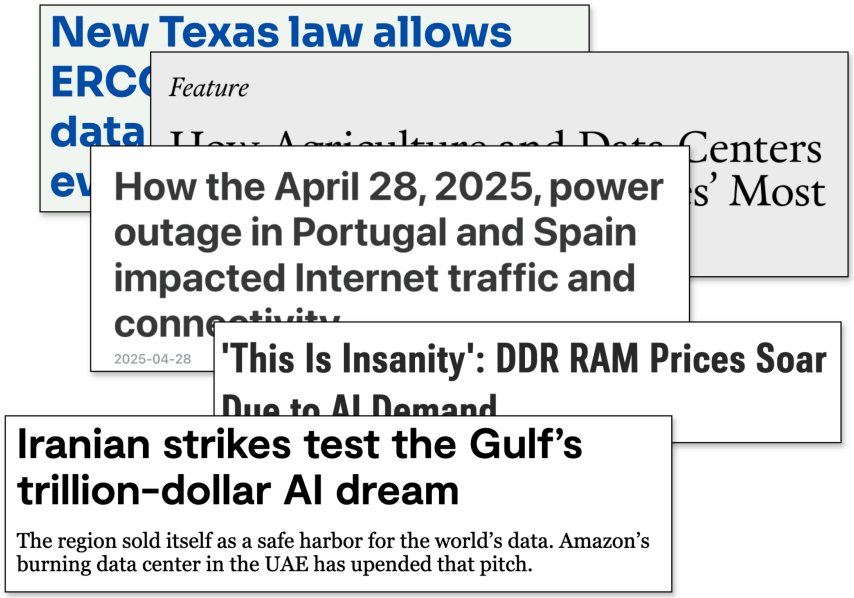
- Electrical fluctuations
- Fluctuations in access to water
- Temperature fluctuations
- Fluctuations on hardware supply
- Fluctuation in network quality or connectivity
- Fluctuations to access data, software, services, applications
- etc.

2.2 The fluctuation paradigm

- Electrical fluctuations
- Fluctuations in access to water
- Temperature fluctuations
- Fluctuations on hardware supply
- Fluctuation in network quality or connectivity
- Fluctuations to access data, software, services, applications
- etc.

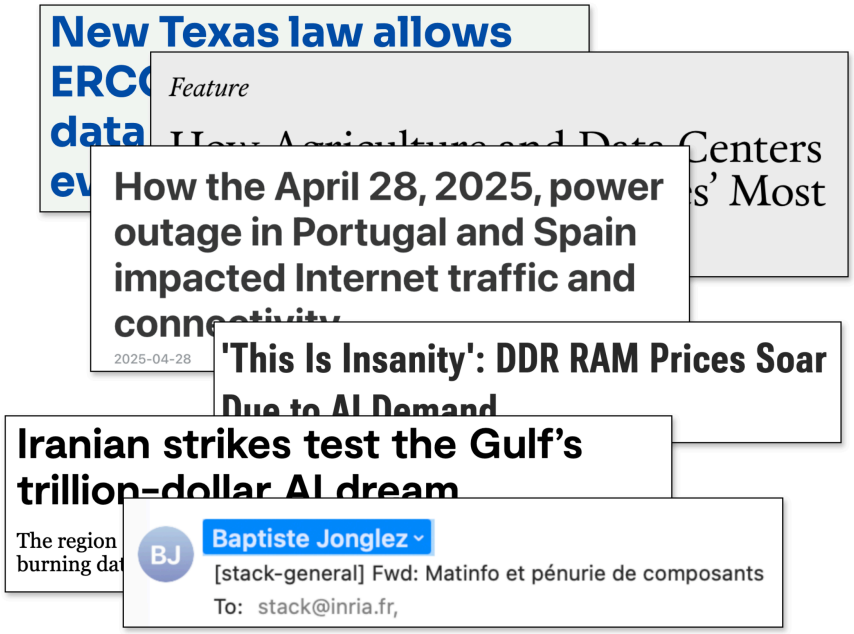
2.2 The fluctuation paradigm

- Electrical fluctuations
- Fluctuations in access to water
- Temperature fluctuations
- Fluctuations on hardware supply
- Fluctuation in network quality or connectivity
- Fluctuations to access data, software, services, applications
- etc.



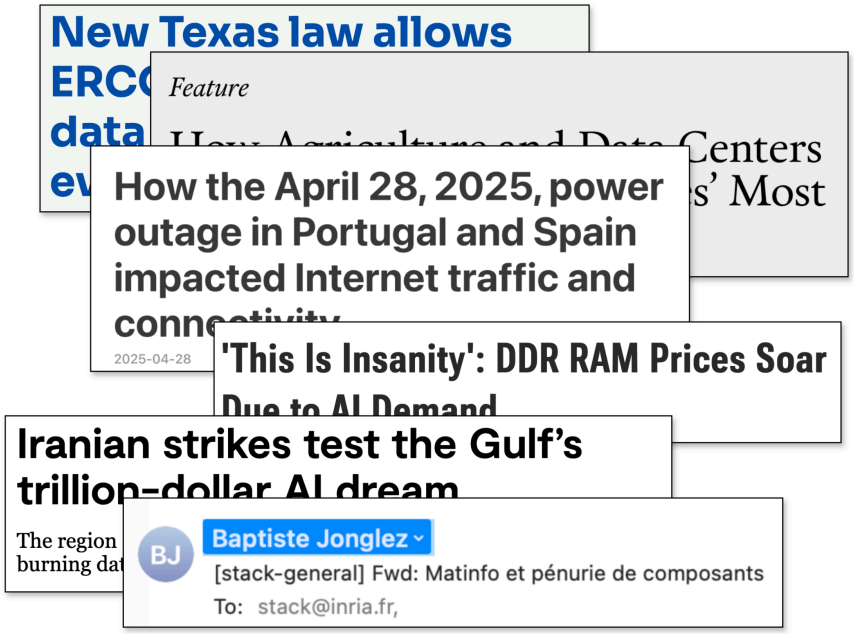
2.2 The fluctuation paradigm

- Electrical fluctuations
- Fluctuations in access to water
- Temperature fluctuations
- Fluctuations on hardware supply
- Fluctuation in network quality or connectivity
- Fluctuations to access data, software, services, applications
- etc.



2.2 The fluctuation paradigm

- Electrical fluctuations
- Fluctuations in access to water
- Temperature fluctuations
- Fluctuations on hardware supply
- Fluctuation in network quality or connectivity
- Fluctuations to access data, software, services, applications
- etc.



"Modern Internet services are composed of hundreds of interdependent systems spanning dozens of geo-distributed data centers. At this scale, seemingly rare natural disasters, such as hurricanes blowing down power lines and flooding, occur regularly"

— Facebook researchers, OSDI 2018 [1]

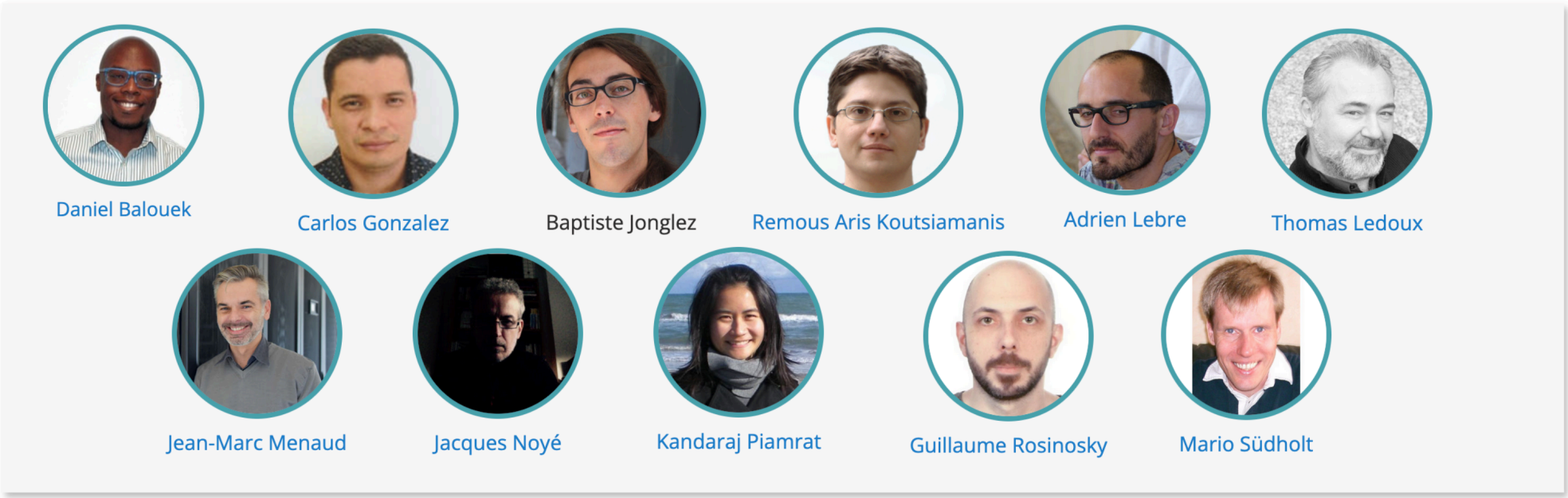
3. RIFT

3.1 New Inria team - RIFT

Resilience of digital Infrastructures to resource FlucTuations

⚠ Still under creation

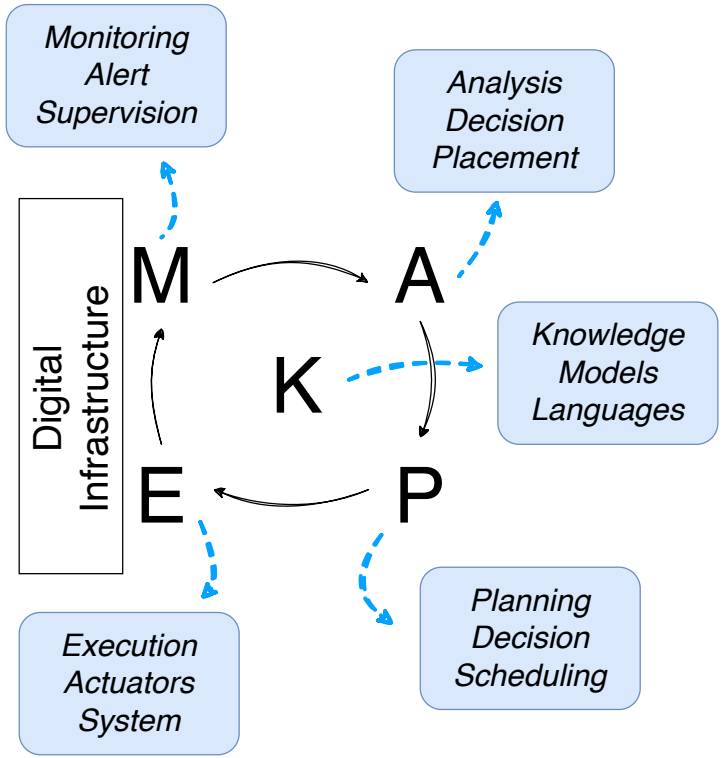
⇨ Following the current STACK research group of Adrien Lebre



3.2 Scope of RIFT

(Resilient) Software systems for SODI management.

Examples: OpenStack, Kubernetes, OpenWISP, OpenFaaS, OpenFL, OpenDaylight etc.



3.3 Research program

- **Axis 1**: Considering limitations and degradations
  - Conectivity, energy, hardware

3.3 Research program

- **Axis 1**: Considering limitations and degradations
  - Conectivity, energy, hardware
- **Axis 2**: Handling the temporal scales of fluctuations
  - urgency, high frequency, long-term permanent

3.3 Research program

- **Axis 1**: Considering limitations and degradations
  - Conectivity, energy, hardware
- **Axis 2**: Handling the temporal scales of fluctuations
  - urgency, high frequency, long-term permanent
- **Axis 3**: Handling the spatial scales of fluctuations
  - geography and territoriality, sovereignty

3.3 Research program

- **Axis 1**: Considering limitations and degradations
  - Conectivity, energy, hardware
- **Axis 2**: Handling the temporal scales of fluctuations
  - urgency, high frequency, long-term permanent
- **Axis 3**: Handling the spatial scales of fluctuations
  - geography and territoriality, sovereignty
- **Axis 4**: Evaluating resiliency
  - Metrics, benchmarks, resilience-oriented platforms

4. Focus on reconfigurations

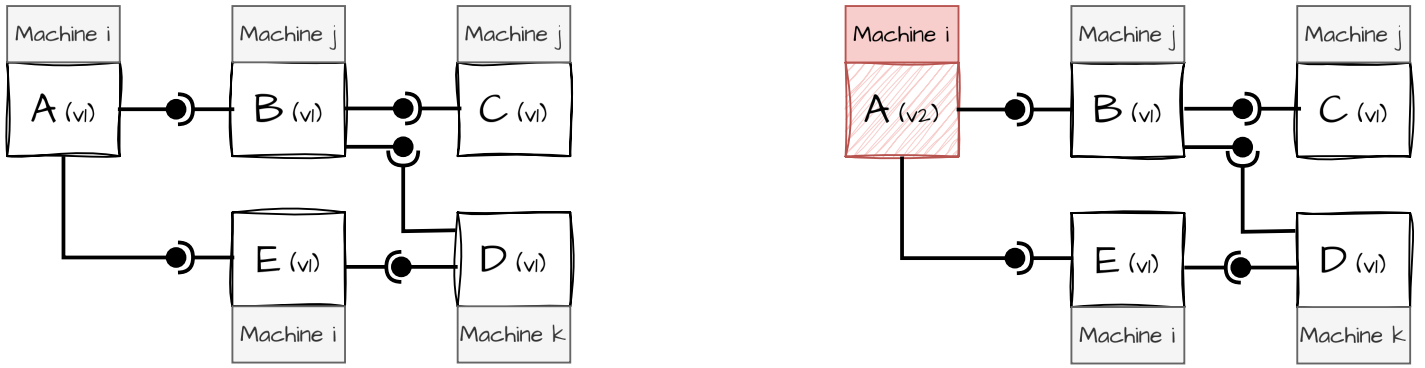
4.1 Dynamic reconfigurations

- Dynamic reconfiguration originates from CBSE [2]
  - models able to dynamically change the assembly of components
  - focus on PE-K of MAPE-K
- Metrics: minimize downtime, mean time to recovery
- Formalization and verification of reconfigurations

4.1 Dynamic reconfigurations

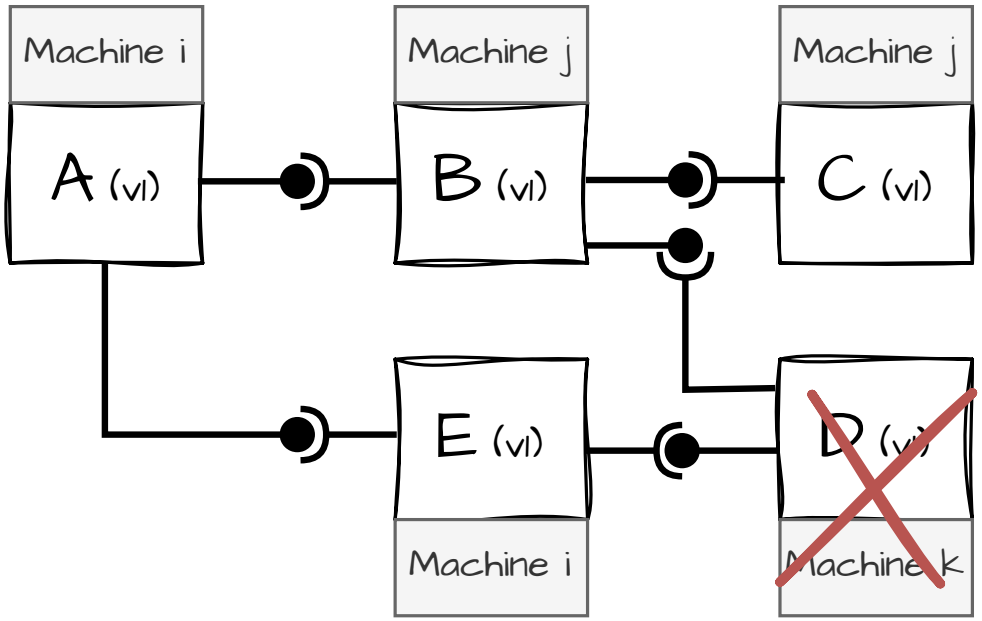
- Dynamic reconfiguration originates from CBSE [2]
  - models able to dynamically change the assembly of components
  - focus on PE-K of MAPE-K
- Metrics: minimize downtime, mean time to recovery
- Formalization and verification of reconfigurations

⇨ **Concrete example: update component A in the assembly**



4.2 Concrete examples

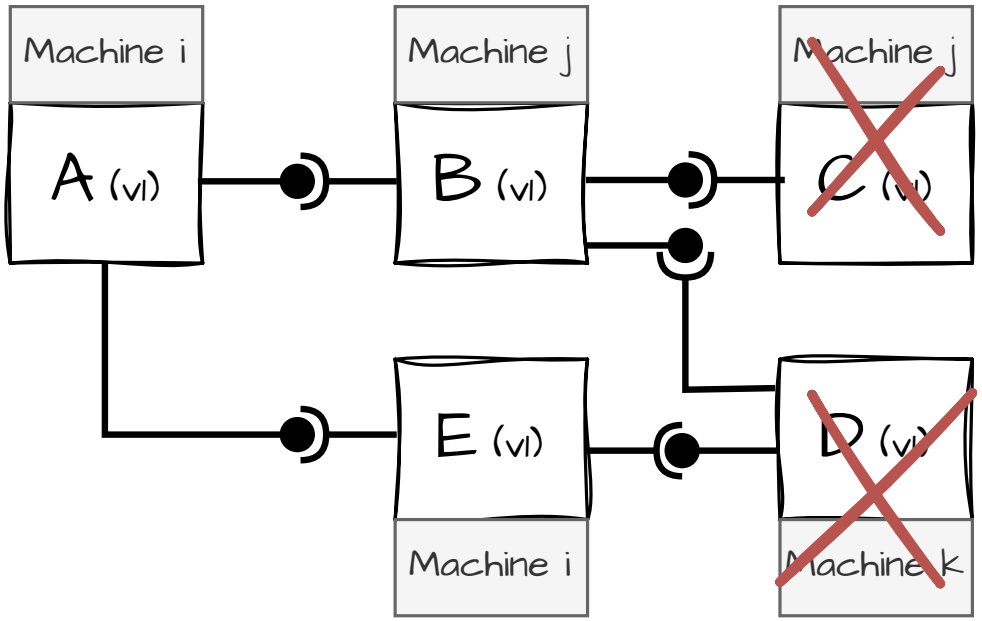
Version 1



delete(D)

4.2 Concrete examples

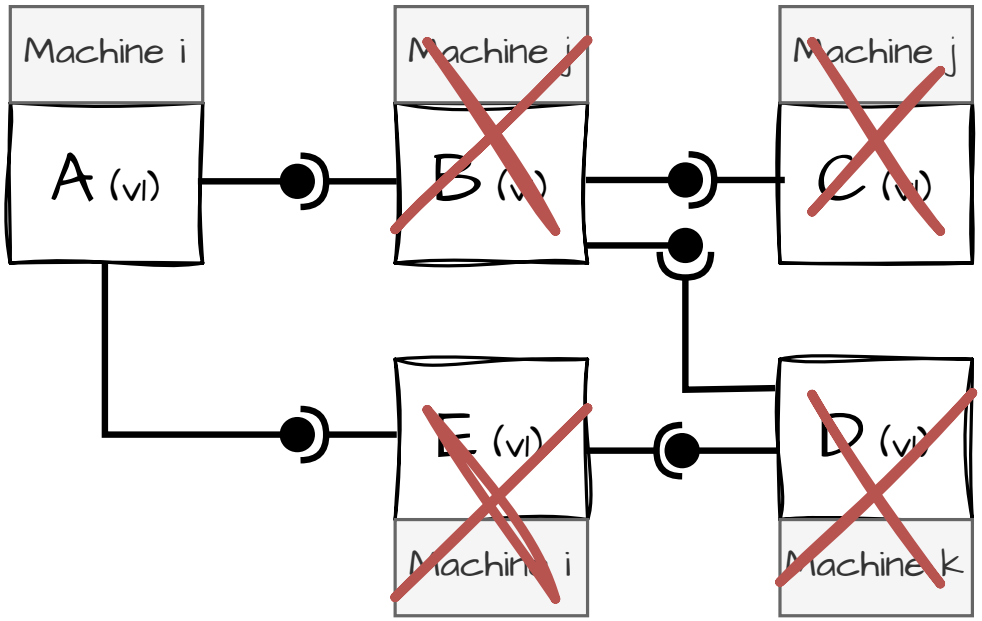
Version 1



delete(D); delete(C)

4.2 Concrete examples

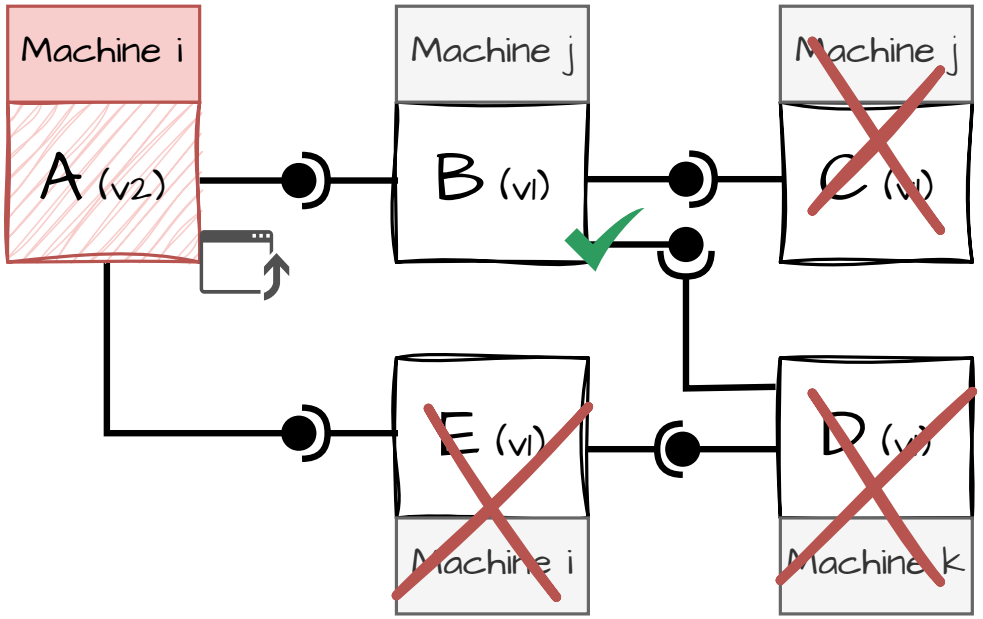
Version 1



delete(D); delete(C); delete(E); delete(B)

4.2 Concrete examples

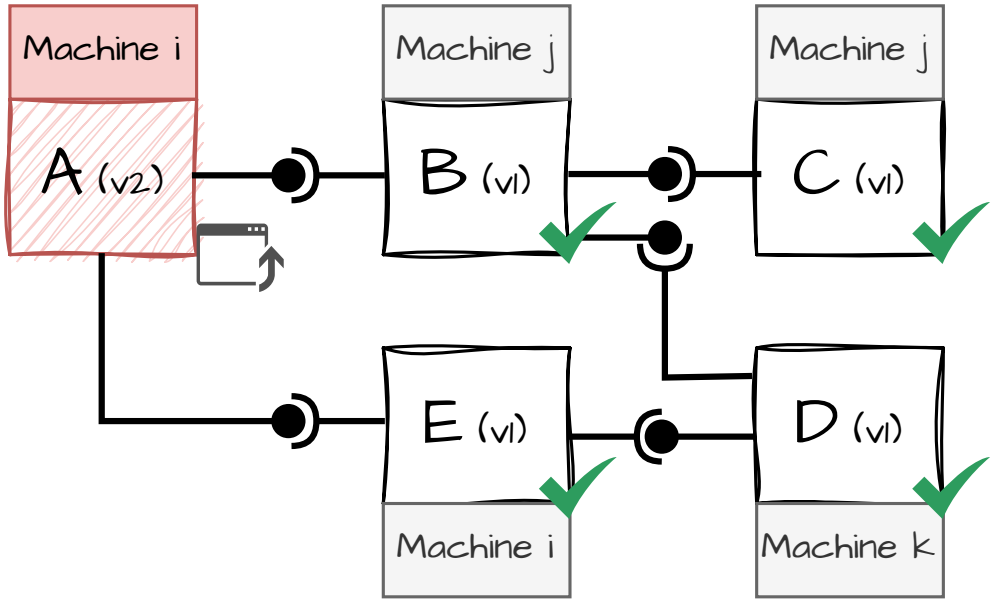
Version 1



delete(D); delete(C); delete(E); delete(B); update(A); start(B)

4.2 Concrete examples

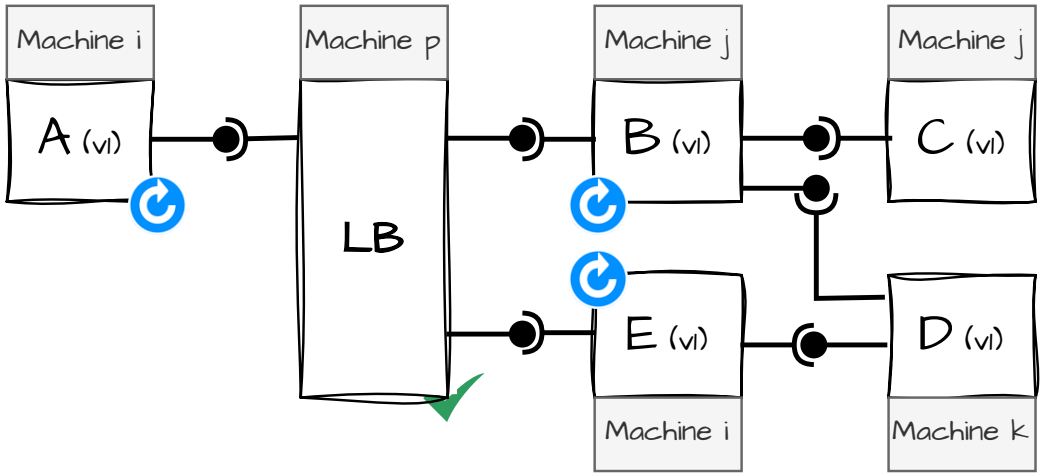
Version 1



delete(D); delete(C); delete(E); delete(B); update(A); start(B); start(E), start(C); start(D)

4.3 Concrete examples

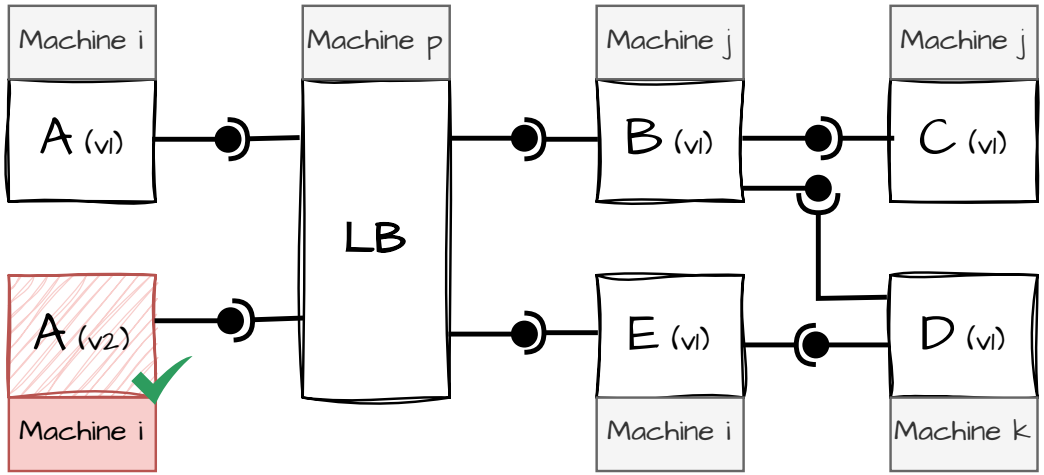
Version 2



start(LB); restart(B); restart(A); restart(E)

4.3 Concrete examples

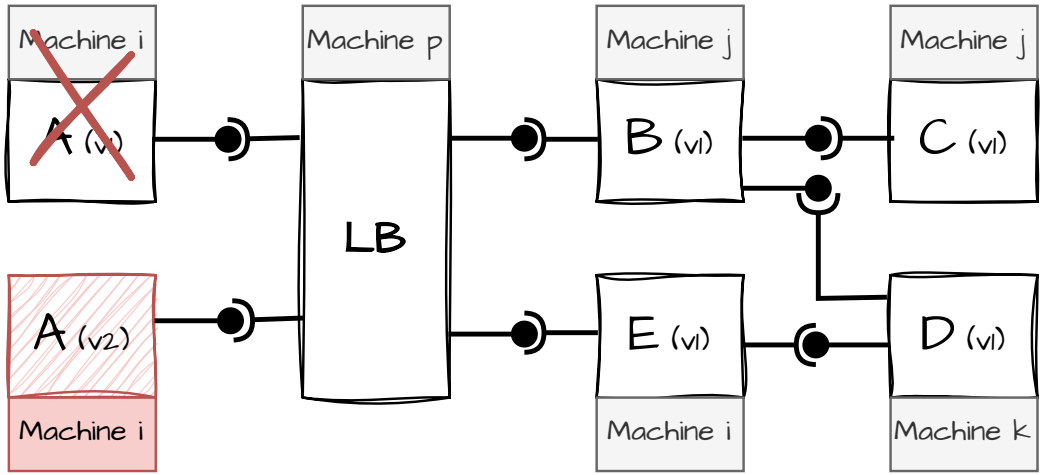
Version 2



start(LB); restart(B); restart(A(v1)); restart(E); start(A(v2))

4.3 Concrete examples

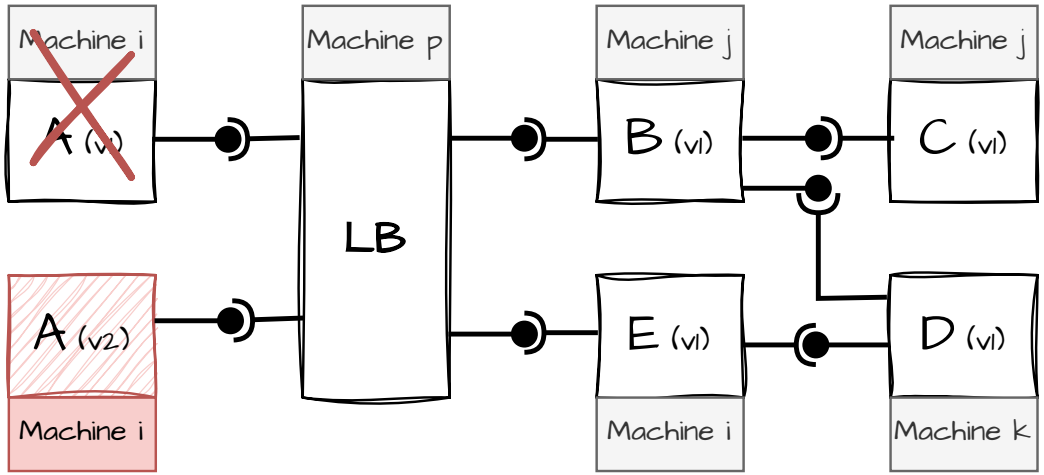
Version 2



start(LB); restart(B); restart(A(v1)); restart(E); start(A(v2)); delete(A(v1))

4.3 Concrete examples

Version 2



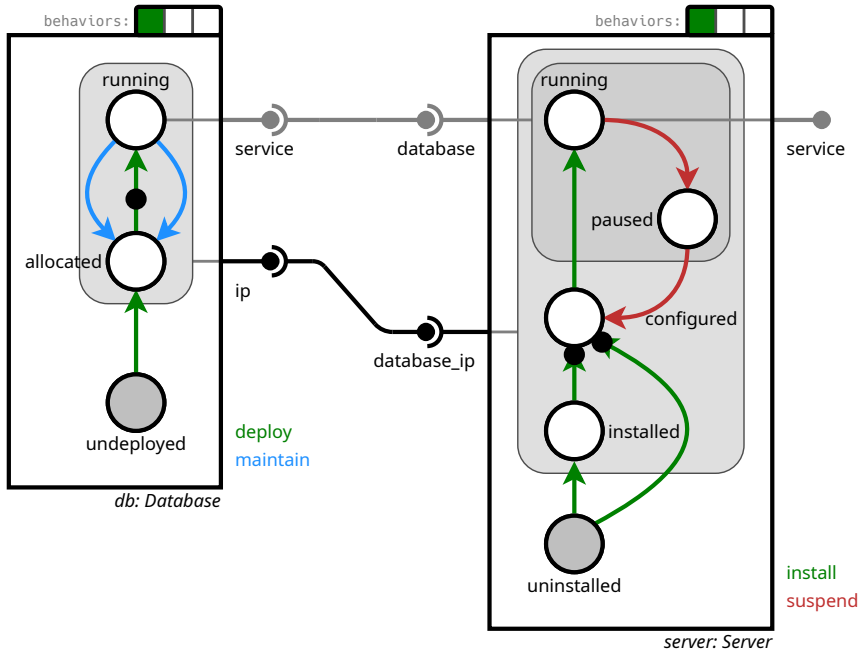
start(LB); restart(B); restart(A(v1)); restart(E); start(A(v2)); delete(A(v1))

⇨ ***Much more complex lifecycle actions than version 1***

4.4 Programmable lifecycle

Safe programmable lifecycle management with Concerto [3]

```python
add(server: Server)
add(db: Database)
con(server.database_ip,db.ip)
con(server.database,db.service)
pushB(server,install)
pushB(db,deploy)
wait(server)
```



- Automatic and safe coordination of programmable lifecycles
- Reconfiguration language with 6 generic instructions

4.5 Infrastructure-as-Code

Infrastructure-as-Code (IaC)

- Tools to automate and manage infrastructures through codes
  - Sharable, testable, versionable, etc.
- IaC embeds reconfiguration, adaptation, and lifecycle management

How the fluctuation paradigm challenges reconfigurations and IaC?

- Reconfiguration and IaC is a way to reach a **new stable state**
  - **Safety** is required to be sure of this state and its reachability
- To reach the new state the **reconfiguration tool has to be resilient**
- If facing urgency the **reconfiguration should respect a deadline**
- etc.

5. Safe reconfigurations

5.1 Code synthesis (plan)

Safe Concerto code generation [4]

- **Inputs**: current state of the system (M) + reconfiguration goals (A)
  - set of behaviors to execute on designated components
  - constraints on the final state of ports

5.1 Code synthesis (plan)

Safe Concerto code generation [4]

- **Inputs**: current state of the system (M) + reconfiguration goals (A)
  - set of behaviors to execute on designated components
  - constraints on the final state of ports
- **Output**: a Concerto reconfiguration program
  - pushB requests
  - wait commands
  - usual to handle create/delete con/dcon before and after behaviors

5.1 Code synthesis (plan)

Safe Concerto code generation [4]

- **Inputs**: current state of the system (M) + reconfiguration goals (A)
  - set of behaviors to execute on designated components
  - constraints on the final state of ports
- **Output**: a Concerto reconfiguration program
  - pushB requests
  - wait commands
  - usual to handle create/delete con/dcon before and after behaviors

Problem formulation

Find a valid (optimal) schedule of pushB and wait instructions (in Z3)

5.2 Safe IaC

Lifecycle coordination in IaC

- **CoAnsible**: Ansible extension for clear composition of roles
- ***CoTerraform**: replace too restrictive static CRUD lifecycles*
  - *Ph.D. of Simon Artus*

5.2 Safe IaC

Lifecycle coordination in IaC

- **CoAnsible**: Ansible extension for clear composition of roles
- ***CoTerraform**: replace too restrictive static CRUD lifecycles*
  - *Ph.D. of Simon Artus*

Formal IaC

- **BPlan** Terraform: augmented plan with model-checking (Maude)
- **For-CoaLa ANR project**: formalization and proofs on Ansible [5]
  - *Ph.D. of Olivia Proust*

6. Resilient reconfigurations

6.1 Centralized reconfigurations

Either in academic reconfiguration or IaC

⇨ **solutions are centralized**

6.1 Centralized reconfigurations

Either in academic reconfiguration or IaC

⇨ **solutions are centralized**

- Central global state
- Central analysis/planning computation

Associated fragilities

- **Failures/disconnections**: Inconsistent state, state drifts
- **Failures/disconnections**: No local progress is possible
- **Privacy**: no shared state in cross-DevOps organization

6.1 Centralized reconfigurations

Either in academic reconfiguration or IaC

⇨ **solutions are centralized**

- Central global state
- Central analysis/planning computation

Associated fragilities

- **Failures/disconnections**: Inconsistent state, state drifts
- **Failures/disconnections**: No local progress is possible
- **Privacy**: no shared state in cross-DevOps organization

Exceptions:

- *Kubernetes (distributed ETCD database, individual controllers)*
- *Mjuz (decentralized Pulumi) [6]*

6.2 Ballet

Decentralized reconfigurations with programmable lifecycles

Decentralized reconfigurations with programmable lifecycles

- **Concerto-D** [7], [8]
  - A formalized collaborative Concerto (Maude)
  - Synchronizations: disconnections, ports and baheviors statuses

6.2 Ballet

Decentralized reconfigurations with programmable lifecycles

- **Concerto-D** [7], [8]
  - A formalized collaborative Concerto (Maude)
  - Synchronizations: disconnections, ports and baheviors statuses
- **Decentralized planning** [7]
  - Local CP (constraint programming) model
  - Gossip constraints propagation for models enrichment
  - Final consensus to end the collaborative process

6.2 Ballet

Decentralized reconfigurations with programmable lifecycles

- **Concerto-D** [7], [8]
  - A formalized collaborative Concerto (Maude)
  - Synchronizations: disconnections, ports and baheviors statuses
- **Decentralized planning** [7]
  - Local CP (constraint programming) model
  - Gossip constraints propagation for models enrichment
  - Final consensus to end the collaborative process

***Future work**: Decentralized and local-first IaC*

6.3 Concurrent reconfigurations

Frequent fluctuations

- No stable/quiescent state
- Concurrency regime of reconfigurations

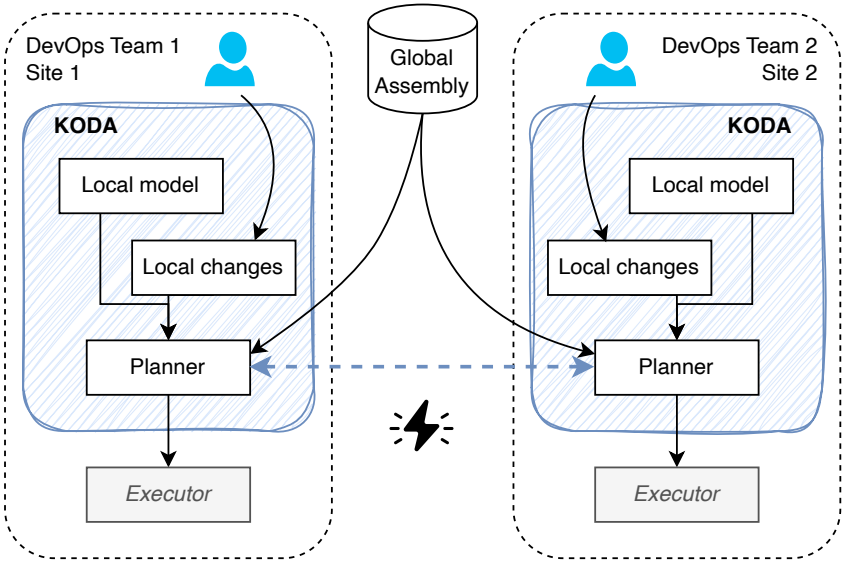
6.3 Concurrent reconfigurations

Frequent fluctuations

- No stable/quiescent state
- Concurrency regime of reconfigurations

Conflict detection with Koda

- Ballet = reconfigurations one by one
- Koda = concurrent reconfigurations
  - detects conflicts
  - finds the causality of conflicts

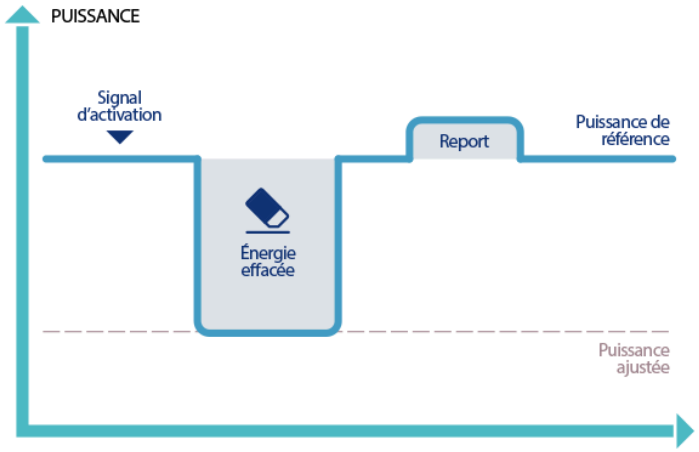


7. Timely constrained reconfigurations

7.1 Context

Deadline to reconfigure

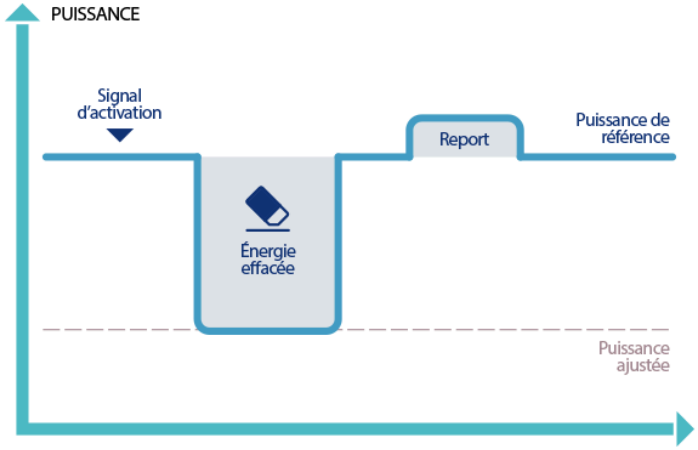
- PEPR Cloud + PEPR TASE
  - Collaboration with RTE
  - Collaboration with Ctrl-A
  - *Ph.D. of Nathan Rabier*



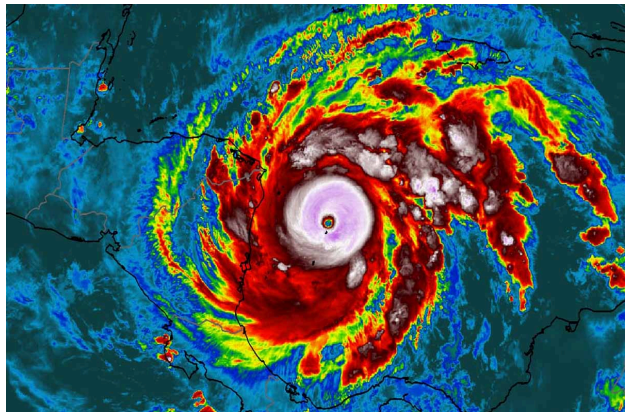
7.1 Context

Deadline to reconfigure

- PEPR Cloud + PEPR TASE
  - Collaboration with RTE
  - Collaboration with Ctrl-A
  - *Ph.D. of Nathan Rabier*



- Urgency in reconfigurations
  - Urgent computing



7.2 Challenge

⇨ **Strong deadline to adapt (MAPE)**

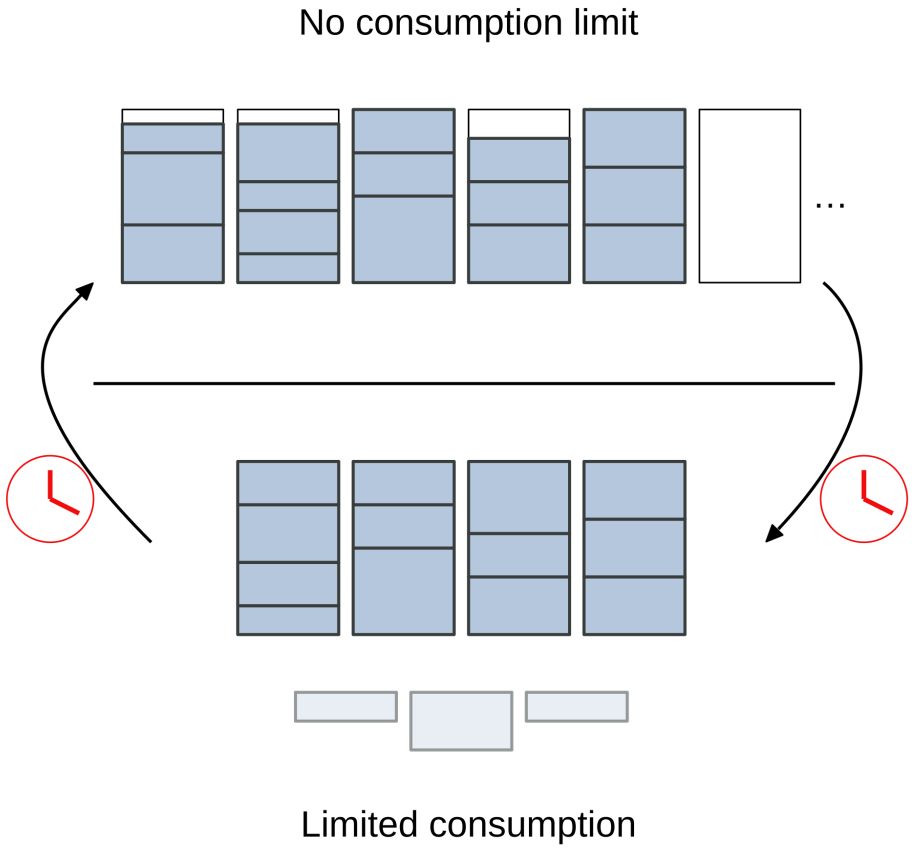
7.2 Challenge

⇨ Strong deadline to adapt (MAPE)

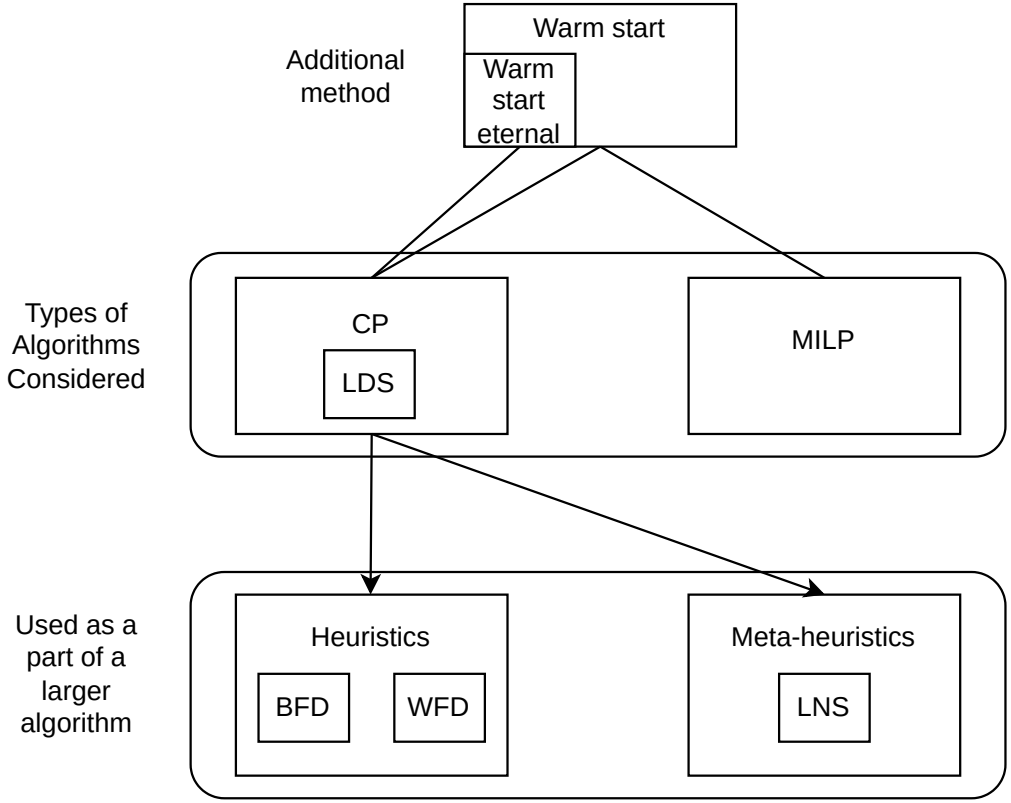
Le réseau de transport d'électricité

Switching between two configurations

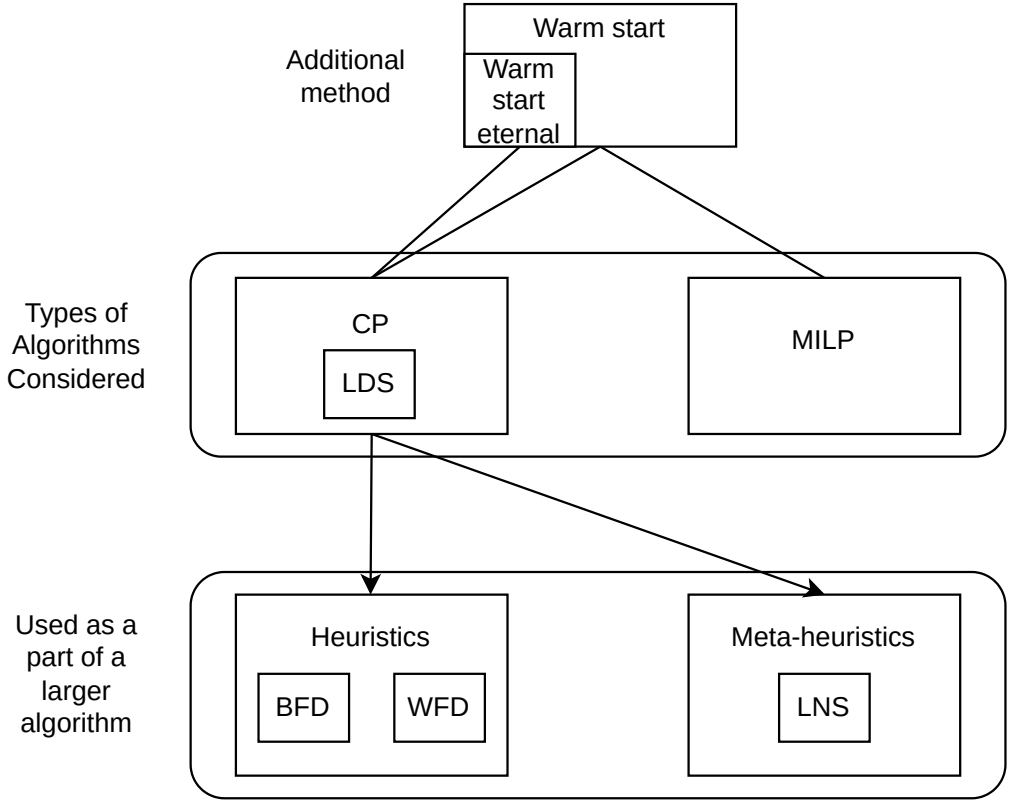
- Unlimited configuration
  - BinPacking problem
- Limited configuration
  - Multi-Knapsack problem



7.3 Work in progress

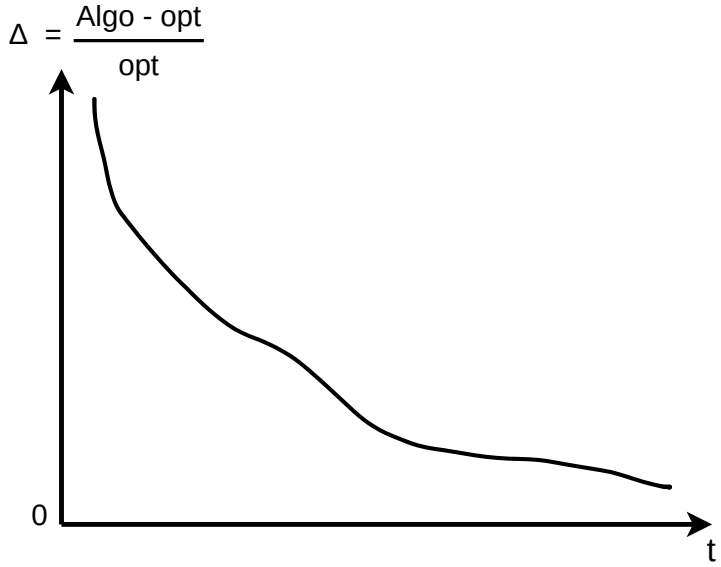


7.3 Work in progress



Metrics of interest:

- Anytime property
- Distance to the initial configuration



8. Sobriety, commons, and resilience

8.1 Context

Long term permanent resource unavailabilities

⇨ Need for alternatives to Cloud computing

8.1 Context

Long term permanent resource unavailabilities

⇨ Need for alternatives to Cloud computing

Collaboration with Deuxfleurs

- PEPR eNSEMBLE + NumEco
  - *Ph.D. of Lucien Astié*
- Digital commons (resilience)
- Limits-by-design (sobriety)

8.1 Context

Long term permanent resource unavailabilities

⇨ Need for alternatives to Cloud computing

Collaboration with Deuxfleurs

- PEPR eNSEMBLE + NumEco
  - *Ph.D. of Lucien Astié*
- Digital commons (resilience)
- Limits-by-design (sobriety)

Research question:

Sobriety ⇨ resilience (commons)?

Resilience (commons) ⇨ sobriety?

9. Conclusion

9.1 Concluding message

From resource infinity to ressources fluctuations

- New technical solutions to reconfigure, adapt, survive, recover...
  - Safety of reconfigurations
  - Resilience of reconfigurations
  - Timely constrained reconfigurations

... yes, but we also need to imagine and study completely new paradigms!

- Lowtech, sustainability, limits, etc.

9.2 People involved in ongoing work

- **Simon Artus**, Ph.D. student Orange, Melan [CoTerraform]
- **Lucien Astié**, Ph.D. student IMT Atlantique, Nantes [PEPR eNSEMBLE]
- Samuel Berlemont, researcher Orange, Melan [CoTerraform]
- Erwan Brousse, Associate prof. Univ. Nantes [PEPR eNSEMBLE & NumEco]
- Sophie Cerf, CR Inria, Grenoble [PEPR Cloud & TASE]
- Baptiste Jonglez, Engineer Inria, Nantes [PEPR eNSEMBLE & NumEco, CoAnsible]
- **Sidi Mohammed Kaddour**, Engineer Inria, Nantes [CoAnsible]
- Théo Le Calvar, Associate prof. IMT Atlantique, Nantes [PEPR eNSEMBLE & NumEco]
- Frédéric Loulergue, Professor Univ. Orléans [PEPR Cloud, Ballet, BPlan, ANR For-CoaLa]
- **Clément Mommessin**, Postdoc Inria, Grenoble [PEPR Cloud & TASE]
- **Eloi Perdereau**, Postdoc Univ. Orléans [BPlan, ANR For-CoaLa]
- Jolan Philippe, Associate prof., Univ. Orléans [Ballet, Koda, BPlan, ANR For-CoaLa]
- **Olivia Proust**, Ph.D. student, IMT Atlantique [ANR For-CoaLa]
- Charles Prud'homme, Associate prof., IMT Atlantique, Nantes [Ballet, Koda]
- **Nathan Rabier**, Ph.D. student Inria, Grenoble [PEPR Cloud & TASE]
- Eric Rutten, CR Inria, Grenoble [PEPR Cloud & TASE]
- Guido Salvaneschi, Professor St Gallen University, Switzerland [BPlan]
- Daniel Sokolowski, Researcher AWS, New York [BPlan]

Bibliography

[1] K. Veeraraghavan et al., "Maelstrom: Mitigating Datacenter-level Disasters by Draining Interdependent Traffic Safely and Efficiently," in 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), Carlsbad, CA: USENIX Association, Oct. 2018.

[2] H. Coullon, L. Henrio, F. Loulergue, and S. Robillard, "Component-based Distributed Software Reconfiguration:A Verification-oriented Survey," ACM Comput. Surv., vol. 56, no. 1, Aug. 2023, doi: 10.1145/3595376.

[3] M. Chardet, H. Coullon, and S. Robillard, "Toward safe and efficient reconfiguration with Concerto," Science of Computer Programming, vol. 203, p. 102582, 2021, doi: https://doi.org/10.1016/j.scico.2020.102582.

[4] S. Robillard and H. Coullon, "SMT-Based Planning Synthesis for Distributed System Reconfigurations," in Fundamental Approaches to Software Engineering, E. B. Johnsen and M. Wimmer, Eds., Cham: Springer International Publishing, 2022, pp. 268–287.

[5] [Online]. Available: https://for-coala.github.io/

[6] D. Sokolowski, "Deployment coordination for cross-functional DevOps teams," in Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, in ESEC/FSE 2021. Athens, Greece: Association for Computing Machinery, 2021. doi: 10.1145/3468264.3473101.

[7] J. Philippe, A. Omond, H. Coullon, C. Prud'Homme, and I. Raïs, "Fast Choreography of Cross-DevOps Reconfiguration with Ballet: A Multi-Site OpenStack Case Study," in 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 2024. doi: 10.1109/SANER60148.2024.00007.

[8] A. Omond, H. Coullon, I. Rais, and O. Anshus, "Leveraging Relay Nodes to Deploy and Update Services in a CPS with Sleeping Nodes," in 2023 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), 2023, pp. 532–539. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics60724.2023.00102.