Continuous Management of Multi-Service Applications over the Cloud-IoT Continuum

Giuseppe Bisicchia

Talk for the Information Security Group @ ETH Zurich

Who I Am

- I was born in 1998 in Catania, Sicily (Italy)
- During high school, I took part several competitions concerning also computer science and robotics
- In 2020, I received a BSc degree cum laude in Computer Science from the University of Pisa
- In 2022, received a MSc degree cum laude in Computer Science (ICT Solutions Architect) from the University of Pisa and a MSc degree (9.88/10) in Computer Engineering (Cybersecurity) from the University of Malaga after pursuing a Double Degree Program and living in Spain

Who I Am

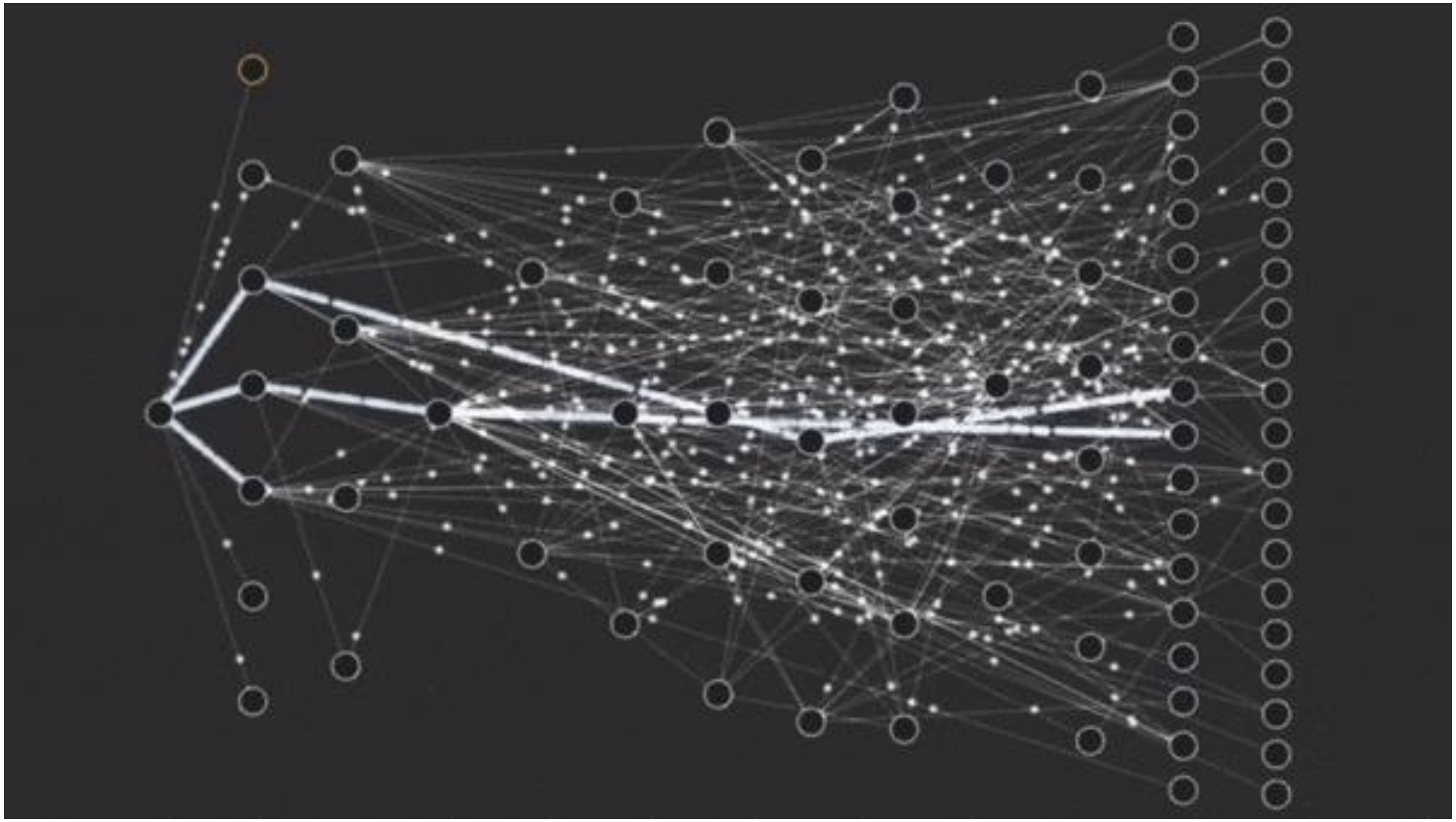
- In 2021, I won 1-year research grant (and in 2022 a 1-year extension) from the GARR Consortium for issues related to the development of innovative digital infrastructures and services
- I presented two papers in a national (*CILC*) and international (*SMARTCOMP)* conferences
- I published an article in the international *Journal of Logic and Computation*
- I am a mentor for the Pisa CoderDojo and I organised several educational workshop, I was first a journalist intern and then I worked as editor for a scientific dissemination site

Continuous Management of Multi-Service Applications over the Cloud-IoT Continuum

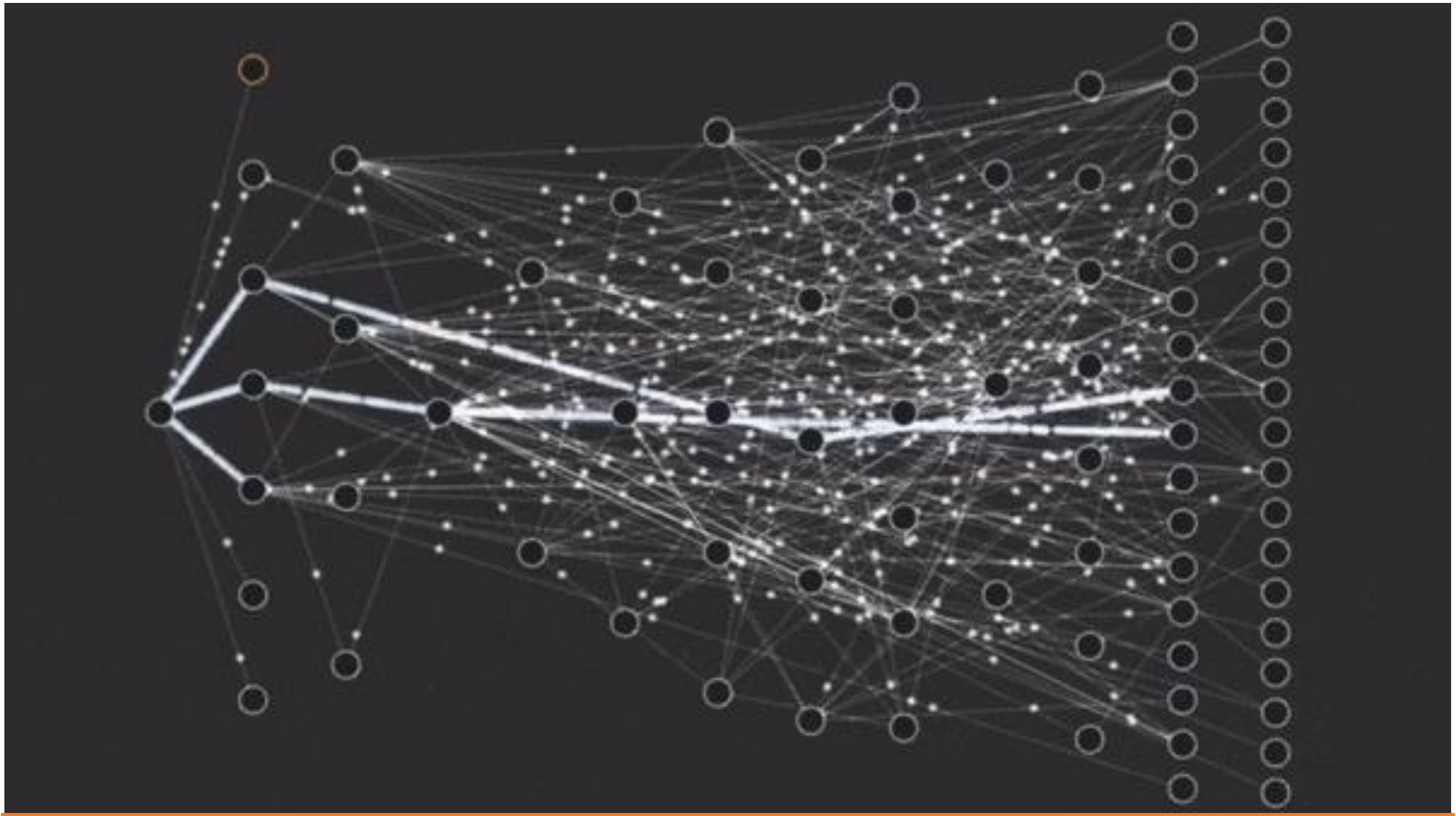
Giuseppe Bisicchia

Talk ***(now for real)*** *for the Information Security Group @ ETH Zurich*

Context: Multi-Service Applications



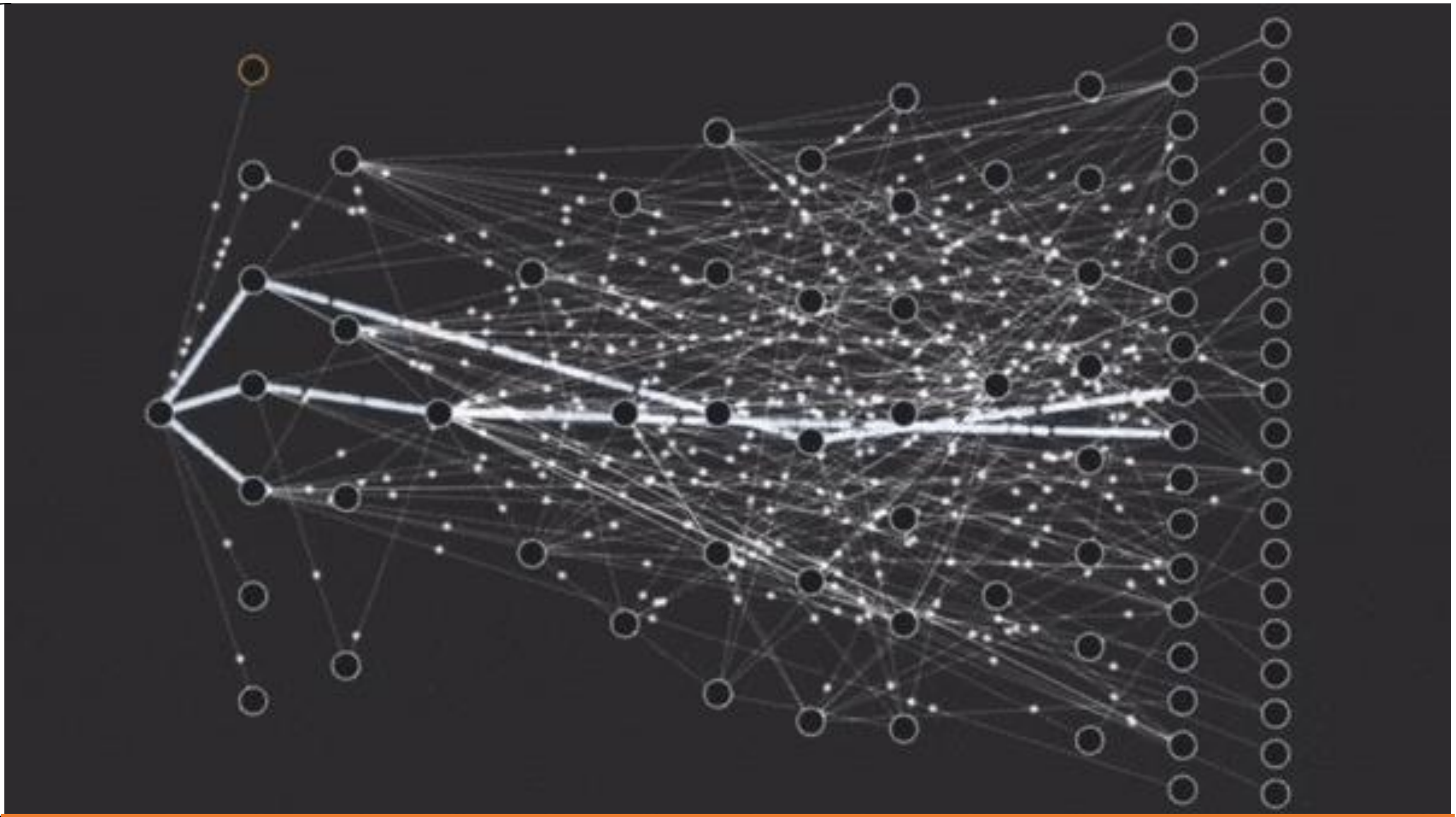
Context: Multi-Service Applications



100+ interacting (micro)services

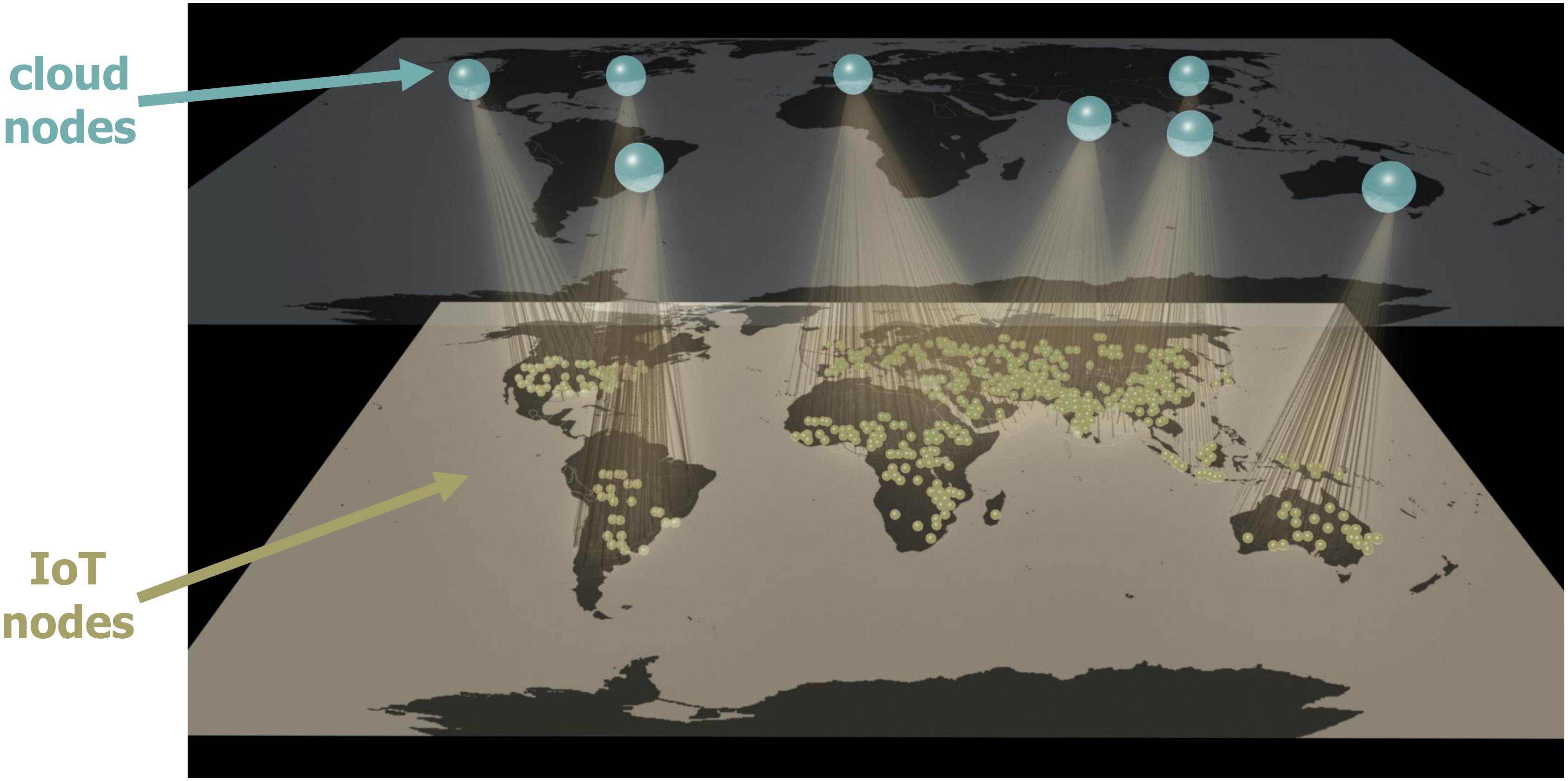
Context: Multi-Service Applications

strictly hardware and **software requirements, QoS ...**

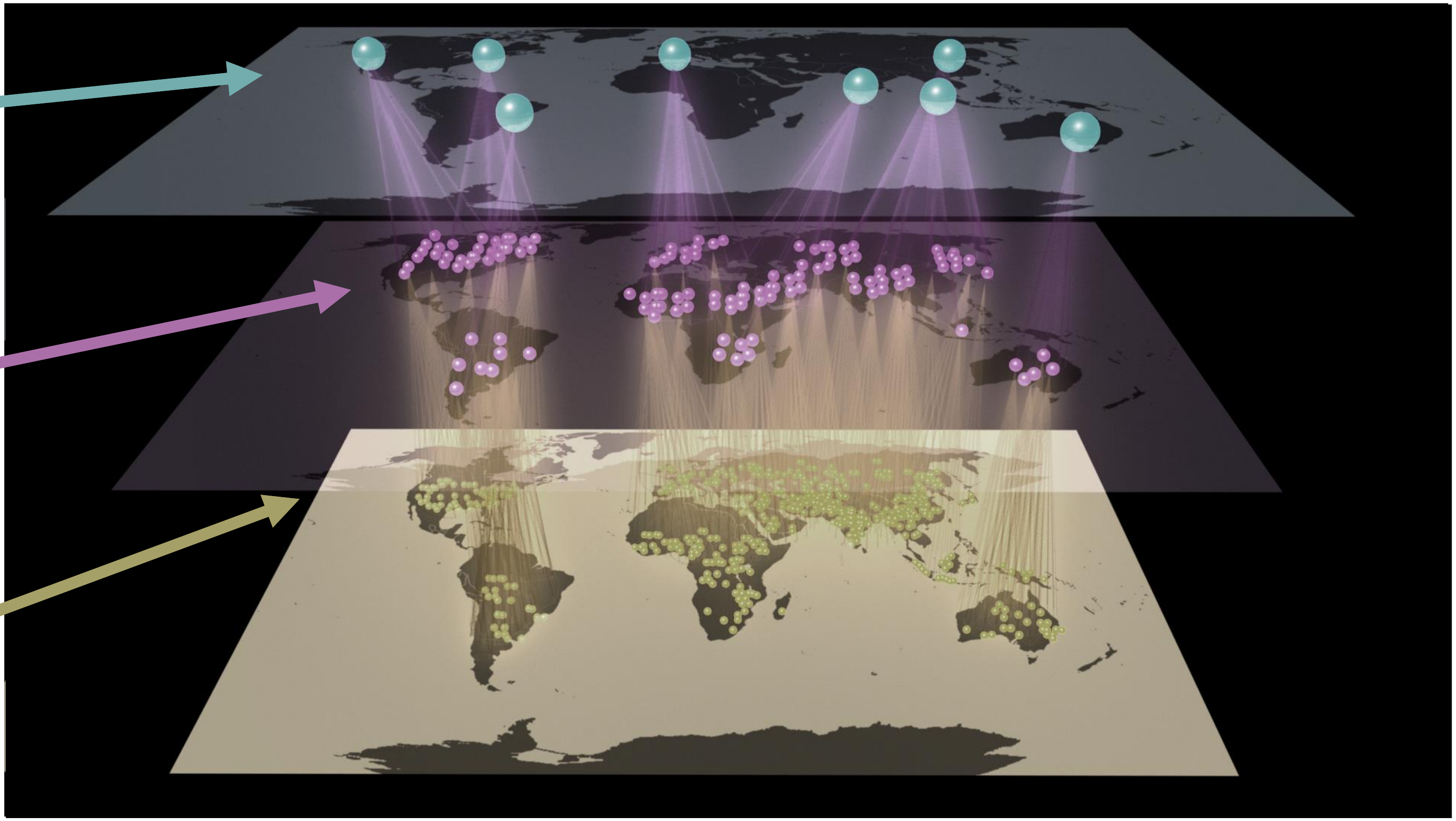


100+ interacting (micro)services

Context: Cloud-IoT Infrastructures



Context: Cloud-IoT Infrastructures

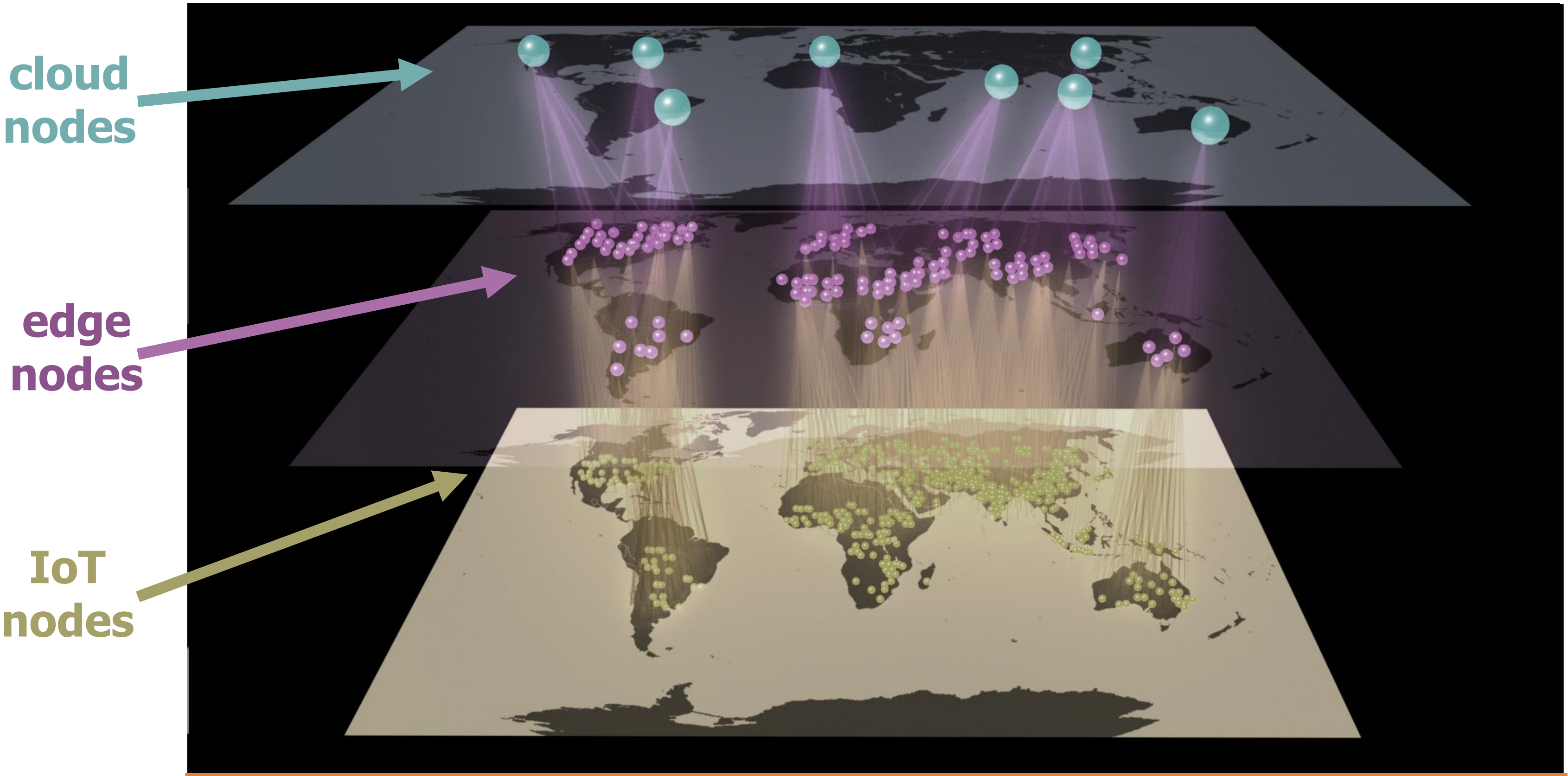


cloud nodes

edge nodes

IoT nodes

Context: Cloud-IoT Infrastructures



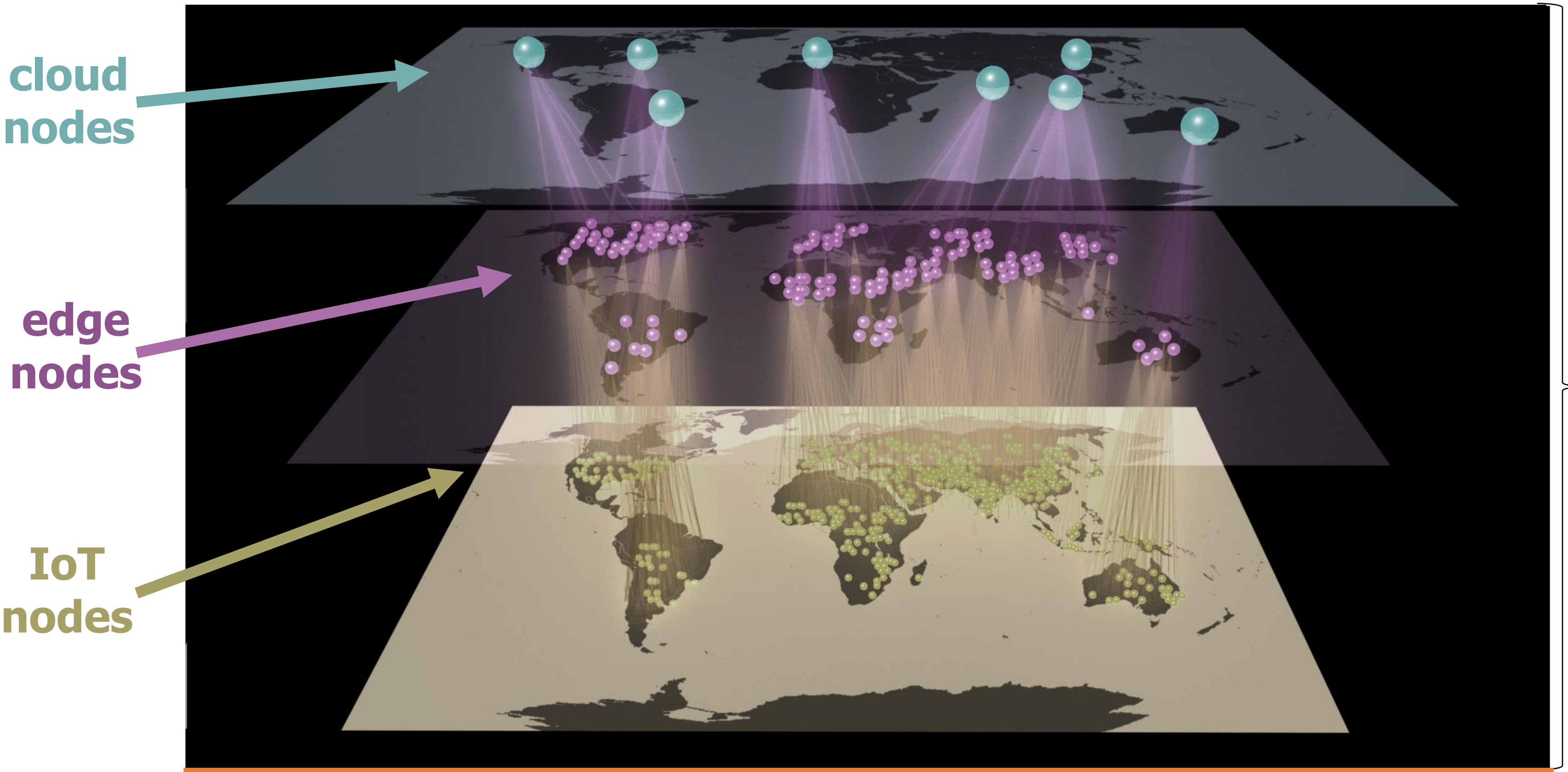
cloud nodes

edge nodes

IoT nodes

1000+ highly pervasive, distributed and heterogeneous nodes

Context: Cloud-IoT Infrastructures



cloud nodes

edge nodes

IoT nodes

node **failures** or **overloading**, traffic **congestion**, **disconnections**

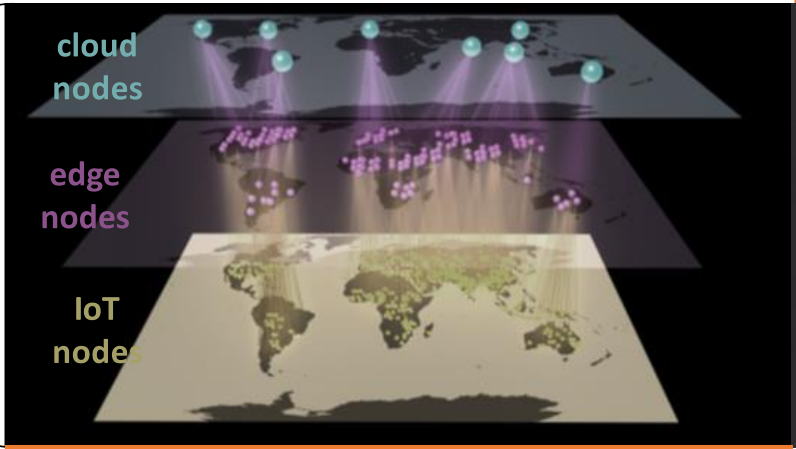
1000+ highly pervasive, distributed and heterogeneous nodes

Research Problem

Cloud-IoT Infrastructures

100+ interacting (micro)services

node **failures** or **overloading**, traffic **congestion**, **disconnections**



strictly hardware and **software requirements, QoS ...**

1000+ highly pervasive, distributed and heterogeneous nodes

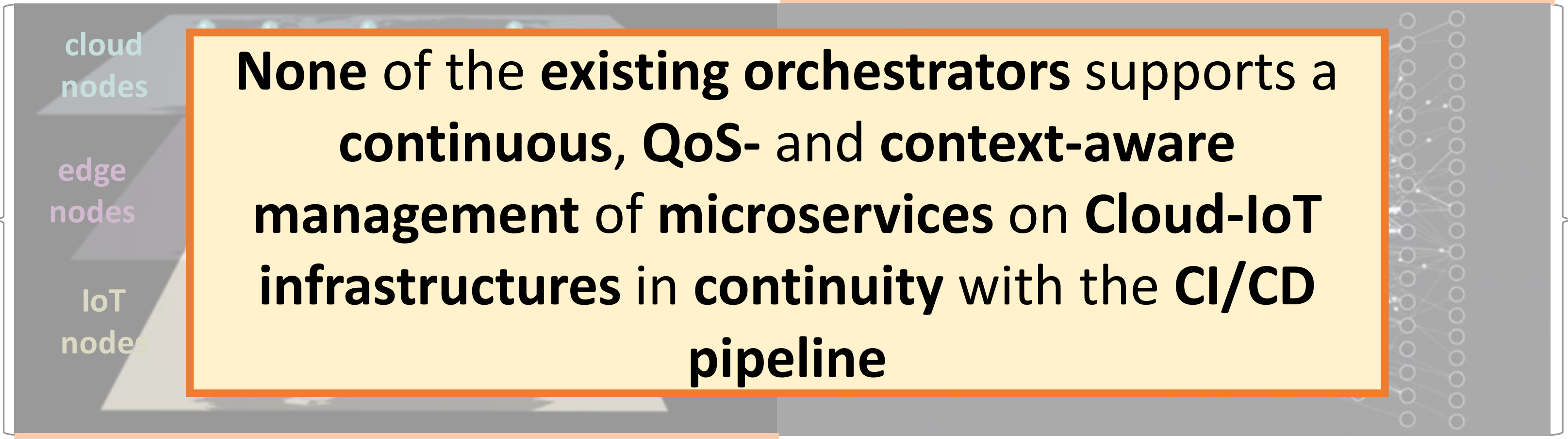
Multi-Service Applications

Research Problem

Cloud-IoT Infrastructures

100+ interacting (micro)services

node **failures** or **overloading**, traffic **congestion**, **disconnections**



None of the **existing orchestrators** supports a **continuous**, **QoS-** and **context-aware management** of **microservices** on **Cloud-IoT infrastructures** in **continuity** with the **CI/CD pipeline**

strictly hardware and **software requirements**, **QoS** ...

1000+ highly pervasive, distributed and heterogeneous nodes

Multi-Service Applications

Design and develop a next-gen orchestrator for a continuous, QoS-compliant management of multi-service applications on Cloud-IoT infrastructures

Continuous Reasoning

continuous and incremental
formal analysis

Continuous Reasoning

continuous and incremental **formal analysis**

focussing on the latest changes

Continuous Reasoning

continuous and incremental **formal analysis**

focussing on the latest changes

reuse previously **computed results**

Continuous Reasoning

continuous and incremental **formal analysis**

focussing on the latest changes

reuse previously **computed results**

FogBrainX is the core of a **Continuous Reasoning** engine for making **informed management decisions** for **multi-service applications** on **Cloud-IoT infrastructures**

Stefano Forti, Giuseppe Bisicchia, and Antonio Brogi. Declarative Continuous Reasoning in the Cloud-IoT Continuum. Journal of Logic and Computation, 2022.

FogBrainX

FogBrainX

declarative

as it is Prolog code: **more concise, easier to understand and maintain w.r.t** existing **procedural solutions**

FogBrainX

explainable

as **it derives proofs** by relying on Prolog and **can explain *why* a** certain management **decision was taken** at runtime

declarative

as it is Prolog code: **more concise, easier to understand and maintain w.r.t** existing **procedural solutions**

FogBrainX

explainable

as **it derives proofs** by relying on Prolog and **can explain *why* a** certain management **decision was taken** at runtime

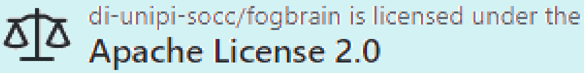
declarative

as it is Prolog code: **more concise, easier to understand and maintain w.r.t** existing **procedural solutions**

scalable

as it exploits continuous reasoning to **reduce the size of the problem instance only to** those application **services in need for attention**

open-source



di-unipi-socc/fogbrain is licensed under the Apache License 2.0

Available at:

https://github.com/di-unipi-socc/fogbrainx

explainable

as **it derives proofs** by relying on Prolog and **can explain *why* a** certain management **decision was taken** at runtime

FogBrainX

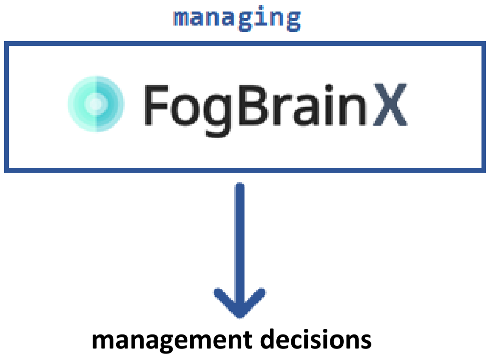
declarative

as it is Prolog code: **more concise, easier to understand and maintain w.r.t** existing **procedural solutions**

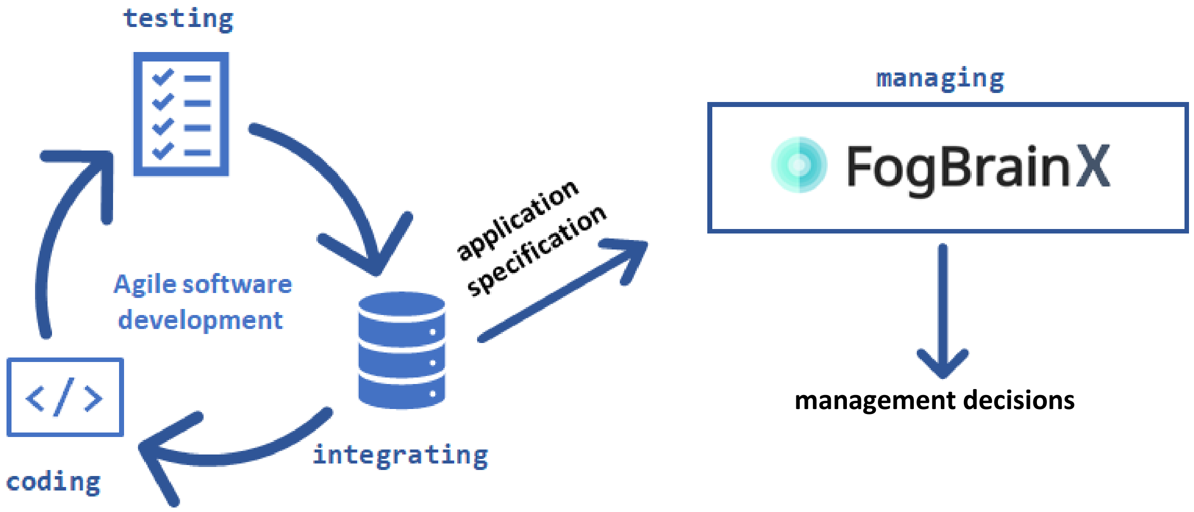
scalable

as it exploits continuous reasoning to **reduce the size of the problem instance only to** those application **services in need for attention**

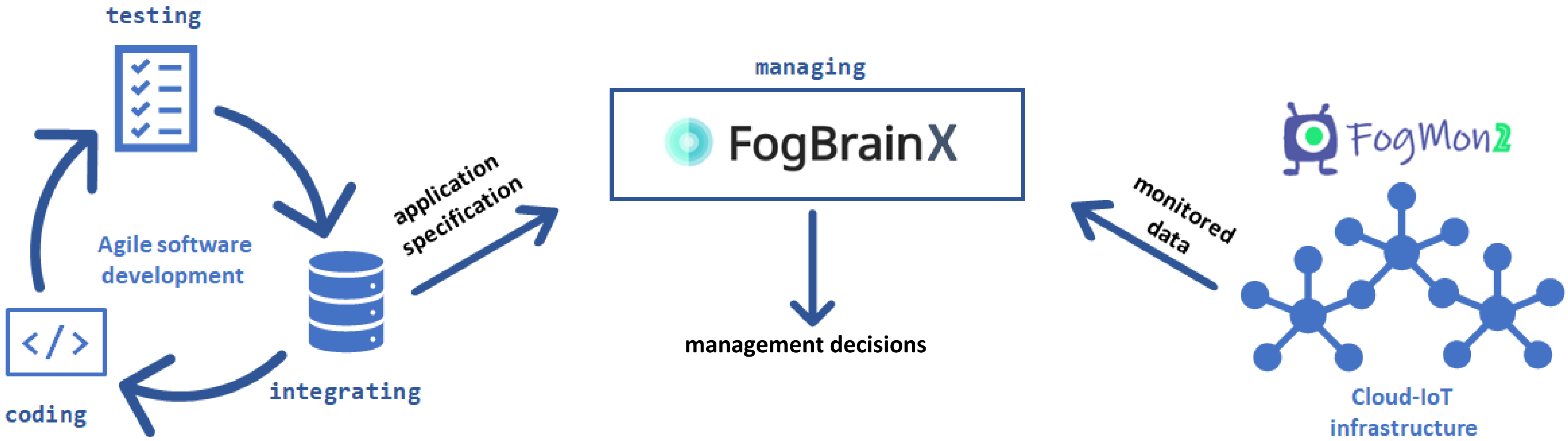
The Orchestrator



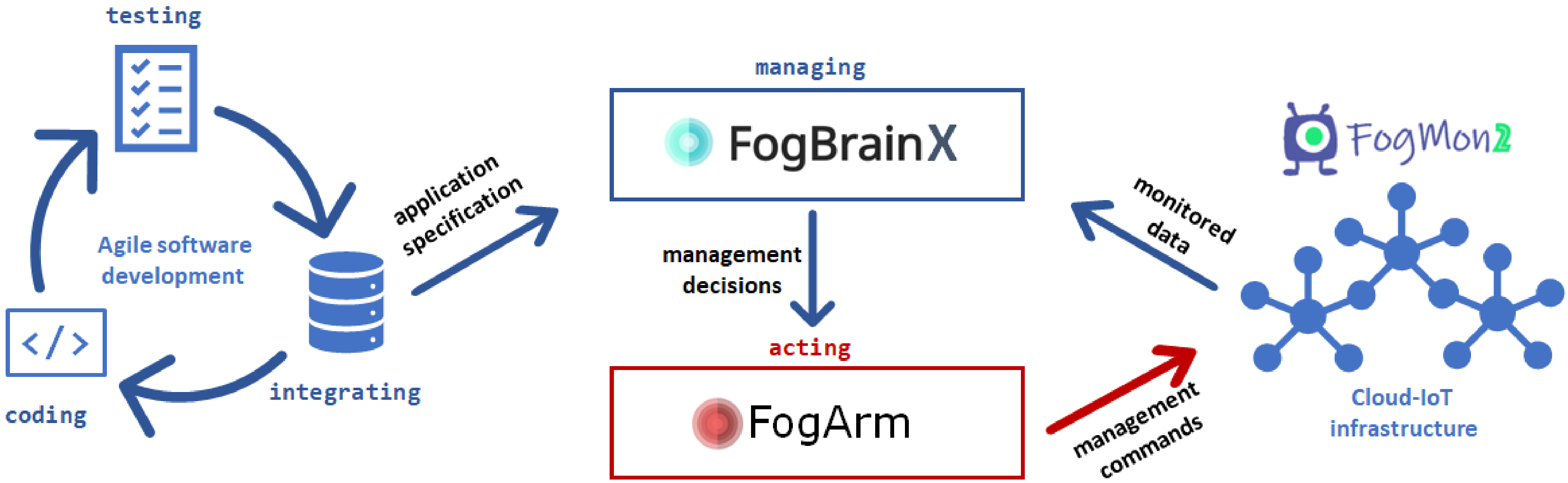
The Orchestrator



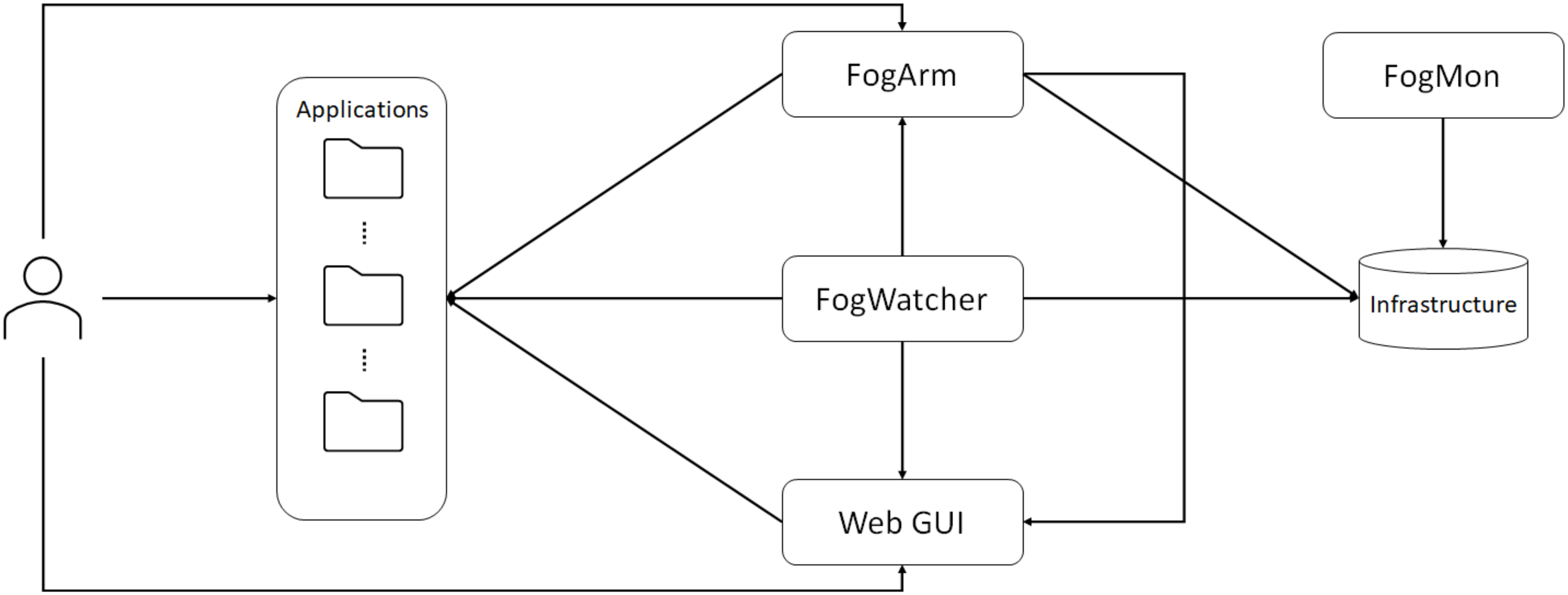
The Orchestrator



The Orchestrator

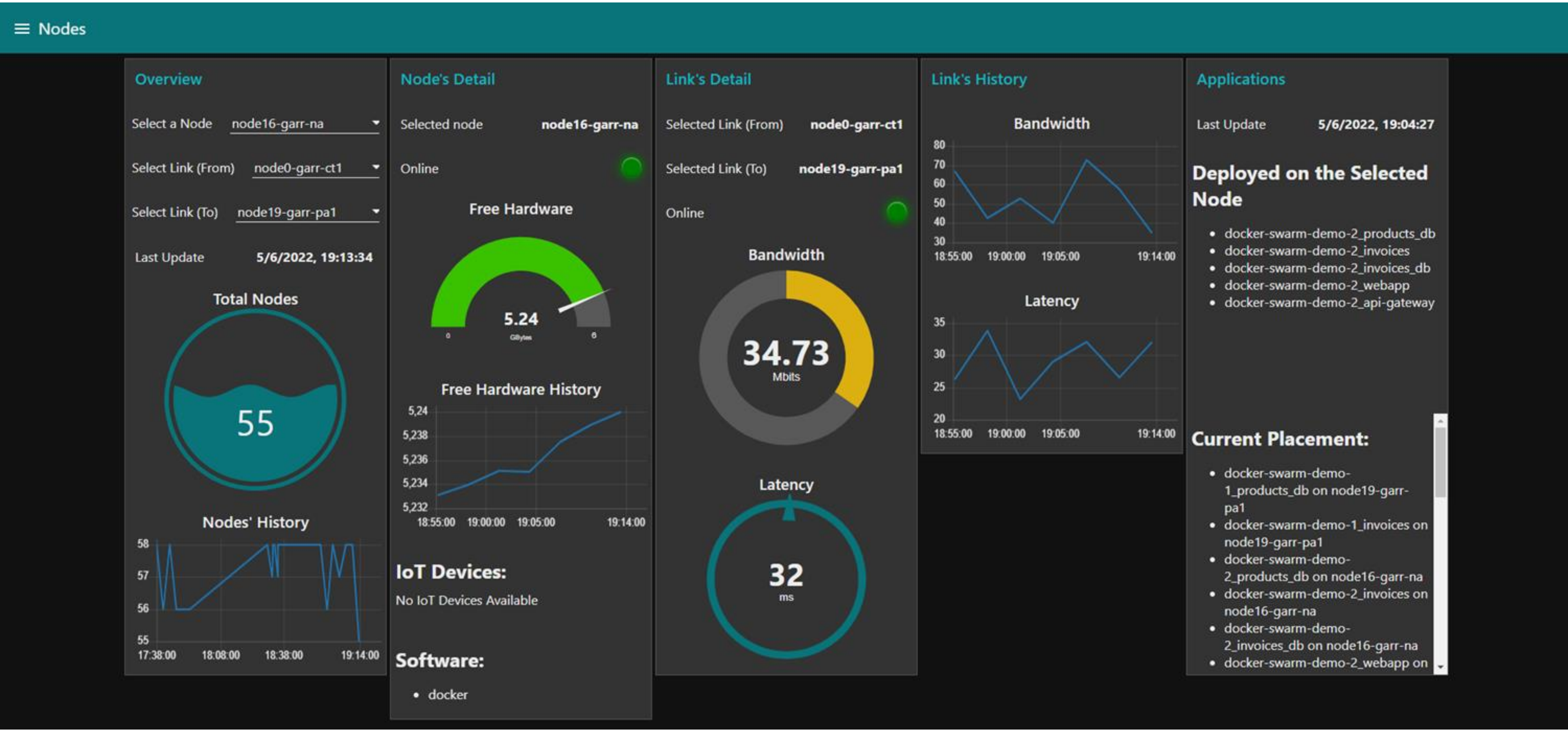


FogArm



Applications
FogArm
FogMon
FogWatcher
Infrastructure
Web GUI

FogArm's WebGUI: Nodes



Overview

Select a Node: node16-garr-na

Select Link (From): node0-garr-ct1

Select Link (To): node19-garr-pa1

Last Update: 5/6/2022, 19:13:34

Node's Detail

Selected node: node16-garr-na

Online ●

IoT Devices:

No IoT Devices Available

Software:

- docker

Link's Detail

Selected Link (From): node0-garr-ct1

Selected Link (To): node19-garr-pa1

Online ●

Link's History

Applications

Last Update: 5/6/2022, 19:04:27

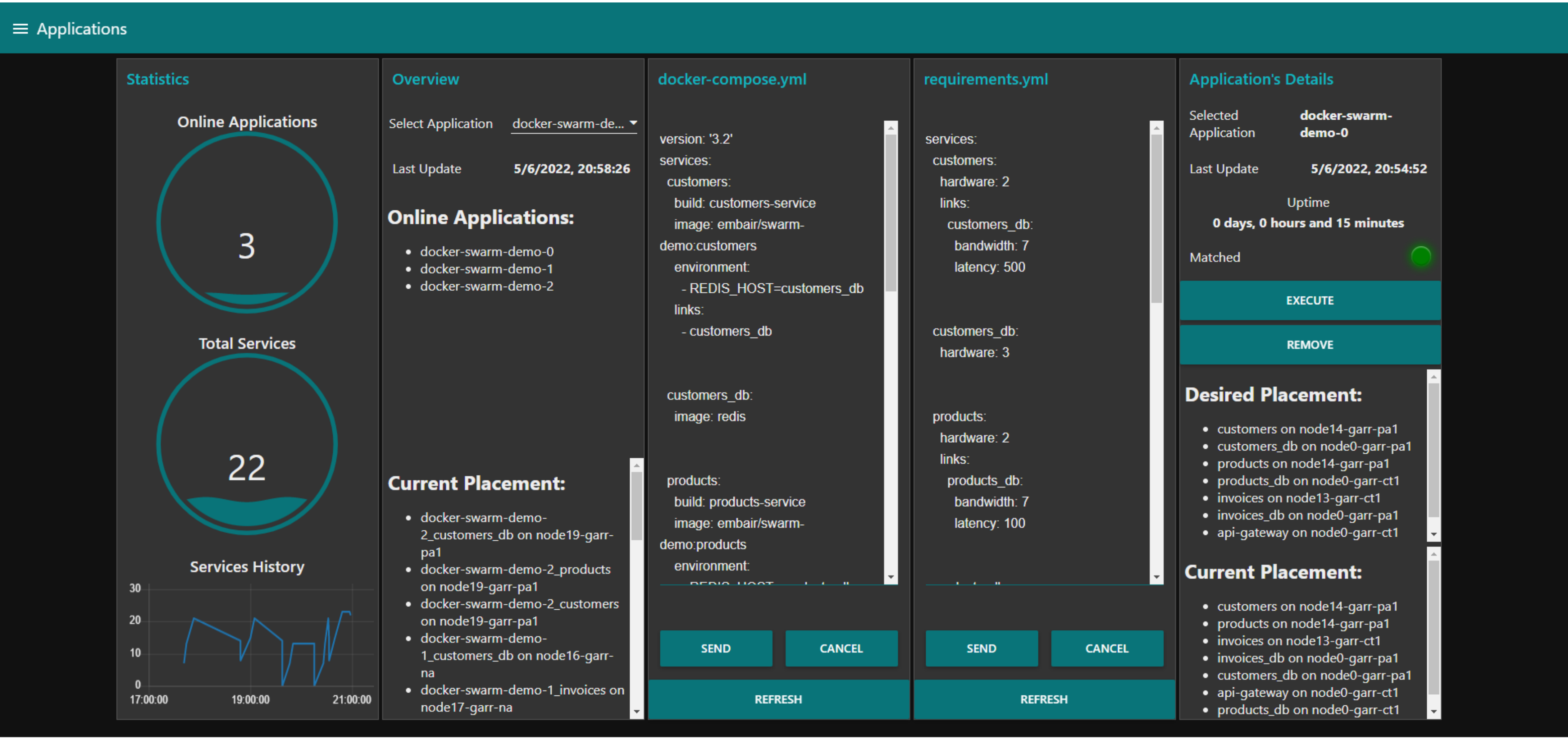
Deployed on the Selected Node

- docker-swarm-demo-2_products_db
- docker-swarm-demo-2_invoices
- docker-swarm-demo-2_invoices_db
- docker-swarm-demo-2_webapp
- docker-swarm-demo-2_api-gateway

Current Placement:

- docker-swarm-demo-1_products_db on node19-garr-pa1
- docker-swarm-demo-1_invoices on node19-garr-pa1
- docker-swarm-demo-2_products_db on node16-garr-na
- docker-swarm-demo-2_invoices on node16-garr-na
- docker-swarm-demo-2_invoices_db on node16-garr-na
- docker-swarm-demo-2_webapp on

FogArm’s WebGUI: Applications



Statistics

Online Applications: 3

Total Services: 22

Services History

Overview

Select Application: docker-swarm-de...

Last Update: **5/6/2022, 20:58:26**

Online Applications:

- docker-swarm-demo-0
- docker-swarm-demo-1
- docker-swarm-demo-2

Current Placement:

- docker-swarm-demo-2_customers_db on node19-garr-pa1
- docker-swarm-demo-2_products on node19-garr-pa1
- docker-swarm-demo-2_customers on node19-garr-pa1
- docker-swarm-demo-1_customers_db on node16-garr-na
- docker-swarm-demo-1_invoices on node17-garr-na

docker-compose.yml

```
version: '3.2'
services:
  customers:
    build: customers-service
    image: embair/swarm-demo:customers
    environment:
      - REDIS_HOST=customers_db
    links:
      - customers_db


  customers_db:
    image: redis


  products:
    build: products-service
    image: embair/swarm-demo:products
    environment:
```

requirements.yml

```
services:
  customers:
    hardware: 2
    links:
      customers_db:
        bandwidth: 7
        latency: 500


  customers_db:
    hardware: 3


  products:
    hardware: 2
    links:
      products_db:
        bandwidth: 7
        latency: 100
```

Application's Details

Selected Application: **docker-swarm-demo-0**

Last Update: **5/6/2022, 20:54:52**

Uptime: **0 days, 0 hours and 15 minutes**

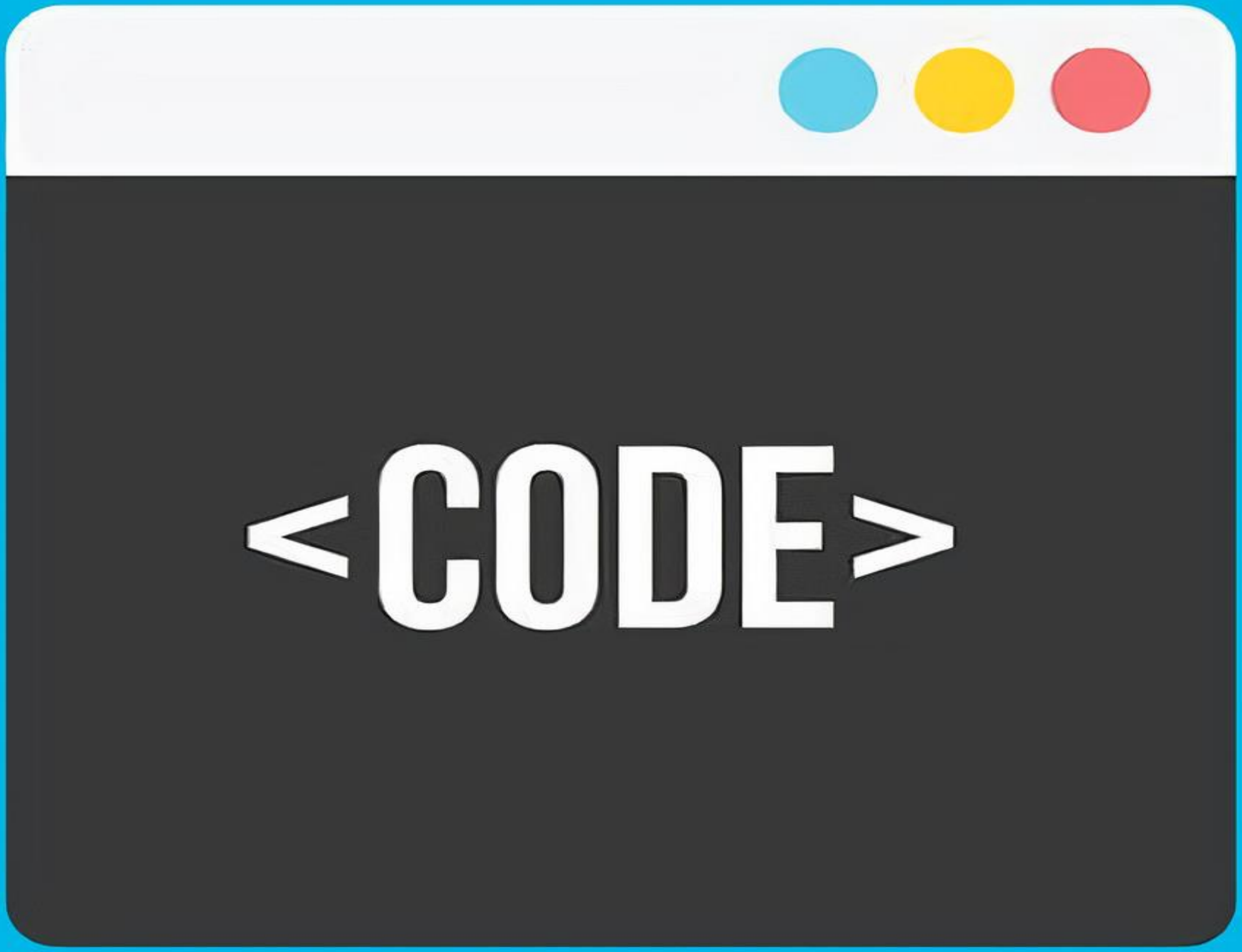
Matched ●

Desired Placement:

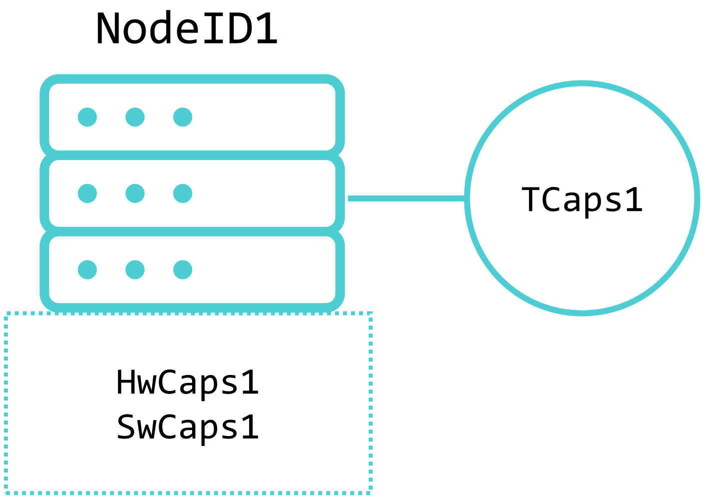
- customers on node14-garr-pa1
- customers_db on node0-garr-pa1
- products on node14-garr-pa1
- products_db on node0-garr-ct1
- invoices on node13-garr-ct1
- invoices_db on node0-garr-pa1
- api-gateway on node0-garr-ct1

Current Placement:

- customers on node14-garr-pa1
- products on node14-garr-pa1
- invoices on node13-garr-ct1
- invoices_db on node0-garr-pa1
- customers_db on node0-garr-pa1
- api-gateway on node0-garr-ct1
- products_db on node0-garr-ct1

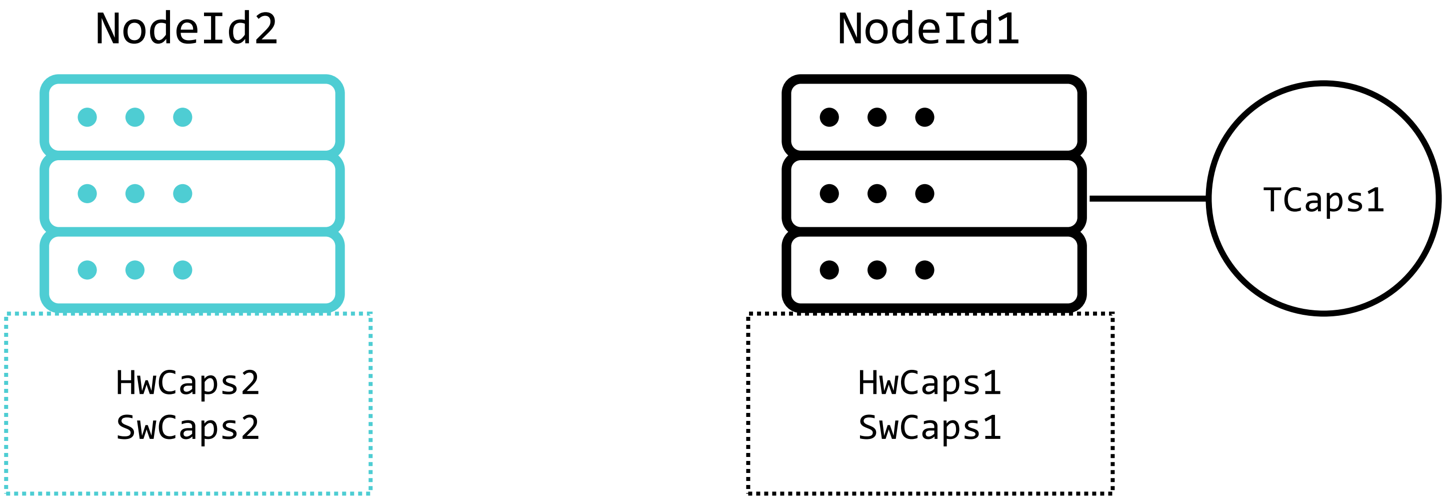


Declaring Infrastructure Capabilities



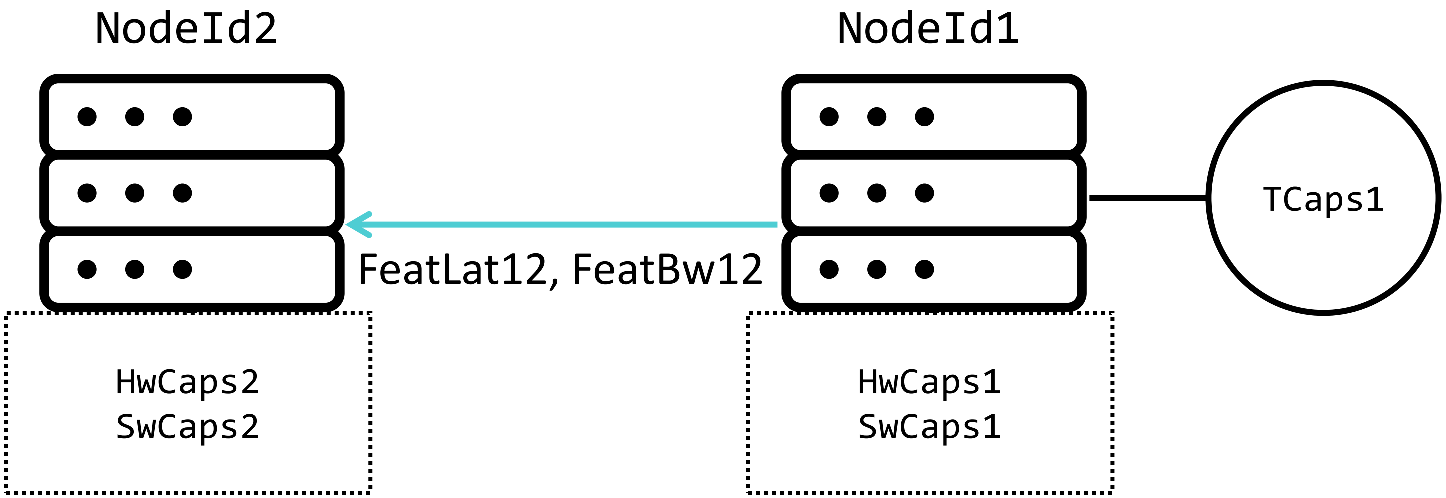
```
node(NodeId1, SwCaps1, HwCaps1, TCaps1).
```

Declaring Infrastructure Capabilities



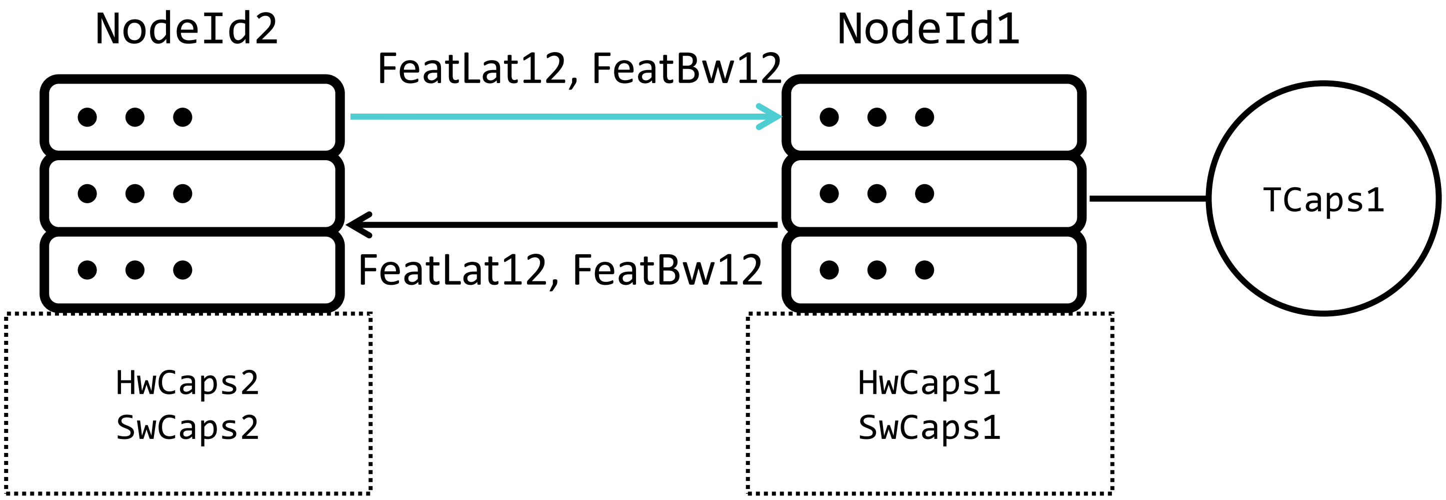
```
node(NodeId1, SwCaps1, HwCaps1, TCaps1).
node(NodeId2, SwCaps2, HwCaps2, []).
```

Declaring Infrastructure Capabilities



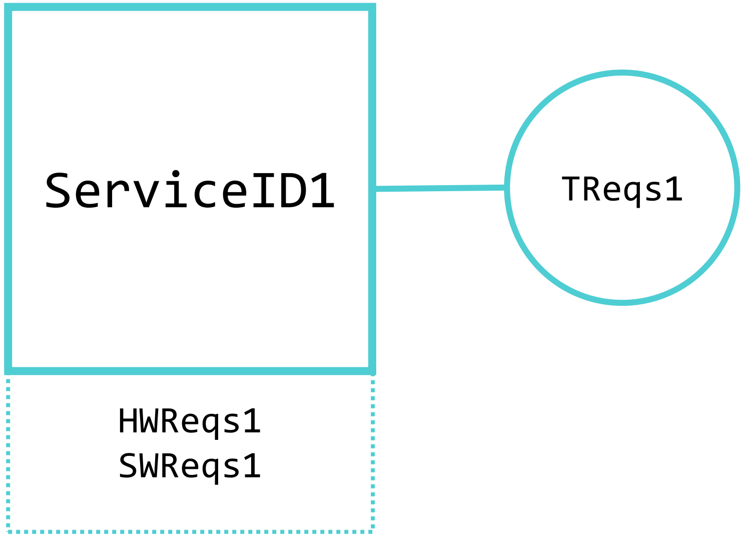
```
node(NodeId1, SwCaps1, HwCaps1, TCaps1).
node(NodeId2, SwCaps2, HwCaps2, []).
link(NodeId1, NodeId2, FeatLat12, FeatBw12).
```

Declaring Infrastructure Capabilities



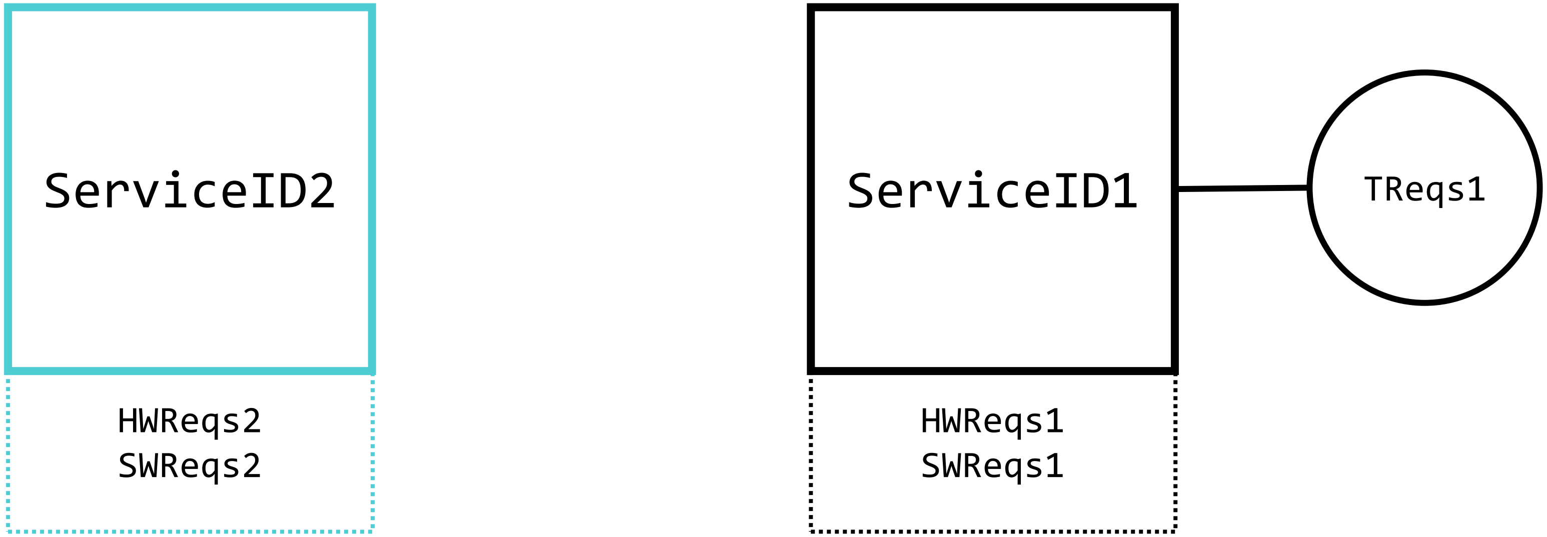
```
node(NodeId1, SwCaps1, HwCaps1, TCaps1).
node(NodeId2, SwCaps2, HwCaps2, []).
link(NodeId1, NodeId2, FeatLat12, FeatBw12).
link(NodeId2, NodeId1, FeatLat21, FeatBw21).
```

Declaring Application Requirements



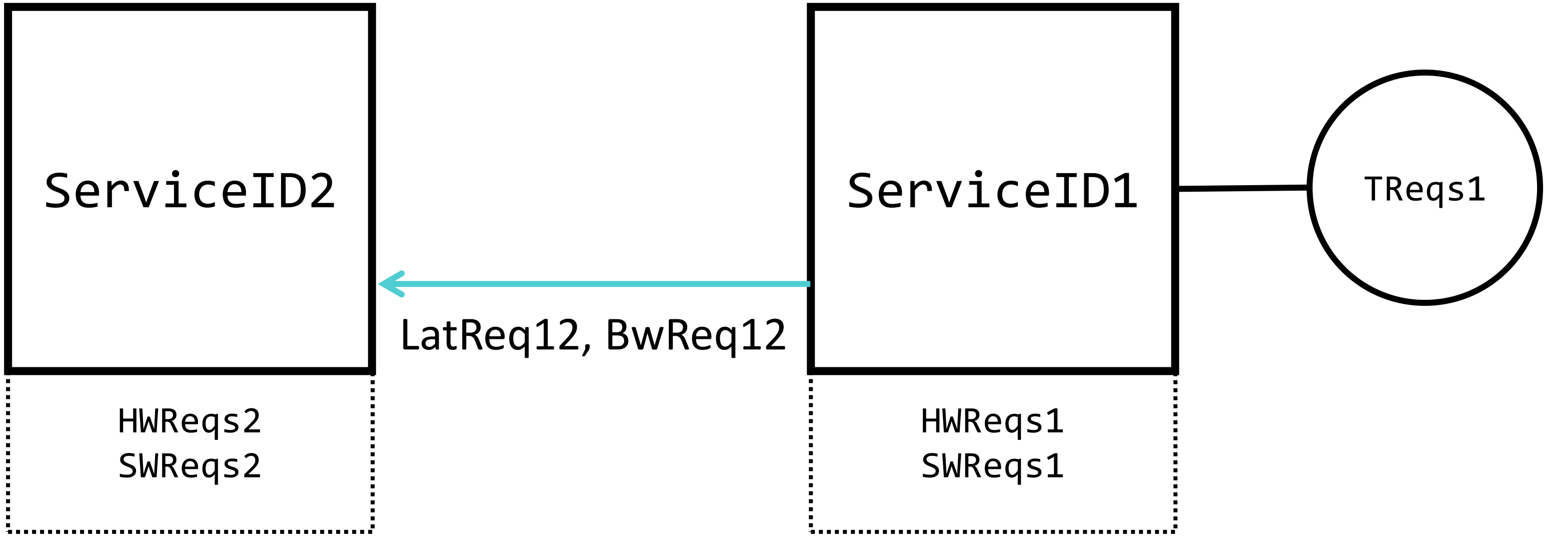
```
service(ServiceID1, SwReqs1, HwReqs1, TReqs1).
```

Declaring Application Requirements



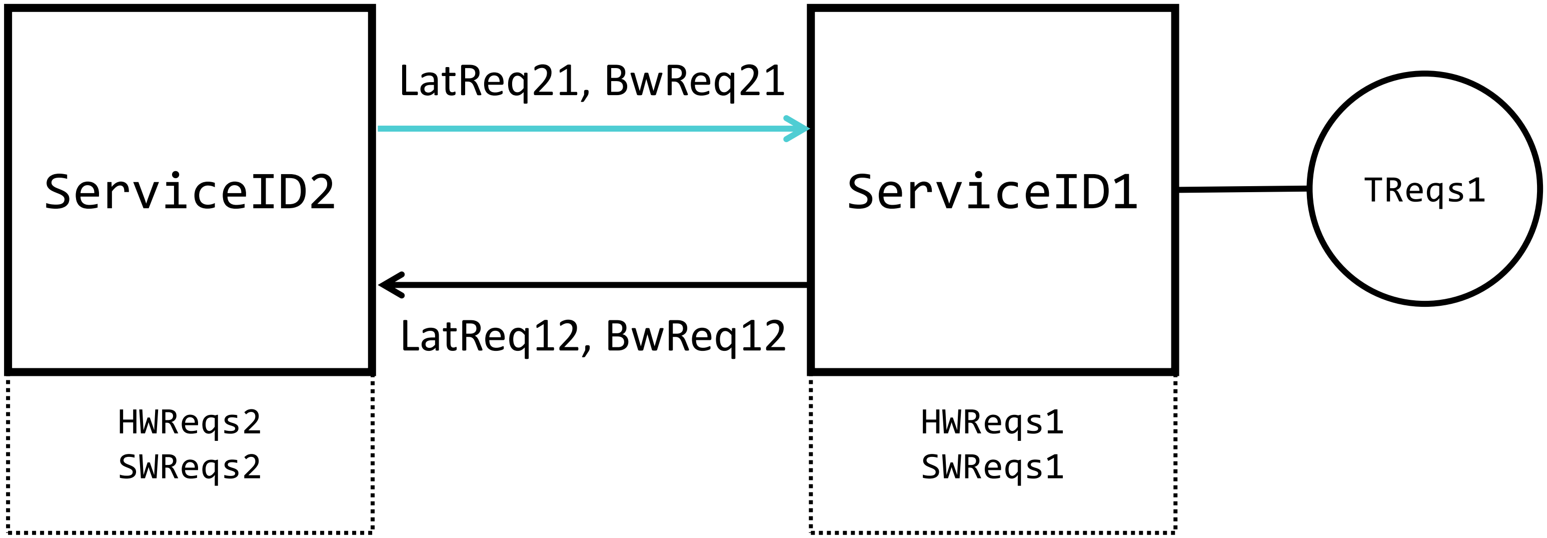
```
service(ServiceID1, SwReqs1, HwReqs1, TReqs1).
service(ServiceID2, SwReqs2, HwReqs2, []).
```

Declaring Application Requirements



```
service(ServiceID1, SwReqs1, HwReqs1, TReqs1).
service(ServiceID2, SwReqs2, HwReqs2, []).
s2s(ServiceID1, ServiceID2, LatReq12, BwReq12).
```

Declaring Application Requirements



```
service(ServiceID1, SwReqs1, HwReqs1, TReqs1).
service(ServiceID2, SwReqs2, HwReqs2, []).
s2s(ServiceID1, ServiceID2, LatReq12, BwReq12).
s2s(ServiceID2, ServiceID1, LatReq21, BwReq21).
```

Triggers

Changes in services requirements

Triggers

Infrastructural changes

Changes in services requirements

Triggers

Infrastructural changes

Changes in services requirements

Addition/removal of services

FogBrainX Reasoning

1. **First deployment**, via a **generate & test** strategy

```
fogBrainX(A,Placement) :-
    \+ deployment(A,_,_), placement(A,Placement).
```

FogBrainX Reasoning

1. **First deployment**, via a **generate & test** strategy, and
2. **Management decisions**, via **continuous reasoning**

```
fogBrainX(A,Placement) :-
    \+ deployment(A,_,_), placement(A,Placement).
fogBrainX(A,NewPlacement) :-
    deployment(A,P,Alloc),
    newServices(P,NewServices),
    reasoningStep(P,Alloc,NotOkServices,[],OkPlacement),
    append(NewServices,NotOkServices,ServicesToPlace),
    placement(ServicesToPlace,OkPlacement,Alloc,NewPlacement),
    allocatedResources(NewPlacement,NewAlloc),
    retract(deployment(A,_,_)), assert(deployment(A,NewPlacement,NewAlloc)).
```

FogBrainX Reasoning Step

1. If the service is removed, remove it form the placement

```
reasoningStep([on(S,_)|Ps],(AllocHW,AllocBW),KOs,POk,StableP) :-
    \+ service(S,_,_,_),
    reasoningStep(Ps,(AllocHW,AllocBW),KOs,POk,StableP).
```

FogBrainX Reasoning Step

1. If the service is removed, remove it form the placement
2. If the service's requirements are satisfied, keep it's placement

```
reasoningStep([on(S,_)|Ps],(AllocHW,AllocBW),KOs,POk,StableP) :-
    \+ service(S,_,_,_),
    reasoningStep(Ps,(AllocHW,AllocBW),KOs,POk,StableP).
reasoningStep([on(S,N)|Ps],(AllocHW,AllocBW), KOs, POk,StableP) :-
    nodeOk(S,N,POk,AllocHW), linksOk(S,N,POk,AllocBW),!,
    reasoningStep(Ps,(AllocHW,AllocBW),KOs,[on(S,N)|POk],StableP).
```

FogBrainX Reasoning Step

1. If the service is removed, remove it form the placement
2. If the service's requirements are satisfied, keep it's placement
3. Otherwise, re-place it

```
reasoningStep([on(S,_)|Ps],(AllocHW,AllocBW),KOs,POk,StableP) :-
    \+ service(S,_,_,_),
    reasoningStep(Ps,(AllocHW,AllocBW),KOs,POk,StableP).
reasoningStep([on(S,N)|Ps],(AllocHW,AllocBW), KOs, POk,StableP) :-
    nodeOk(S,N,POk,AllocHW), linksOk(S,N,POk,AllocBW),!,
    reasoningStep(Ps,(AllocHW,AllocBW),KOs,[on(S,N)|POk],StableP).
reasoningStep([on(S,_)|Ps],(AllocHW,AllocBW),[S|KOs],POk,StableP) :-
    reasoningStep(Ps,(AllocHW,AllocBW),KOs,POk,StableP).
reasoningStep([],_,[],P,P).
```

Default Policies

Node Requirements (SW, IoT and cumulative HW)

```
nodeOk(S,N,P,AllocHW) :-
    service(S,SWReqs,HWReqs,IoTReqs),
    node(N,SWCaps,HWCaps,IoTCaps),
    swReqsOk(SWReqs,SWCaps),
    thingReqsOk(IoTReqs,IoTCaps),
    hwOk(N,HWCaps,HWReqs,P,AllocHW)
```

Default Policies

Node Requirements (SW, IoT and cumulative HW)

```
nodeOk(S,N,P,AllocHW) :-
    service(S,SWReqs,HWReqs,IoTReqs),
    node(N,SWCaps,HWCaps,IoTCaps),
    swReqsOk(SWReqs,SWCaps),
    thingReqsOk(IoTReqs,IoTCaps),
    hwOk(N,HWCaps,HWReqs,P,AllocHW)
```

Links Requirements (latency and cumulative bandwidth)

```
linksOk(S,N,P,AllocBW) :-
    findall((N1N2,ReqLat), distinct(relevant(S,N,P,N1N2,ReqLat)), N2Ns),
latencyOk(N2Ns),
    findall(N1N2, distinct(member((N1N2,ReqLat),N2Ns)), N1N2s), bwOk(N1N2s,
AllocBW, [on(S,N)|P]).
```

FogBrainX Placer

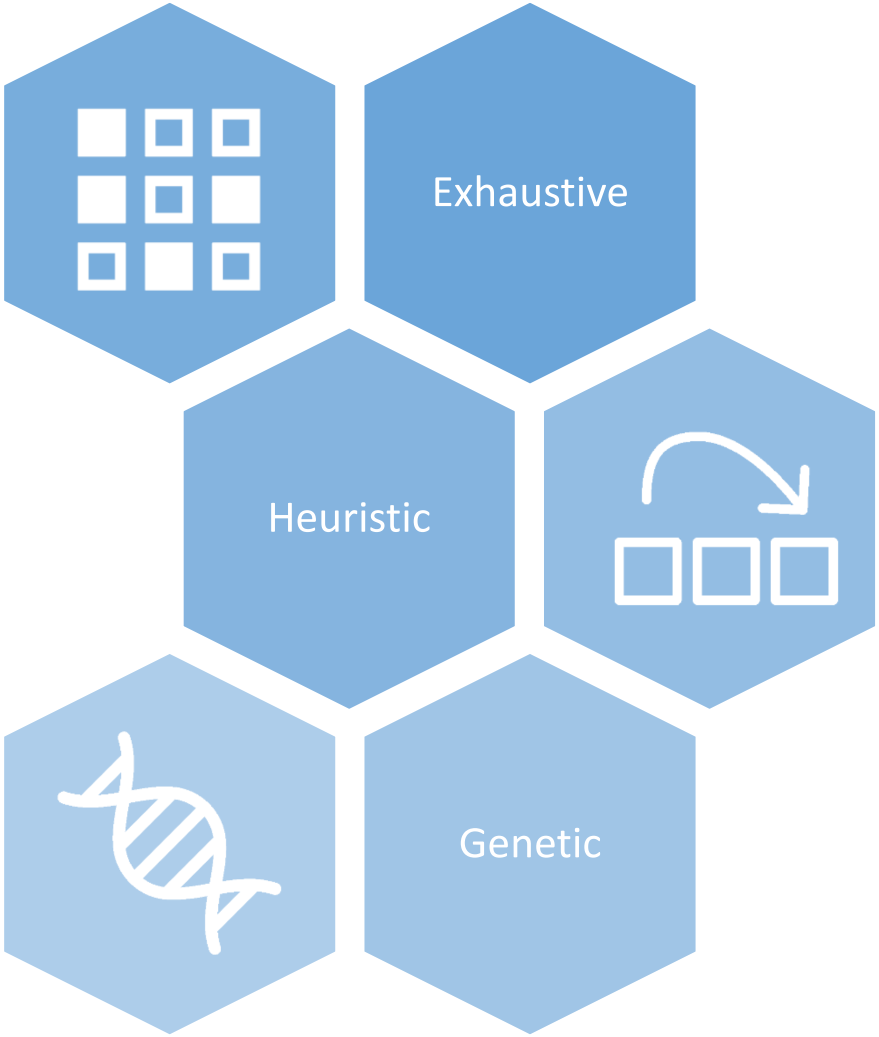
Exploiting ***generate & test*** **(with backtracking)**

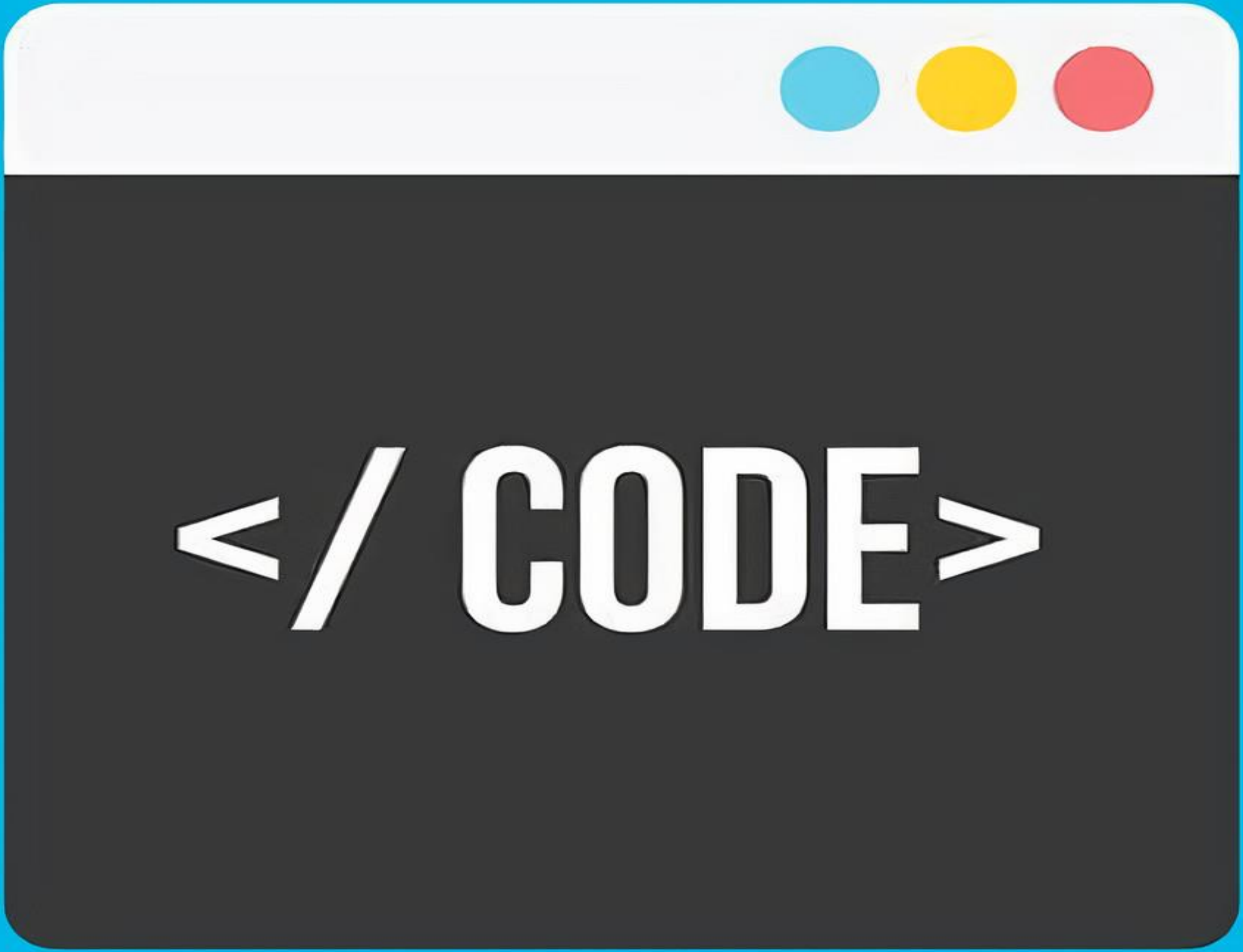
1. Checks node requirements for **service S on node N**

```
placement([S|Ss],P,(AllocHW,AllocBW),Placement) :-
    nodeOk(S,N,P,AllocHW), linksOk(S,N,P,AllocBW),
    placement(Ss,[on(S,N)|P],(AllocHW,AllocBW),Placement).
placement([],P,_,P).
```

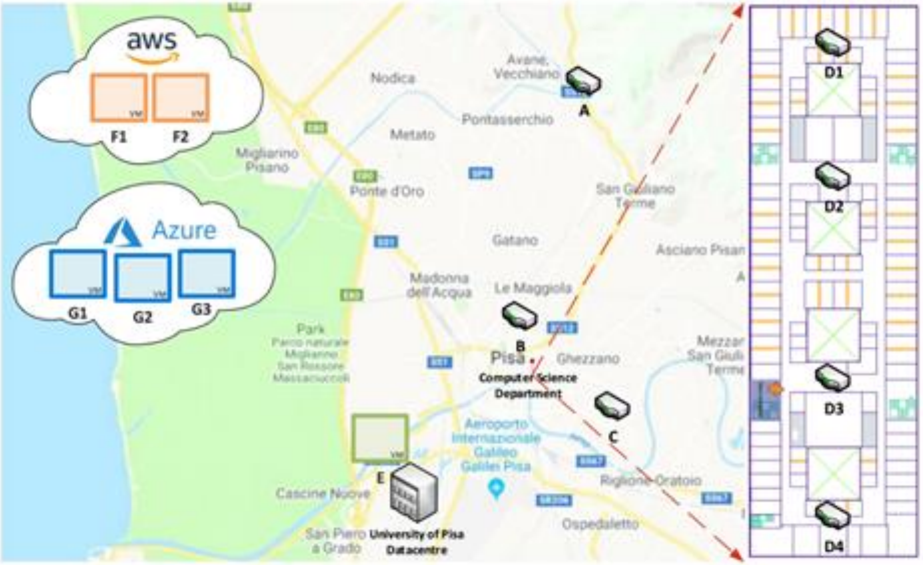
2. Checks link requirements for all (placed) **services communicating with S**
3. Repeated **until all services have been placed**

Continuous Reasoning as a Booster



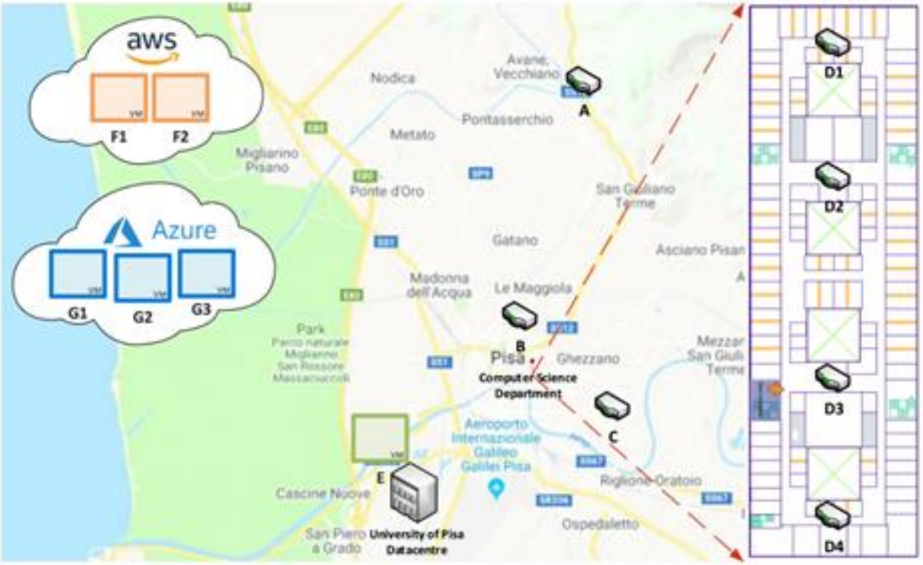


SockShop Use Case

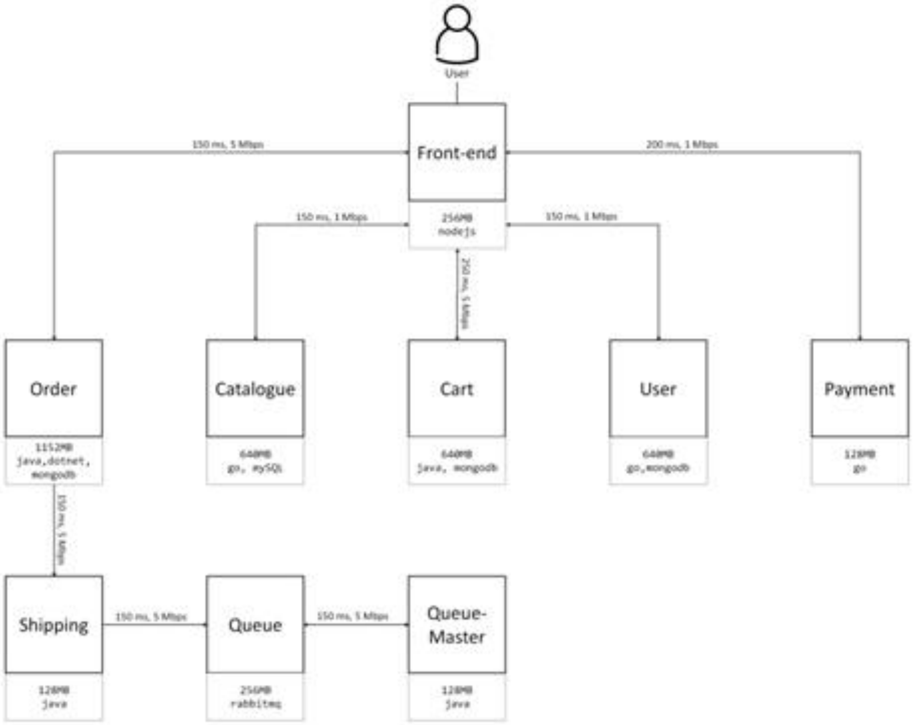


(a) Use case infrastructure.

SockShop Use Case

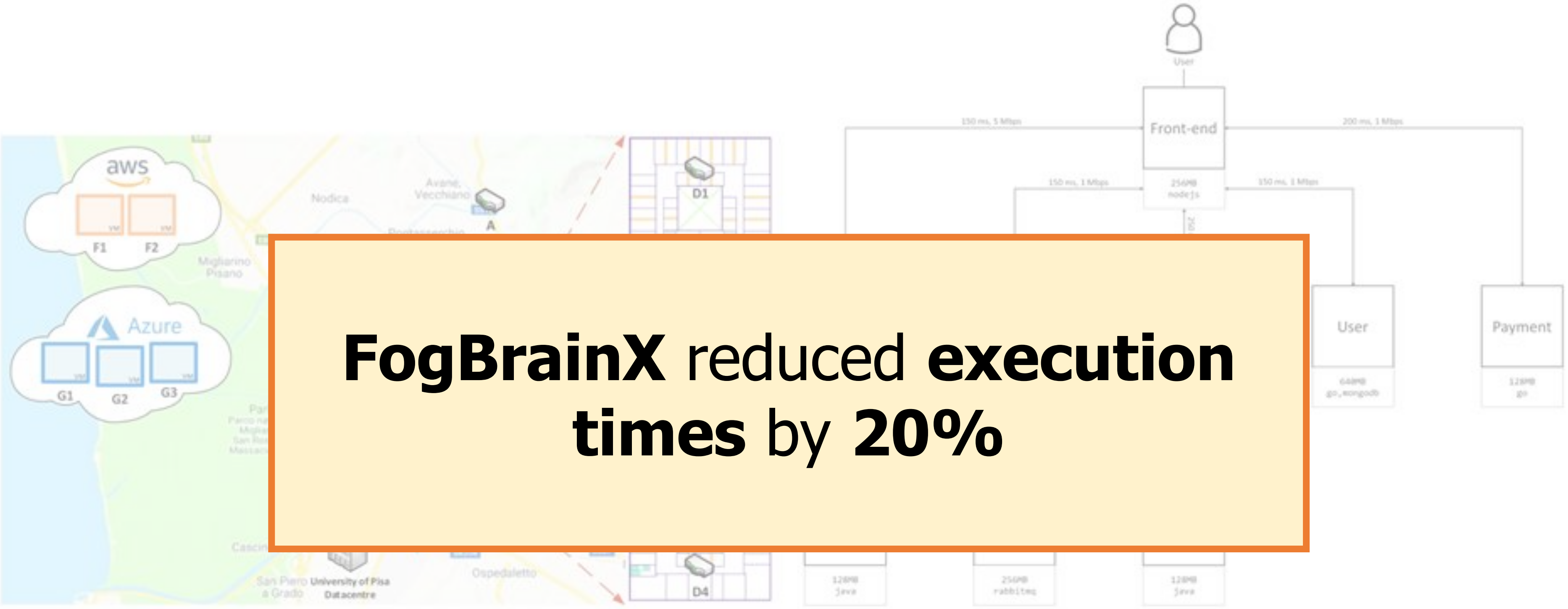


(a) Use case infrastructure.



(b) Use case application specification.

SockShop Use Case

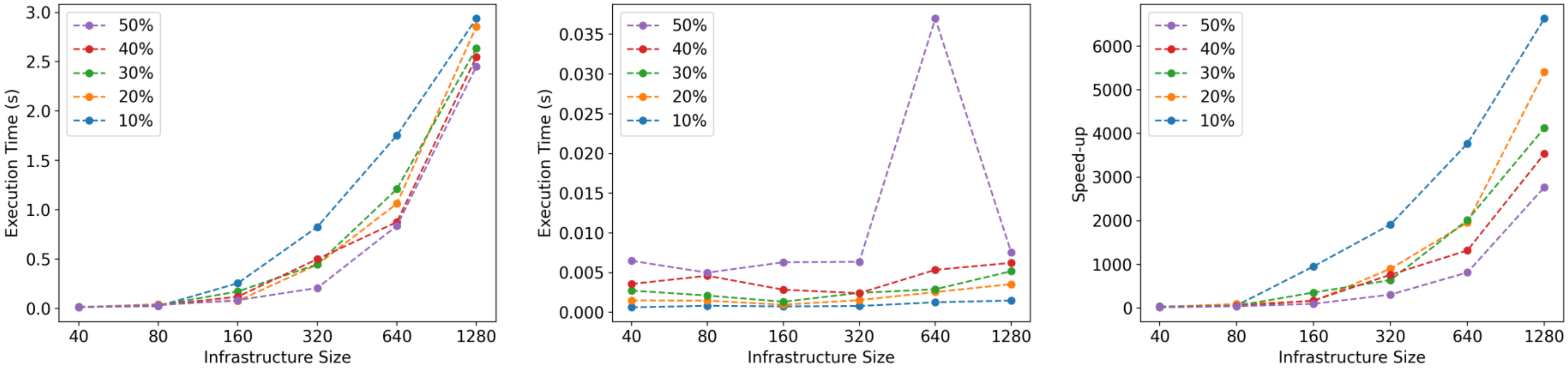


FogBrainX reduced **execution times** by **20%**

(a) Use case infrastructure.

(b) Use case application specification.

FogBrainX Scalability Assessment

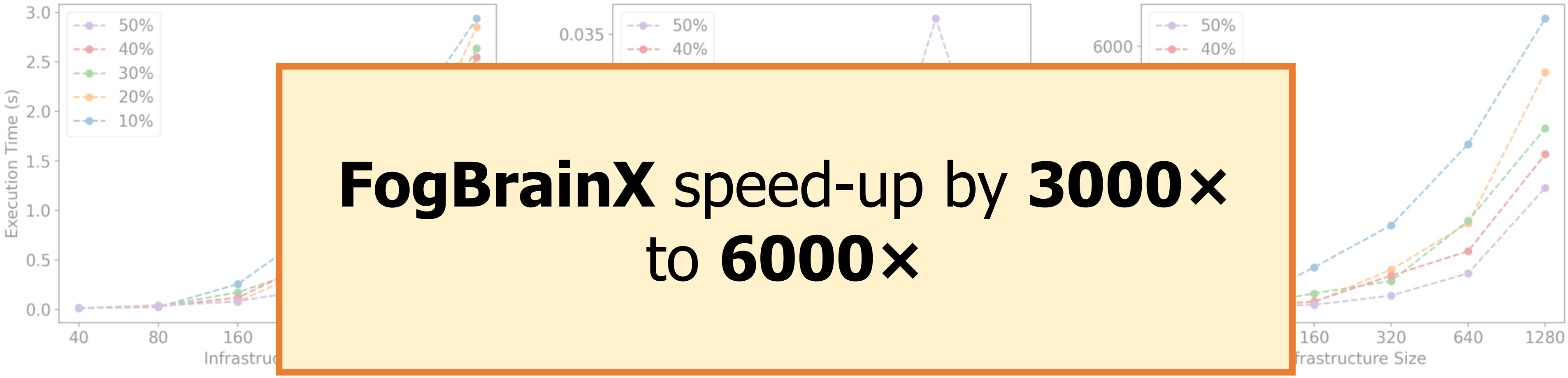


(a) Exhaustive placement

(b) Continuous Reasoning

(c) Speed-up.

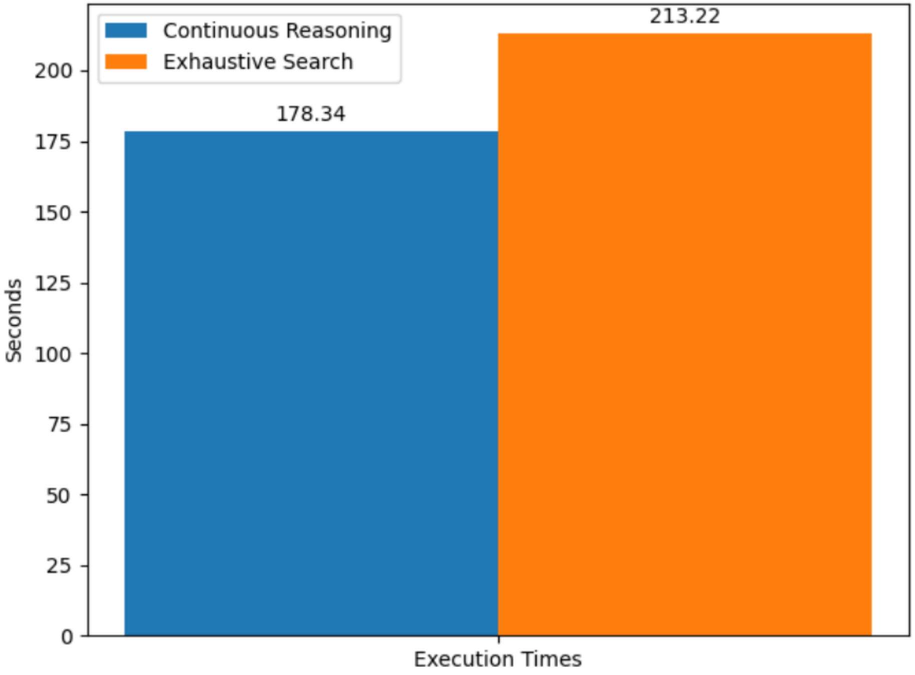
FogBrainX Scalability Assessment



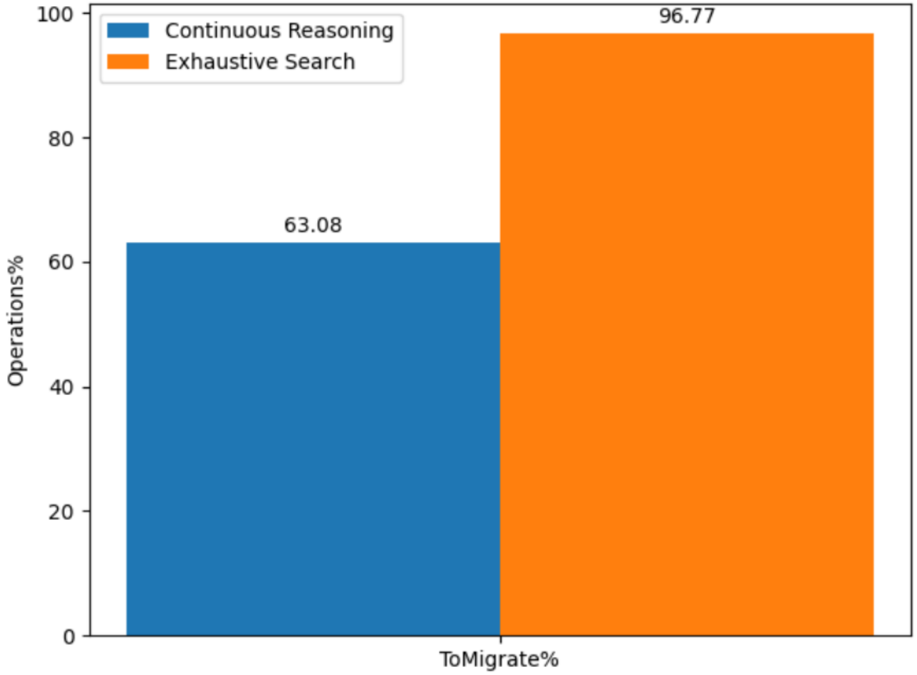
(a) Exhaustive placement (b) Continuous Reasoning (c) Speed-up.

FogBrainX speed-up by **3000×** to **6000×**

Continuous Reasoning Assessment

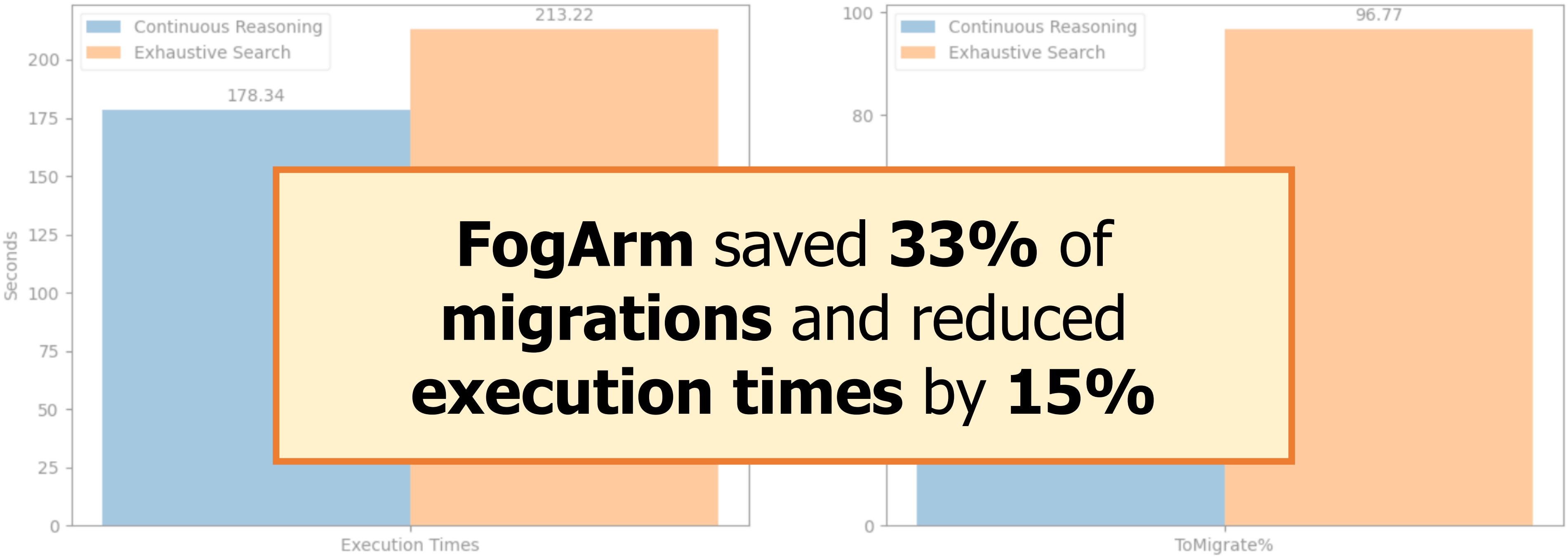


(a) Average times.



(b) Average migrations.

Continuous Reasoning Assessment



FogArm saved **33%** of **migrations** and reduced **execution times** by **15%**

(a) Average times.

(b) Average migrations.

Conclusions

FogBrainX

- FogBrainX is a declarative engine to support application management via continuous reasoning, considering both variations in the infrastructure and changes in the application requirements.
- FogBrainX speed-up placement decision-making execution time in the order of 3000× to 6000×, even in the presence of thousand of nodes and high variation rates.
- Speed-up increases as the infrastructure size increases.

- FogArm is a prototype of a next-gen orchestrator for the continuous QoS-compliant management of multiservice applications on geographically distributed Cloud-IoT infrastructures.
- Scales up to tens of nodes and hundreds of services saving 15% of execution time and migrating 33% fewer services.

FogArm

Future Work

Support data migrations

Future Work

Support data migrations

Support scaling and adaptation of services

Future Work

Support scaling and adaptation of services

Support data migrations

Cost models & heuristics

Future Work

Support scaling and adaptation of services

Support data migrations

Theoretical compositional model

Cost models & heuristics

Thank You