AI-SPRINT

Docker Containers

Germán Moltó

Departamento de Sistemas Informáticos y Computación - Instituto de Instrumentación para Imagen Molecular

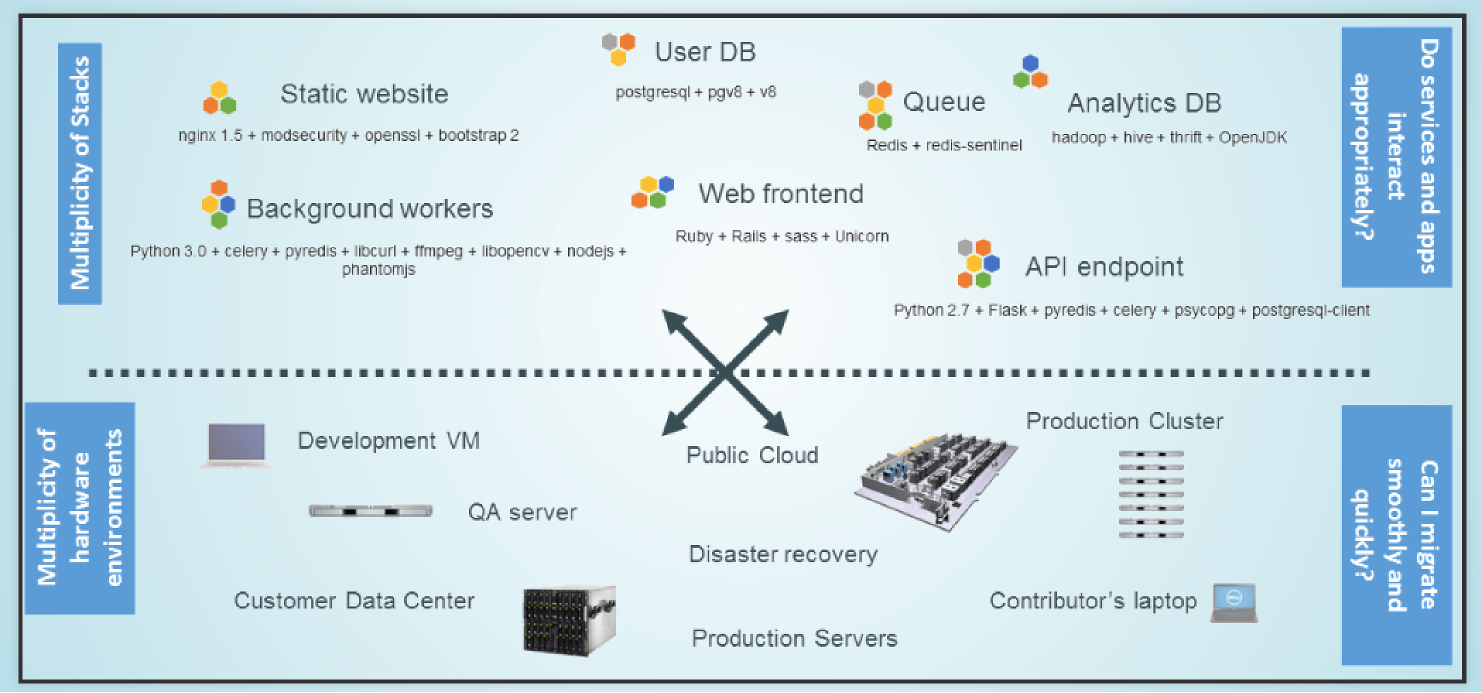
gmolto@dsic.upv.es

https://www.grycap.upv.es/gmolto

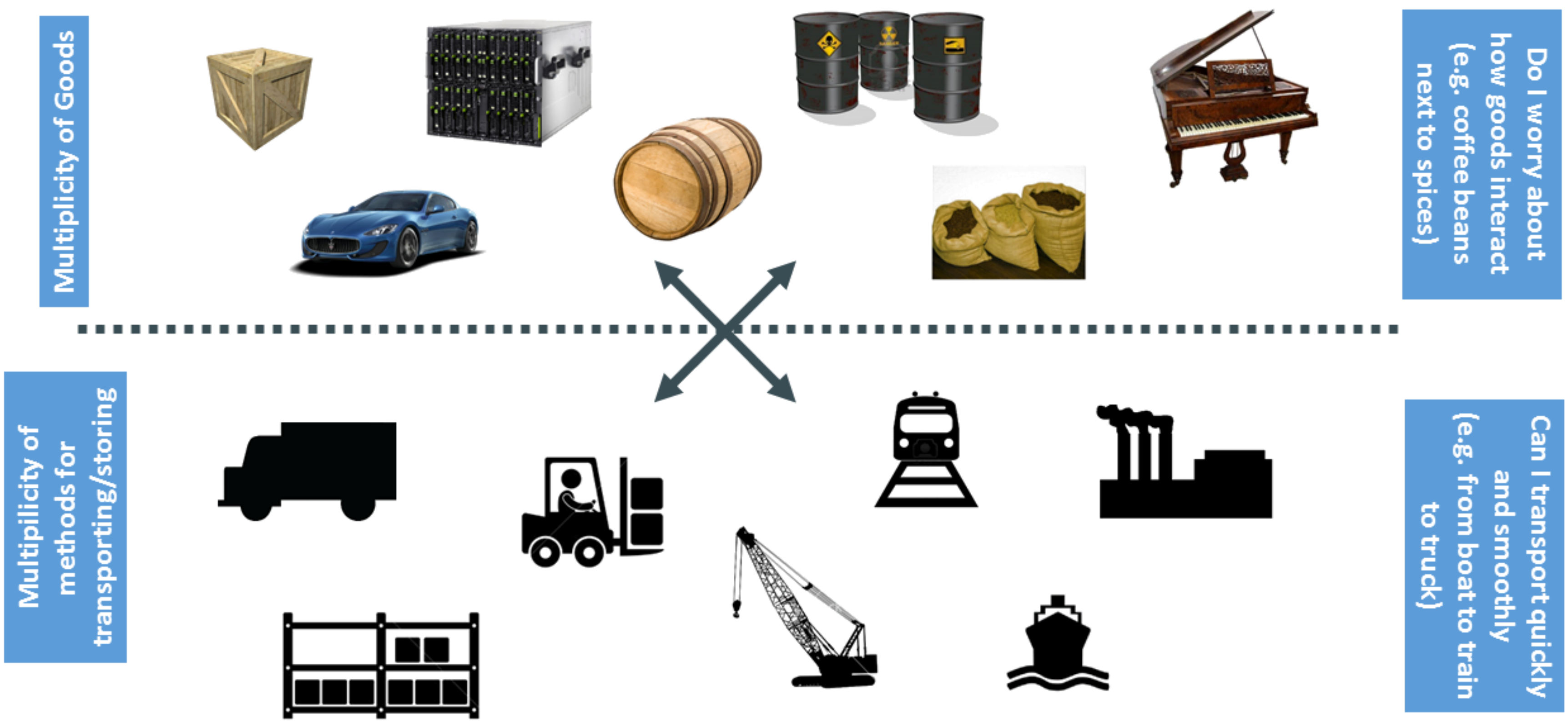
Beyond Virtual Machines …

- Virtual machines have introduced numerous advantages:
  - Server consolidation, isolation between applications, etc.
- But:
  - Virtual machine images are heavy and specific for each hypervisor.
  - They take (little) time to start.
  - Virtualization overhead
- What if you could run processes on the same host in isolation and securely?

Problem

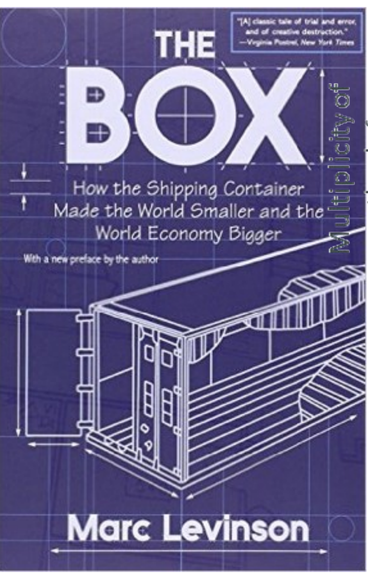
- Developing distributed applications requires different OS, programming languages, execution environments, libraries, etc. and can be deployed on multiple platforms.
- 

Analogy with the Real World



http://disney.github.io/docker-training

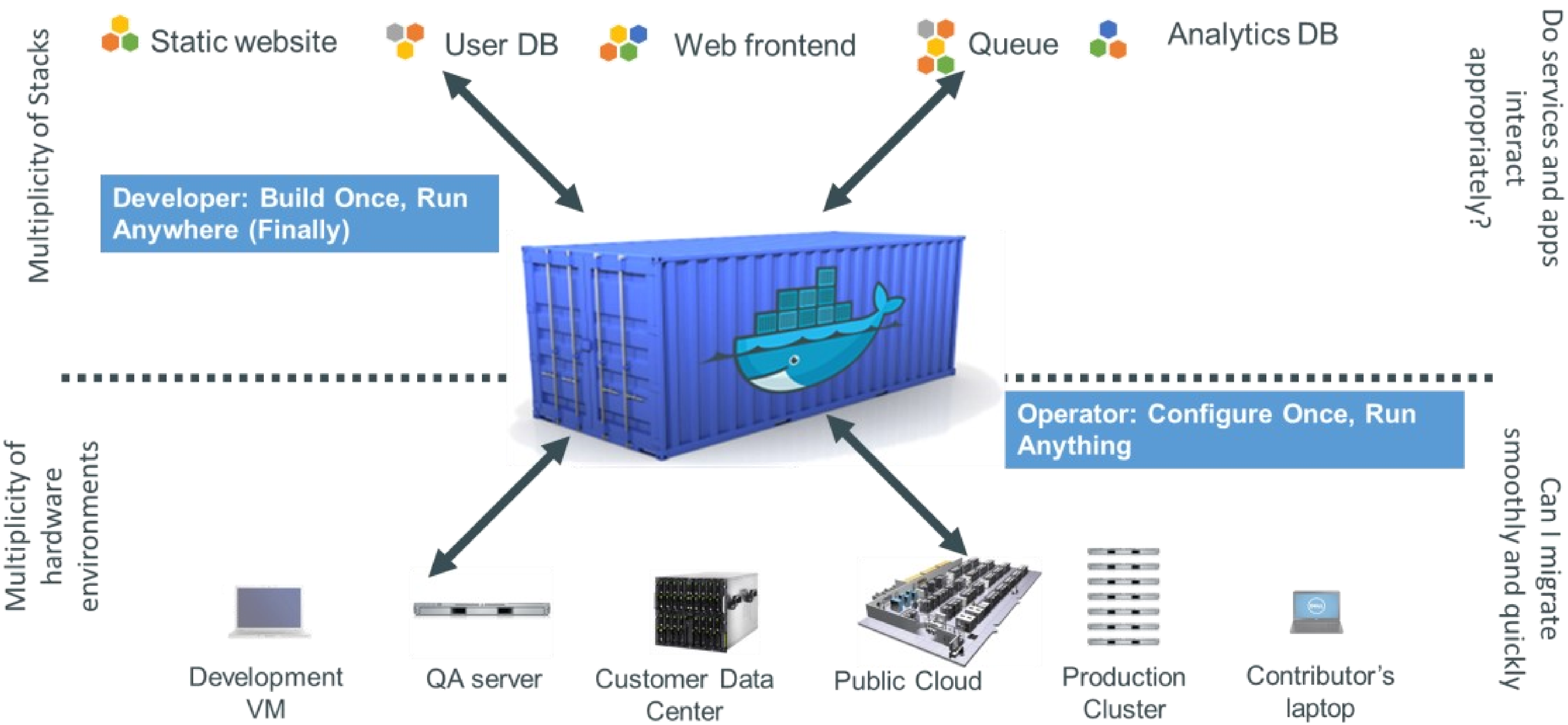
Real-World Solution



http://disney.github.io/docker-training

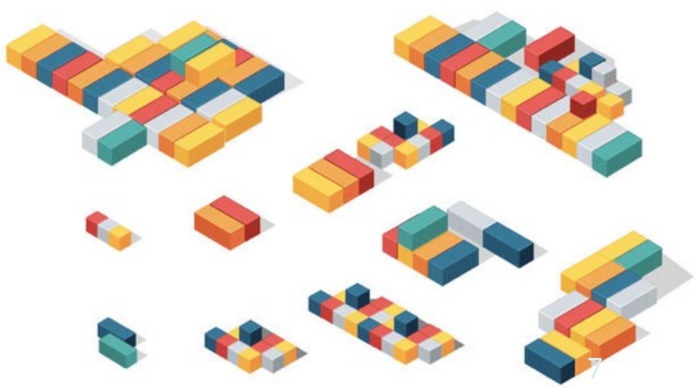
http://www.amazon.com/The-Box-Shipping-Container-Smaller/dp/0691136408

Docker Containers

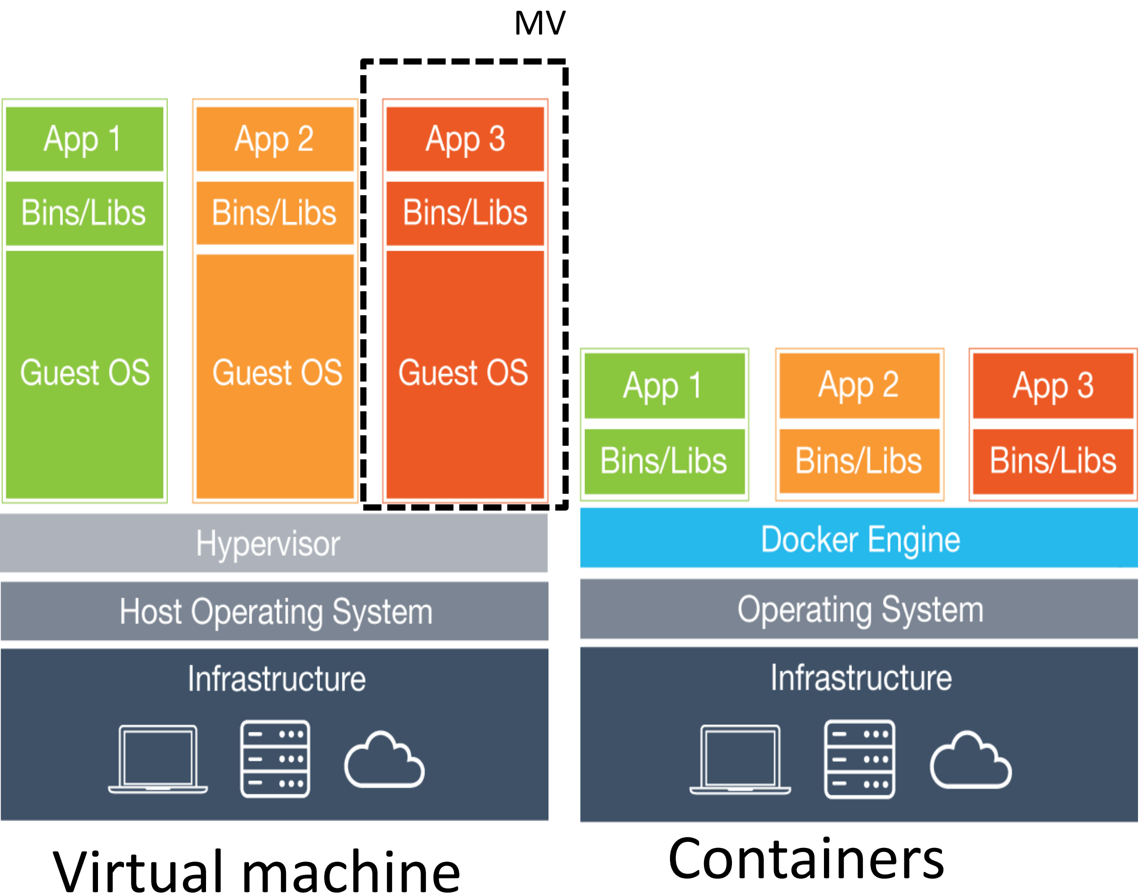


Containers

- A container is an encapsulation of an entire file system that contains everything needed to run an application (code, libraries, OS, tools, etc.)
- Instead of emulating hardware (such as VMs) they use technologies such as **cgroups** and Linux kernel **namespaces** to create the containers.
- Container Technologies:
  - LXC – https://linuxcontainers.org
  - Docker - https://www.docker.com
  - ~~rkt- https://coreos.com/rkt~~



Containers vs Virtual Machines



- Containers (PROS)
  - Smaller image size
  - Instant execution
  - No virtualization overhead
  - Encapsulates all dependencies, ensuring correct execution
  - Write Once Run Anywhere*

Containers (CONS)

  - Unable to run Windows on Linux
  - Security isolation
    - Host kernel sharing

* x86 con Linux 3.2+ ó 2.6.32+ para Fedora, CentOS, etc.

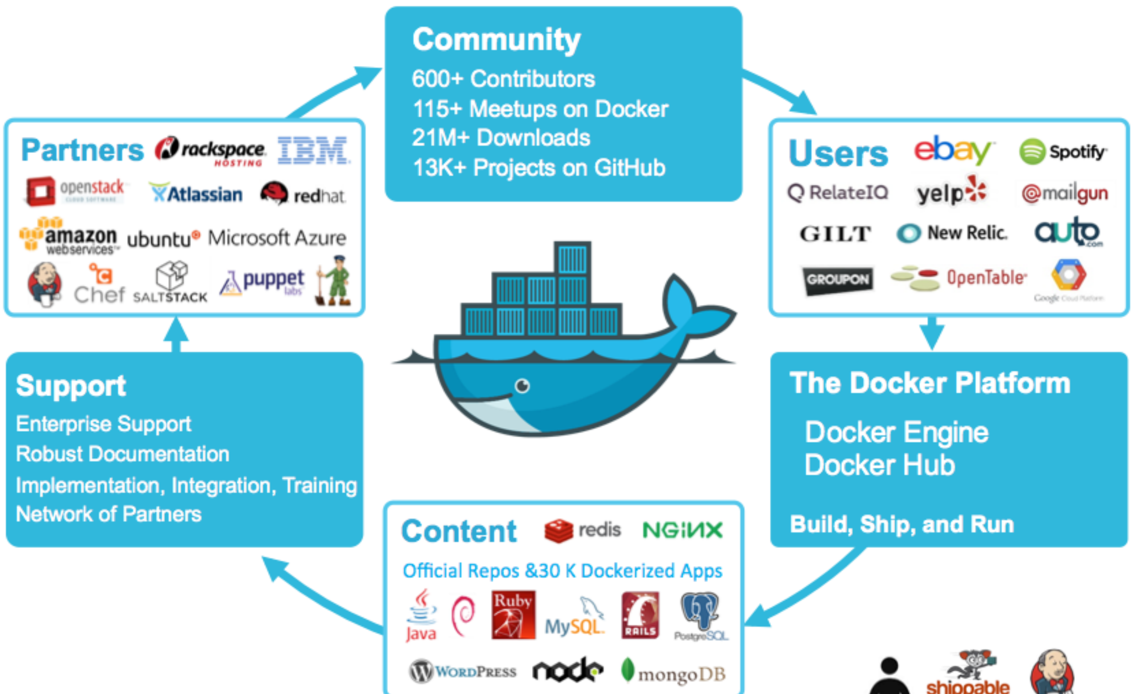
Containers or Virtual Machines?

- For Linux virtualization scenarios on Linux, containers can offer a very good advantage over virtual machines
- In full virtualization scenarios (e.g. Windows over Linux), virtual machines must be used.
- In the field of Cloud Computing, virtual machines are used as computing capacity on which multiple containers with the applications are then executed.
  - Take advantage of the multiple vCPUs of a virtual machine.

What is Docker?

- Docker – https://www.docker.com/
- An open platform for developers and system administrators to build, ship and run distributed applications.
- Package an application with all its dependencies (OS, libraries, applications, etc.) to be executed on different platforms.
  - Objective: Fast, *consistent delivery of applications*
- Deploy application runtime environments quickly and repeatably.

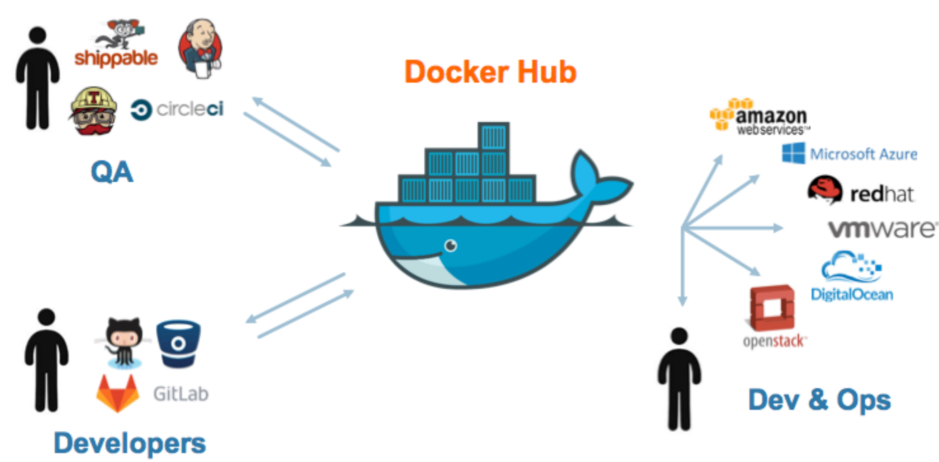
Docker



Docker has had spectacular growth in recent years.

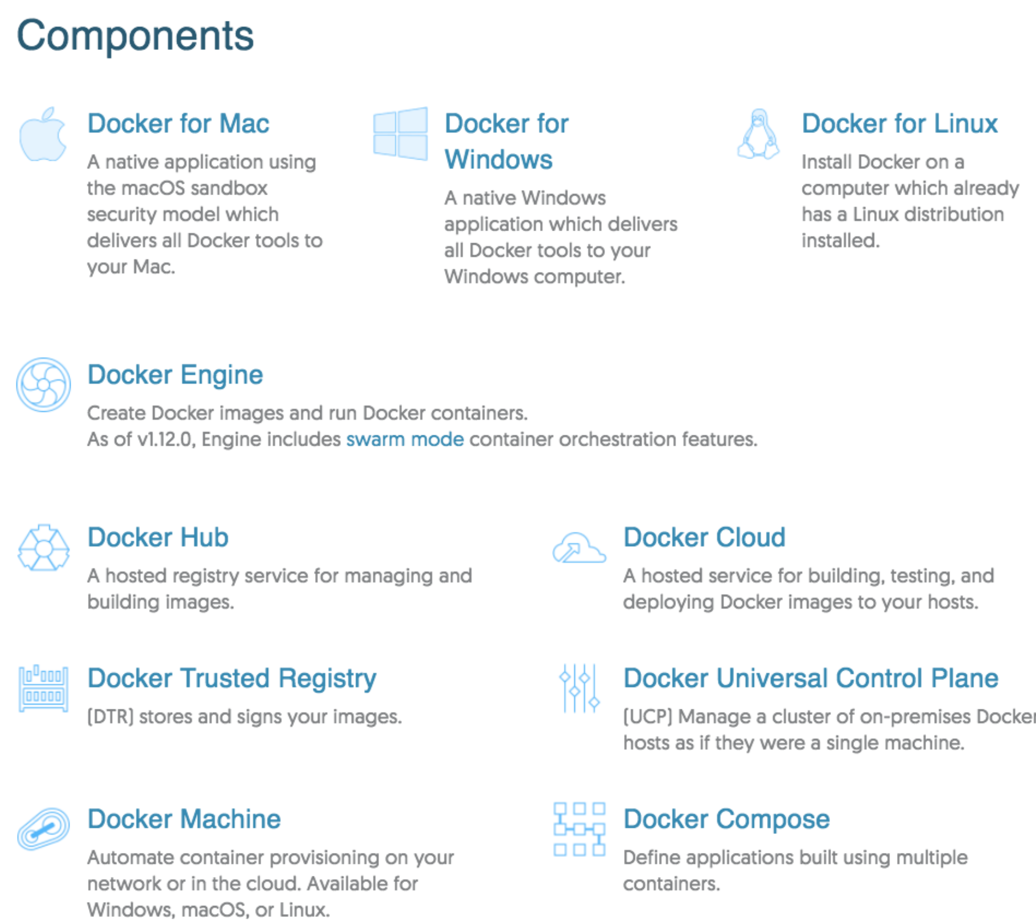
Many adoption scenarios.

- Although with difficulties to monetize it.
  - https://www.zdnet.com/article/docker-is-in-deep-trouble/



Docker Components

- Docker consists of an ecosystem of tools around Docker Engine.
- OCI (Open Container Initiative)
  - https://www.opencontainers.org/



Components

Docker for Mac
A native application using the macOS sandbox security model which delivers all Docker tools to your Mac.

Docker for Windows
A native Windows application which delivers all Docker tools to your Windows computer.

Docker for Linux
Install Docker on a computer which already has a Linux distribution installed.

Docker Engine
Create Docker images and run Docker containers.
As of v1.12.0, Engine includes swarm mode container orchestration features.

Docker Hub
A hosted registry service for managing and building images.

Docker Cloud
A hosted service for building, testing, and deploying Docker images to your hosts.

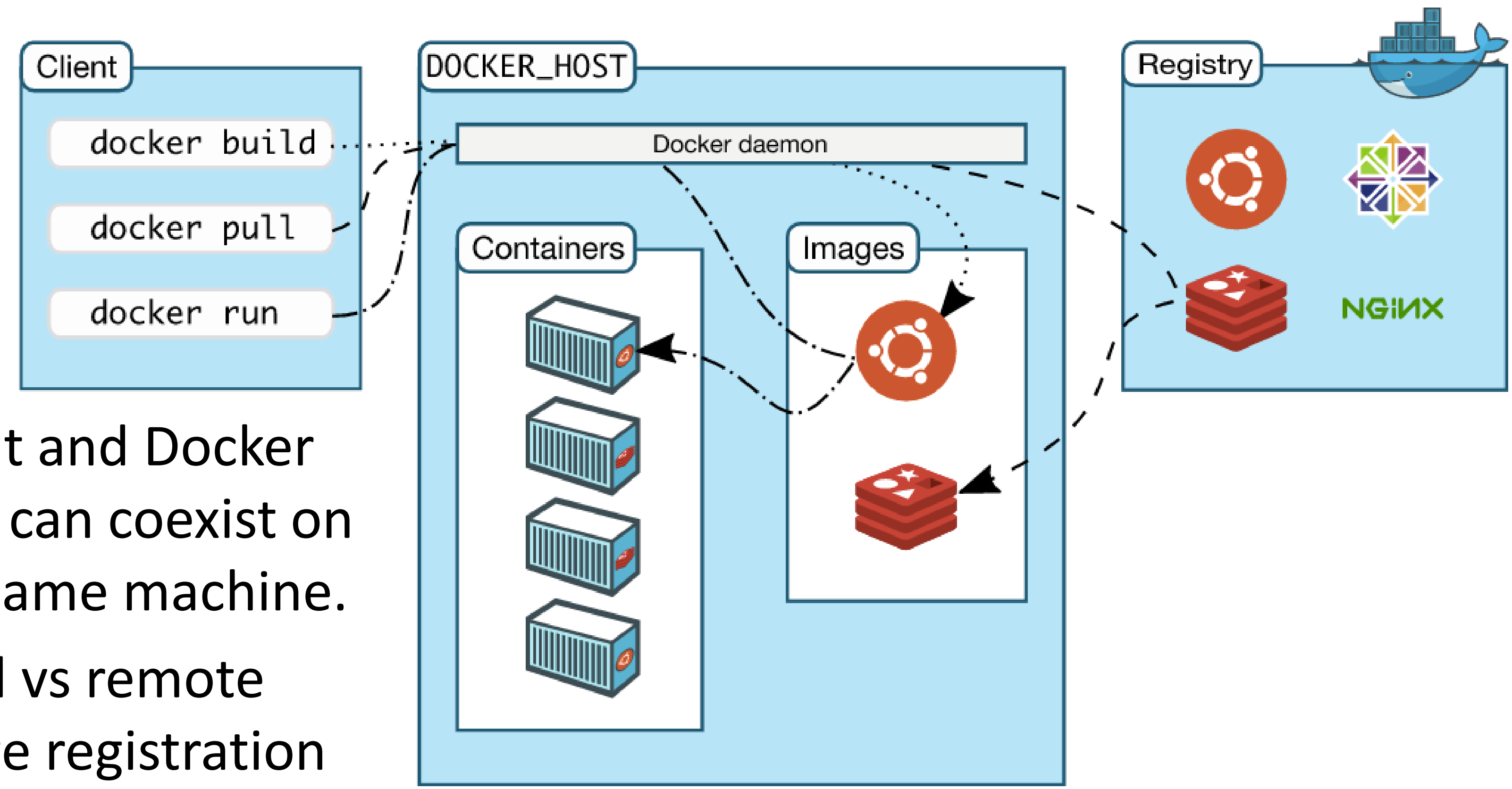
Docker Trusted Registry
(DTR) stores and signs your images.

Docker Universal Control Plane
(UCP) Manage a cluster of on-premises Docker hosts as if they were a single machine.

Docker Machine
Automate container provisioning on your network or in the cloud. Available for Windows, macOS, or Linux.

Docker Compose
Define applications built using multiple containers.

Docker Engine architecture



- Client and Docker Host can coexist on the same machine.
- Local vs remote image registration (e.g. Docker Hub).

Installing Docker Engine

curl -fsSL https://get.docker.com/ | sh

- There are other ways to install it:
  - https://docs.docker.com/engine/installation/

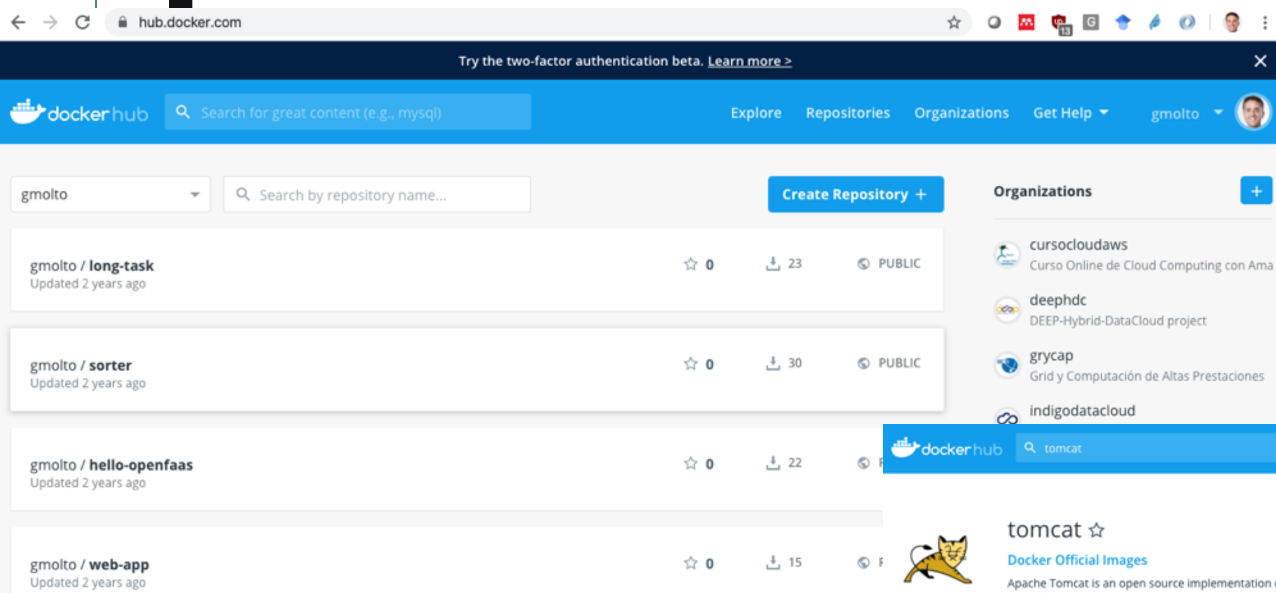
Docker Engine Basics

- **Image**
  - It contains an OS distribution (e.g. Ubuntu 22.04) and a certain configuration of packages/applications/data determined by the creator of the image.
- **Docker Hub**
  - Image catalog and repository, accessible via CLI, web interface and REST API.
- **Container**
  - It is an instance of a specific image executed as an isolated process on a specific machine (Docker Host)
- **Docker Host**
  - It is the machine that has installed Docker Engine and runs the containers.
- **Docker Client**
  - The machine from which the deployment of Docker containers is requested (can match the Docker Host). Also corresponds to the client tool for interacting with Docker.

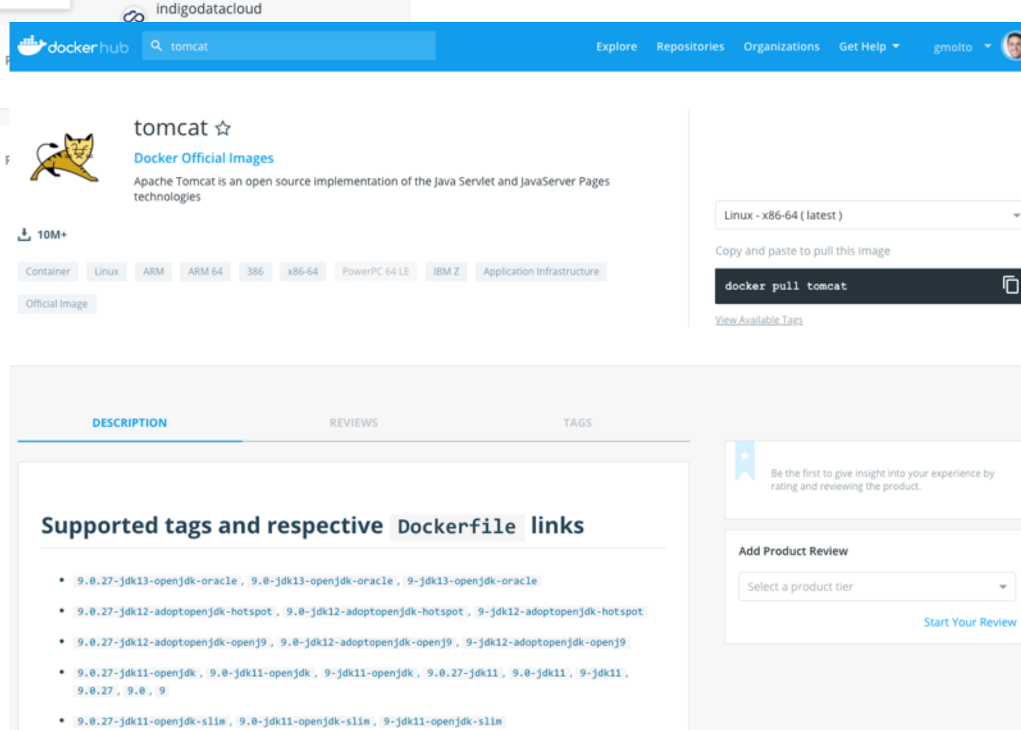
What can you do with Docker Engine?

- Manage the container lifecycle
  - start, stop, kill, restart, etc.
- Manage container images
  - push, pull, tag, rmi, etc.
- Inspect/access the container
  - logs, attach
- …
- And where can we find a catalog of Docker images?

Docker Hub



https://hub.docker.com



- Repositories containing Docker container images
- Automated Builds from GitHub

Docker 101: Containers (1)

```
gmolto@felis-2  ~  docker run alpine echo hello world
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
0a8490d0dfd3: Pull complete
Digest: sha256:dfbd4a3a8ebca874ebd2474f044a0b33600d4523d03b0df76e5c5986cb02d7e8
Status: Downloaded newer image for alpine:latest
hello world
```

- Docker automatically downloads the alpine:latest image from Docker Hub
- Stores it in the Docker Engine local registry of the Docker Host
- Run the container and, within it, the command, displaying the output on the screen.

Docker 101: Images (1)

- Docker images contain (certain libraries) + Apps.
- They can be tagged and stored in different Docker registries.
- https://docs.docker.com/registry/deploying/

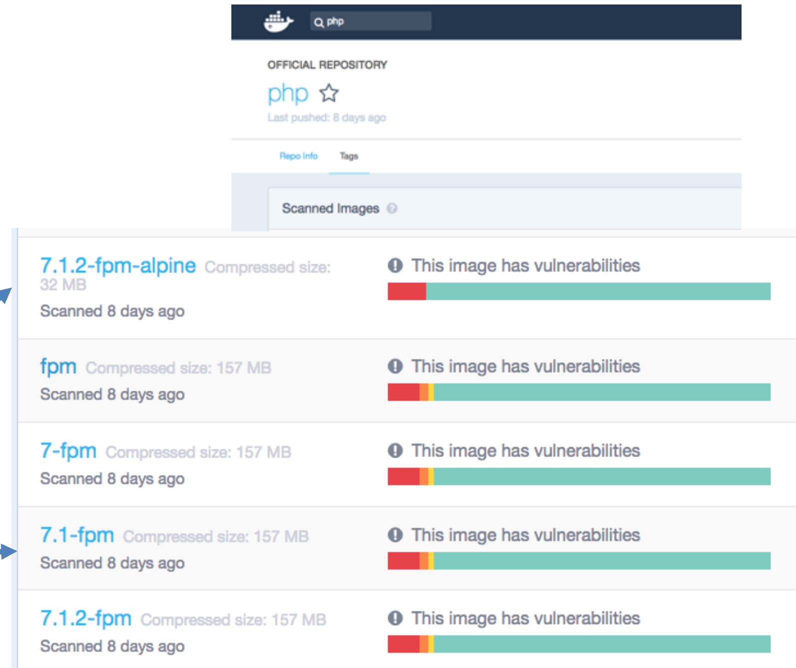
```
 gmolto@felis-2  ~  docker images
REPOSITORY                          TAG                     IMAGE ID        CREATED         SIZE
redis                               latest                  74d8f543ac97    9 days ago      184 MB
jjmerelo/docker-daleksay            latest                  5bf18c53ecd5    3 weeks ago     72.1 MB
starefossen/node-imagemagick        latest                  fd39b463447c    3 weeks ago     675 MB
busybox                             latest                  7968321274dc    3 weeks ago     1.11 MB
alpine                              latest                  88e169ea8f46    6 weeks ago     3.98 MB
examplevotingapp_result             latest                  4b1b9a9aa48e    8 weeks ago     227 MB
examplevotingapp_worker             latest                  a9bb84ce3459    8 weeks ago     574 MB
examplevotingapp_vote               latest                  607747fc0e0c    8 weeks ago     84.3 MB
postgres                            9.4                     452864725827    8 weeks ago     265 MB
grycap/odisea                       latest                  5e795229a921    2 months ago    560 MB
redis                               alpine                  9947c5a33865    2 months ago    21 MB
python                              2.7-alpine              9c8c07c0f9b7    2 months ago    72.2 MB
microsoft/dotnet                    1.0.0-preview2-sdk      6704971aa9c1    3 months ago    537 MB
jpetazzo/trainingwheels             latest                  db38019622f1    8 months ago    686 MB
node                                5.11.0-slim             cb888ea932ad    9 months ago    207 MB
 gmolto@felis-2  ~
```

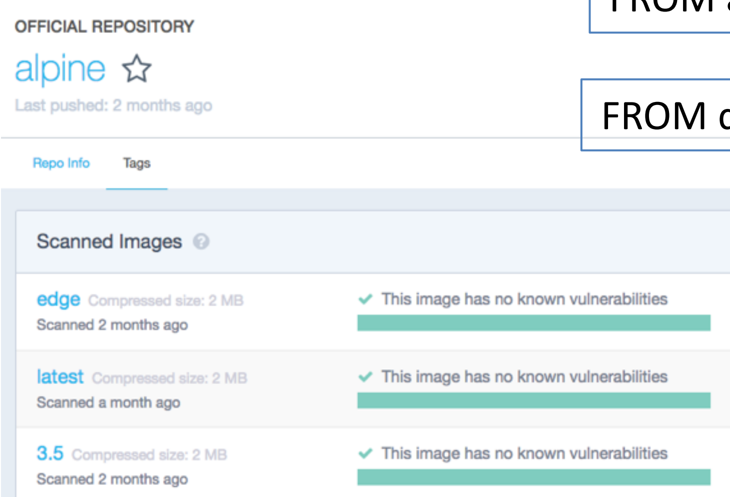
Docker 101: Images (2)

- Size matters.
  - Reduction to one quarter of the size of the original image when using Alpine base OS as a base OS against other distributions (e.g. Ubuntu, CentOS, etc.)

FROM alpine:3.4

FROM debian:jessie

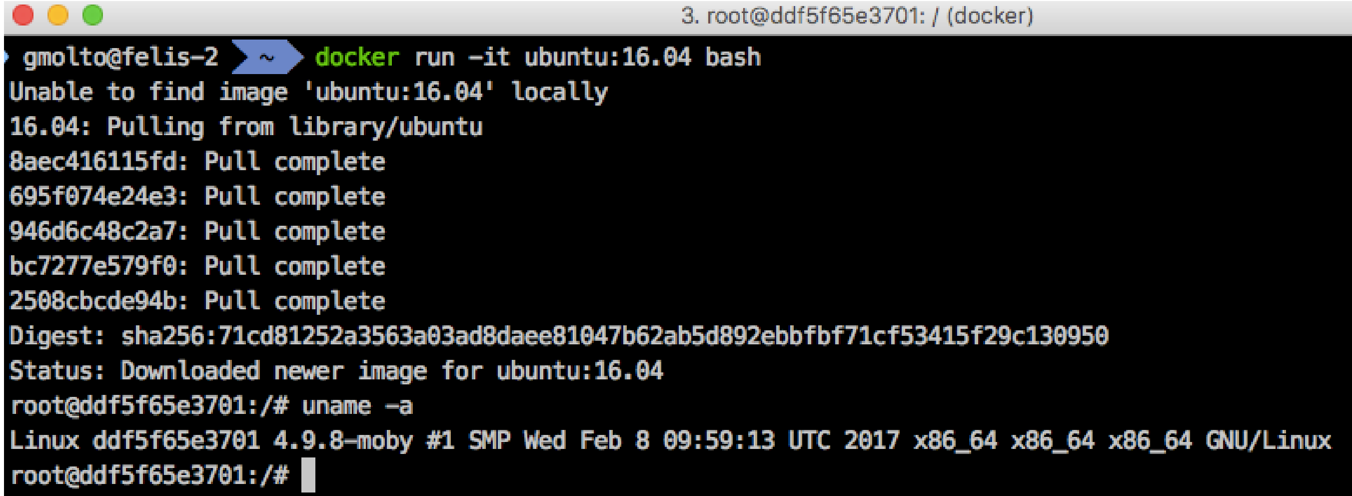




https://hub.docker.com/r/_/alpine/
https://alpinelinux.org/

Docker 101: Containers (2)

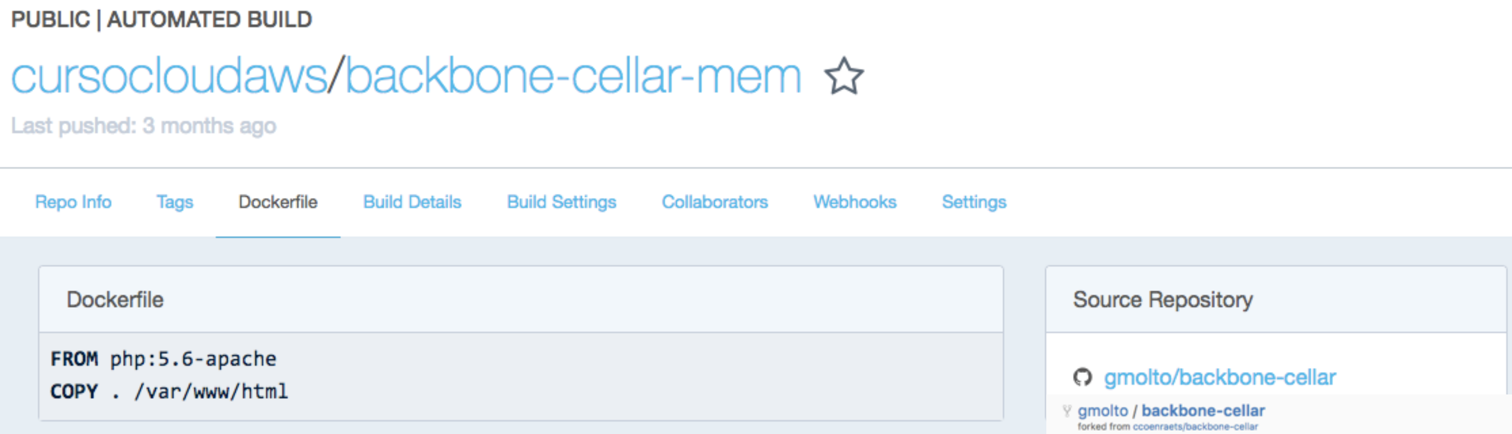
- Interactive session with a Docker container
  - docker run -it ubuntu:22.04 bash
  - This container can be used like any machine: install applications, output to the Internet, etc.

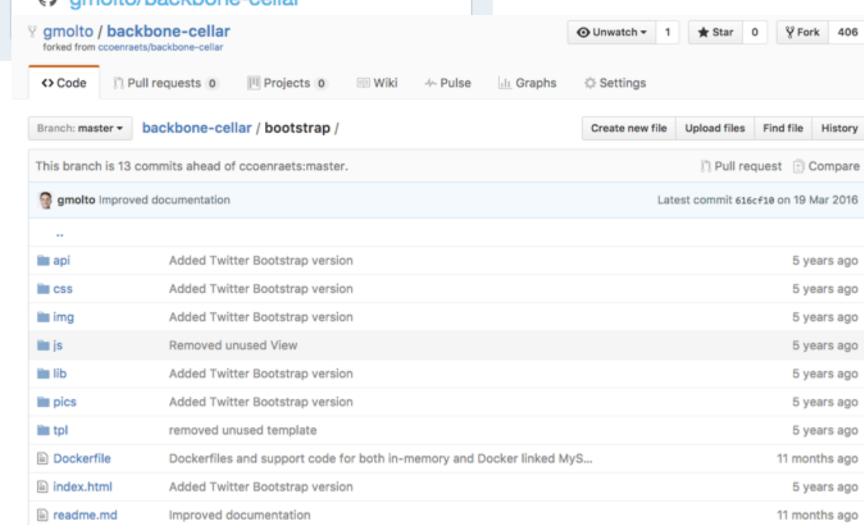


Docker 101: Image Building

- Option 1:
  - Modify a running container, exit the container and save the contents of the container as a new image that can be saved in a registry (own or DockerHub).
  - docker exec; docker commit; docker push
- Option 2:
  - Create the Docker container image from a Dockerfile, which contains a recipe for installing the application on a given OS.

Docker 101: Dockerfile (1)





- The Dockerfile is based on an existing image and describes the application installation process.
  - docker build -t cellar-mem .

Docker 101: Dockerfile (2)

- Dockerfile example to install [Infrastructure Manager]

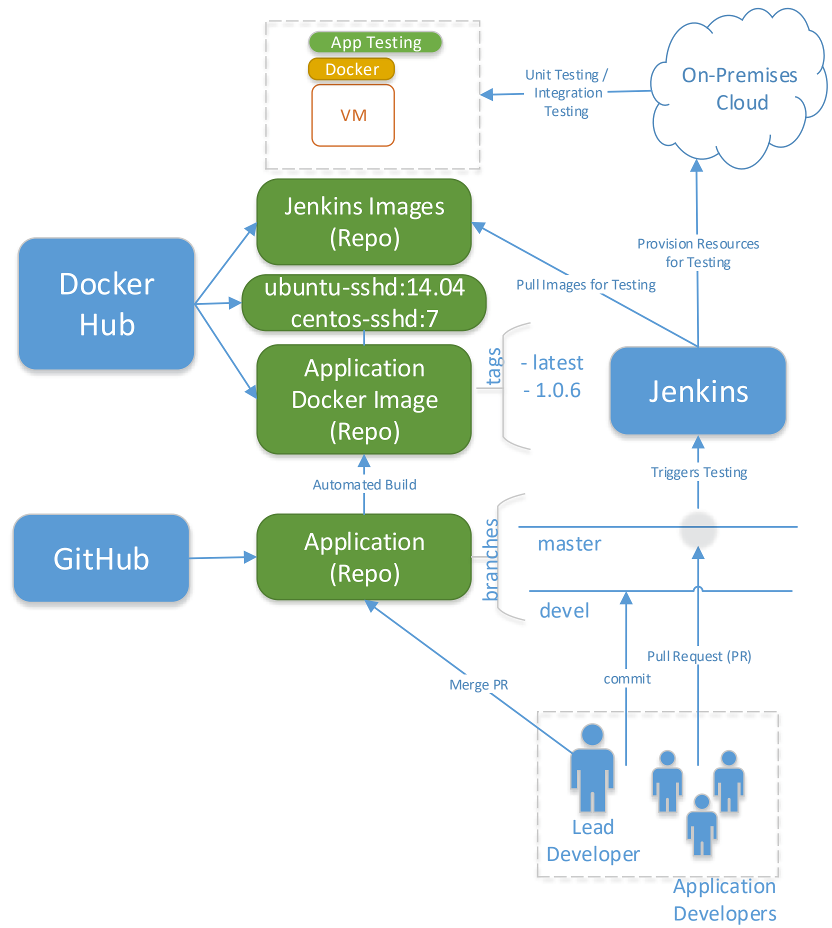
```
FROM ubuntu:22.04
MAINTAINER Miguel Caballer <micafer1@upv.es>
LABEL version="1.5.1"
LABEL description="Container image to run the IM service. (http://www.grycap.upv.es/im)"
EXPOSE 8899 8800
RUN apt-get update && apt-get install -y gcc python-dbg python-dev python-pip libmysqld-dev python-
pysqlite2 openssh-client sshpass libssl-dev libffi-dev python-requests
RUN pip install setuptools --upgrade -I
RUN pip install CherryPy==8.9.1
RUN pip install pyOpenSSL --upgrade -I
RUN pip install MySQL-python msrest msrestazure azure-common azure-mgmt-storage azure-mgmt-compute
azure-mgmt-network azure-mgmt-resource
RUN pip install IM
COPY ansible.cfg /etc/ansible/ansible.cfg
CMD im_service.py
```

https://github.com/grycap/im/blob/master/docker/Dockerfile

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

Usage Example: Docker-based CI

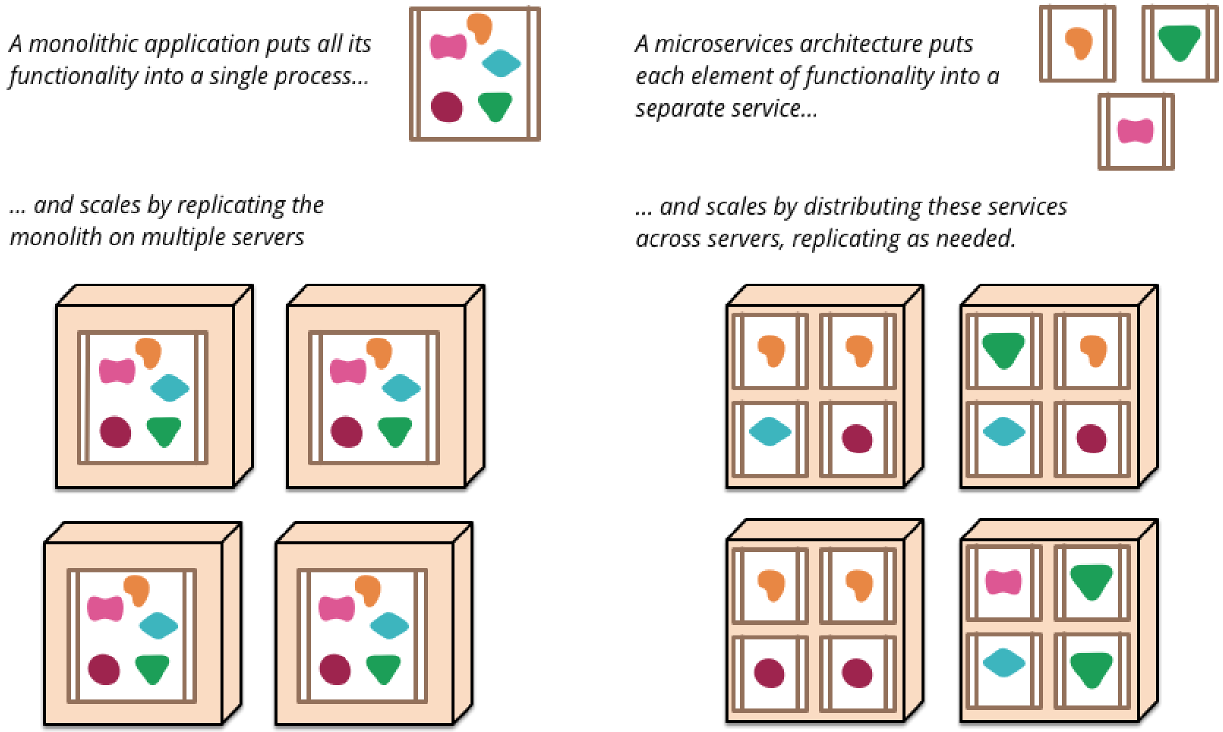
- Developers working on the devel branch of a GitHub repo.
- A PR on the master branch triggers the CI in Jenkins/Travis.
- Docker images in Docker Hub are used to execute the Jenkins jobs in the right execution env.
- Merging the PR into the master branch triggers an Automated Build to create a new Docker image in Docker Hub.



Microservices (I)

- Microservices is a software architecture pattern for designing applications as a set of deployable services independently.

- Services with a single function
- Decentralized accountability
- Multiple languages, libraries, etc.
- REST API + HTTP
- Stateless vs Stateful
- Independent updates by service.

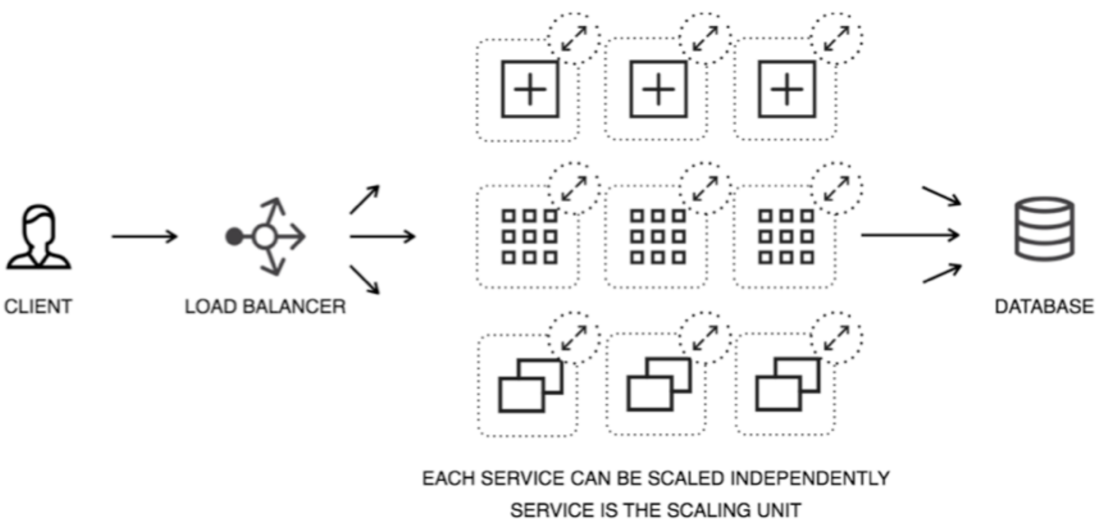


http://martinfowler.com/articles/microservices/images/sketch.png

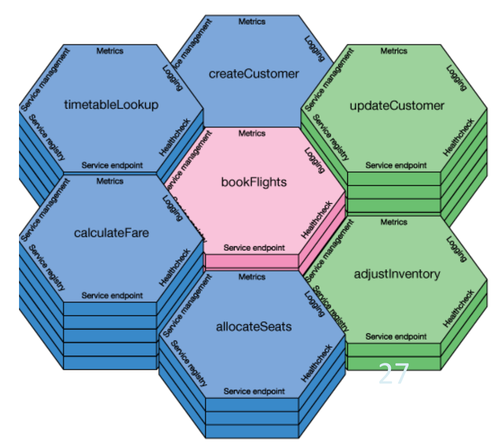
Microservices (II)

- Microservices-based architectures typically use:
  - Containers to encapsulate dependencies
  - CI/CD strategies for frequent updates.

The Microservices Architecture

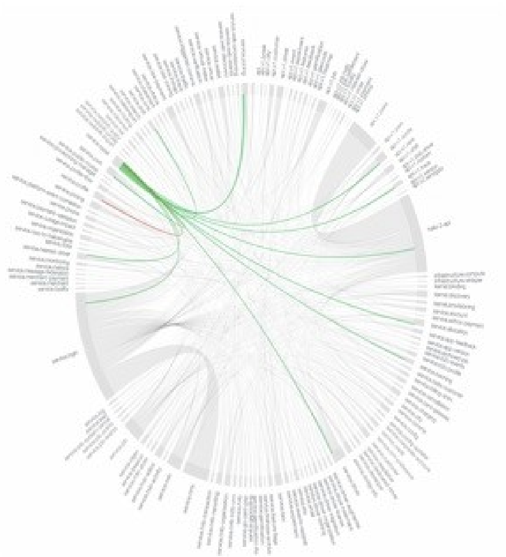


- Application as a set of containers that run microservices and can be scaled and updated.



Microservices Death Stars

450 microservices



500+ microservices

500+ microservices

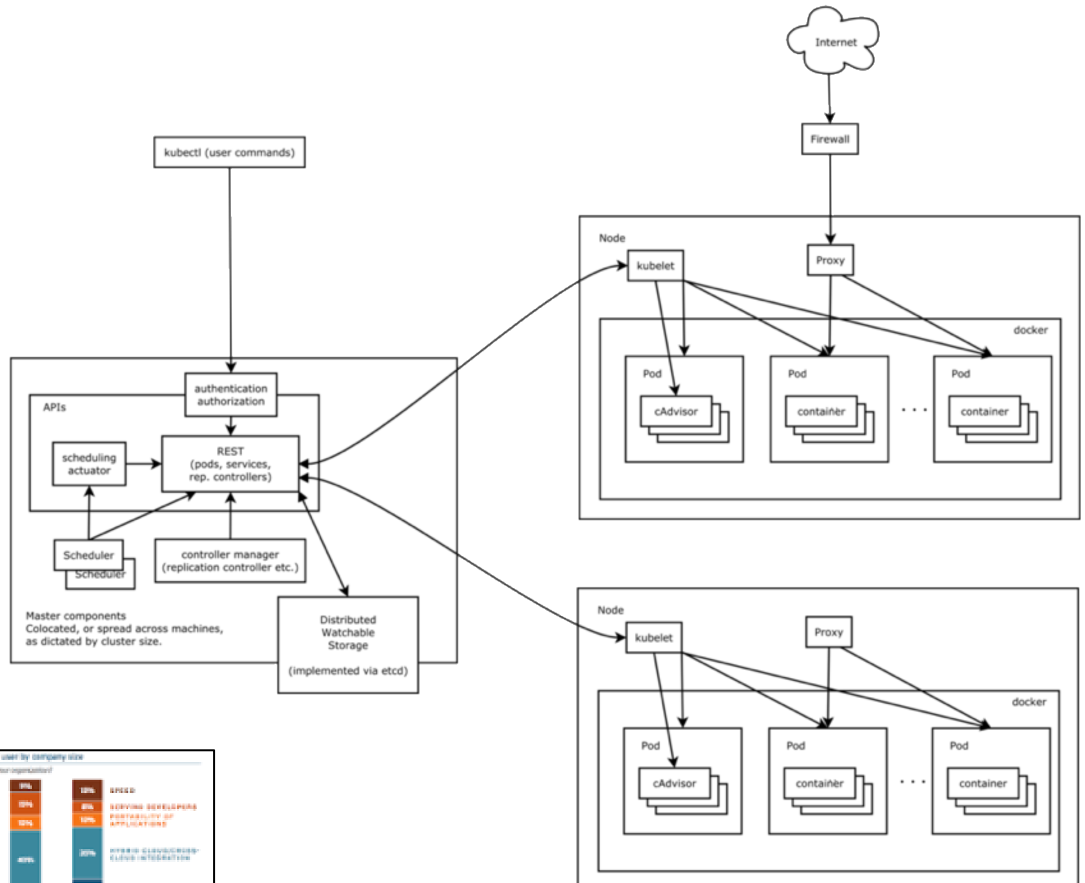
Source:
Netflix: http://www.slideshare.net/BruceWong3/the-case-for-chaos
Twitter: https://twitter.com/adrianco/status/441883572618948608

Container Management Platforms

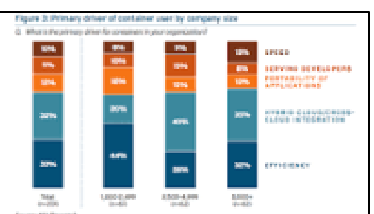
- Managing multiple containers requires the use of container management platforms.
- Main
  - Open source
    - Docker Swarm
    - **Kubernetes**
    - ~~Apache Mesos~~
      - Chronos, Marathon
    - Nomad
  - Managed
    - Amazon ECS
    - Amazon EKS
    - Azure Container Service
    - Google Cloud Run

Kubernetes

- Developed by Google and released as open source.
- Deploy, scale, and manage containerized applications.
- Higher learning curve but higher adoption



Kubernetes won the Container Orchestration **War** on 29 November 2017. On that day AWS announced their Elastic **Container** Service for **Kubernetes** (EKS). ... That's on top of Google and Azure offering managed **Kubernetes** services and OpenShift being based upon **Kubernetes**. 21 may. 2018



[Why Did Kubernetes Win? - DZone Cloud](https://dzone.com/articles/why-did-kubernetes-win)
https://dzone.com › articles › why-did-kubernetes-win

AI-SPRINT

AI SPRINT
AI IN SECURE PRIVACY-PRESERVING COMPUTING CONTINUUM

Serverless Computing

Germán Moltó

Departamento de Sistemas Informáticos y Computación - Instituto de Instrumentación para Imagen Molecular

gmolto@dsic.upv.es

https://www.grycap.upv.es/gmolto

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Grid y Computación de Altas Prestaciones GRyCAP

Escola Tècnica Superior d'Enginyeria Informàtica

etsinf

i3M

Cloud Services and Applications

- User cloud services and applications require management:
  - Data (i.e. status, in the form of files, databases, memory values, etc.)
  - Computing (resources and execution environments).
- Resilient/fault-tolerant application: Manage Replication and Distribution of both data and computing.

Object-Oriented Storage Systems

- Amazon S3 democratized access access to scalable, low-cost, long-term storage through simple APIs.
- AWS is responsible for capacity planning, storage provisioning, fault tolerance, and long-term durability through replication.

Amazon Simple Storage Service (S3)

Bucket with objects

Abstracting Computing

- Cloud computing (e.g. AWS) enabled the introduction of a virtualized representation of a classic datacenter.
  - Compute capacity provisioning and, sometimes (IaaS) configuration is required for application deployment.
- Isn't it possible to abstract the infrastructure further so that an application could run natively on top of the Cloud without needing to know the details of the underlying (virtualized) infrastructure?
  - Just like Amazon S3 does for storage
- Why not have something similar for computing?

AWS Lambda

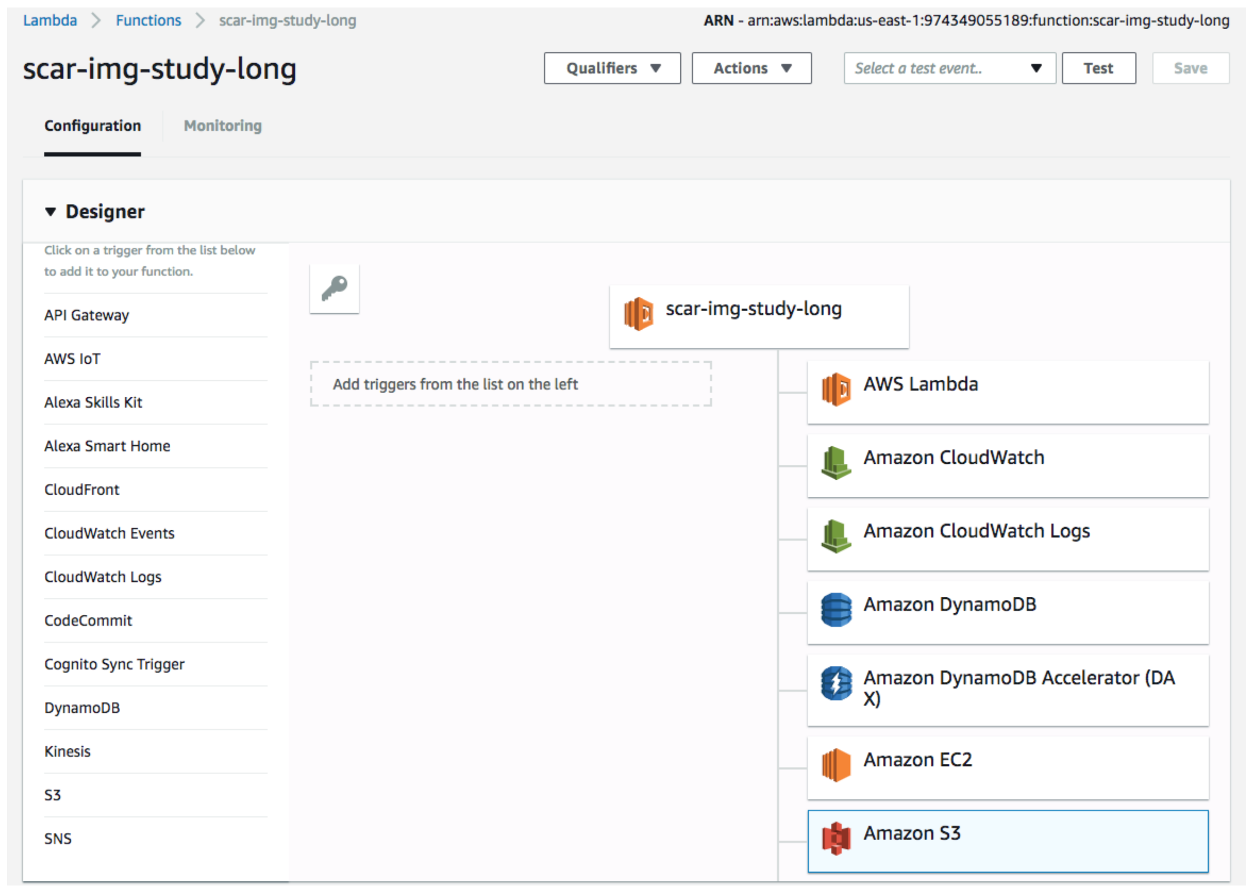
- AWS Lambda (https://aws.amazon.com/es/lambda/) allows you to run functions in response to events so that scaling is done automatically.
  - Stateless functions executed in micro-VMs with a maximum duration of 15 minutes, written in different programming languages (Node.JS, Python, Java, C#, Go).
  - Event: Invocation of REST API, file upload to S3, etc.
- Advantage: No dealing with ELBs, auto-scaling pools, EC2 instances, etc.
- Disadvantage: Requires redesigning the application.
- Price
  - In blocks of 1 ms. Pay per use (real). No costs when not in use. Free usage tier of 1M requests and 400,000 GB/second of computation per month per user.

About the AWS Lambda Runtime Environment (I)

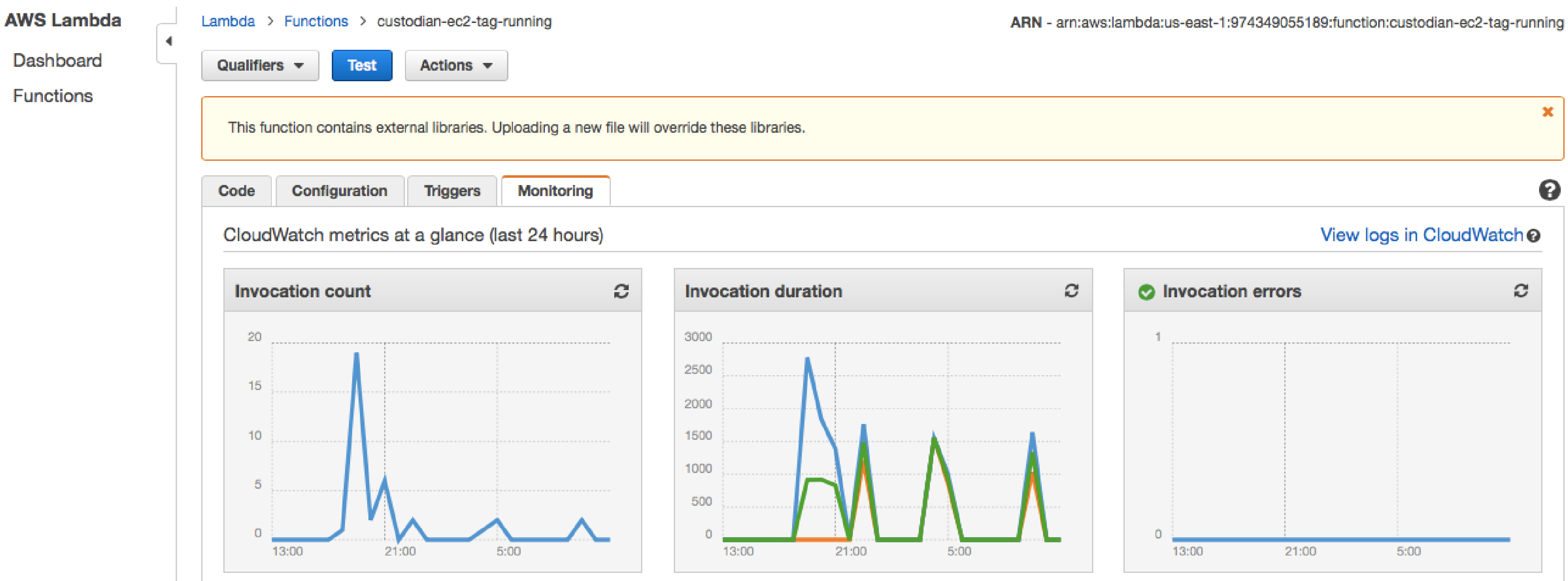
- AWS Lambda uses microVMs (Firecracker) with a series of predefined applications on top of which it executes the code of the Lambda functions
  - https://docs.aws.amazon.com/lambda/latest/dg/current-supported-versions.html
  - For testing, there is an AMI pre-configured that environment:
    - Node.js
    - Java
    - Python
    - .NET Core (C#)
- Restrictions
  - 3000 concurrent executions
  - [128, 10240] MB RAM (1 MB increments)
  - [512 , 10240] MB of non-persistent space* in /tmp
  - 15 minutes maximum runtime

AWS Lambda: Triggers

- Event Sources:
  - CloudWatch Events
  - S3
  - DynamoDB
  - Kinesis
  - SNS
  - API Gateway
  - …



AWS Lambda: Monitoring



- It allows to detect throttling problems:
  – Limit of 3000 concurrent executions per account and per region.

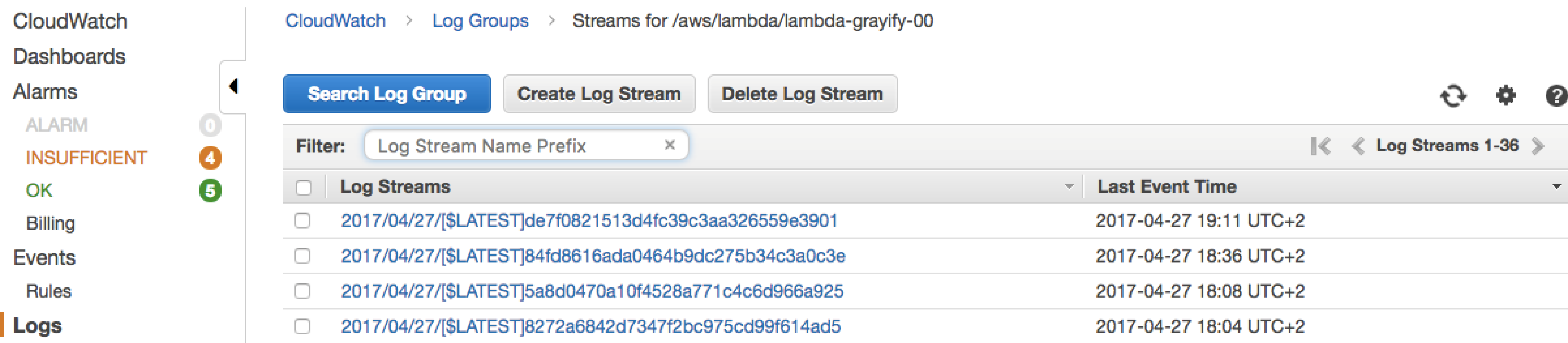
AWS Lambda: Execution Types

- There are two types of execution in AWS Lambda
  - https://docs.aws.amazon.com/es_es/lambda/latest/dg/API_Invoke.html
- RequestResponse
  - Synchronous invocation
- Async
  - Asynchronous invocation. It will be used for event processing.

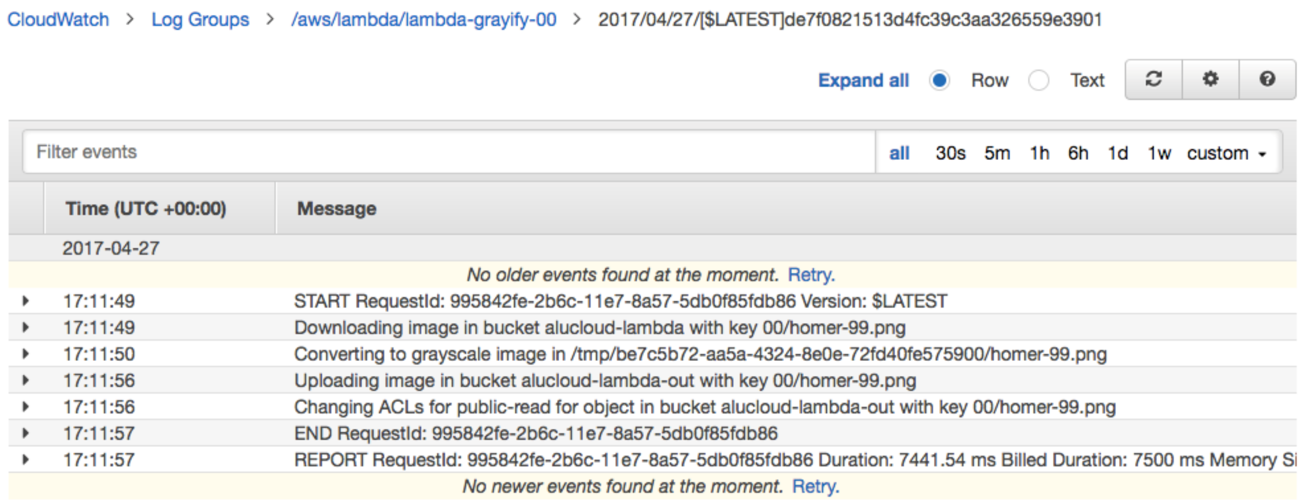
AWS Lambda: Throttling

- Throttling occurs when the maximum number of concurrent invocations (function or account-level) is exceeded
  - Synchronous invocation: Error HTTP 429
  - Asynchronous invocation: AWS Lambda automatically retries the event for up to 6 hours.

AWS Lambda: CloudWatch Logs

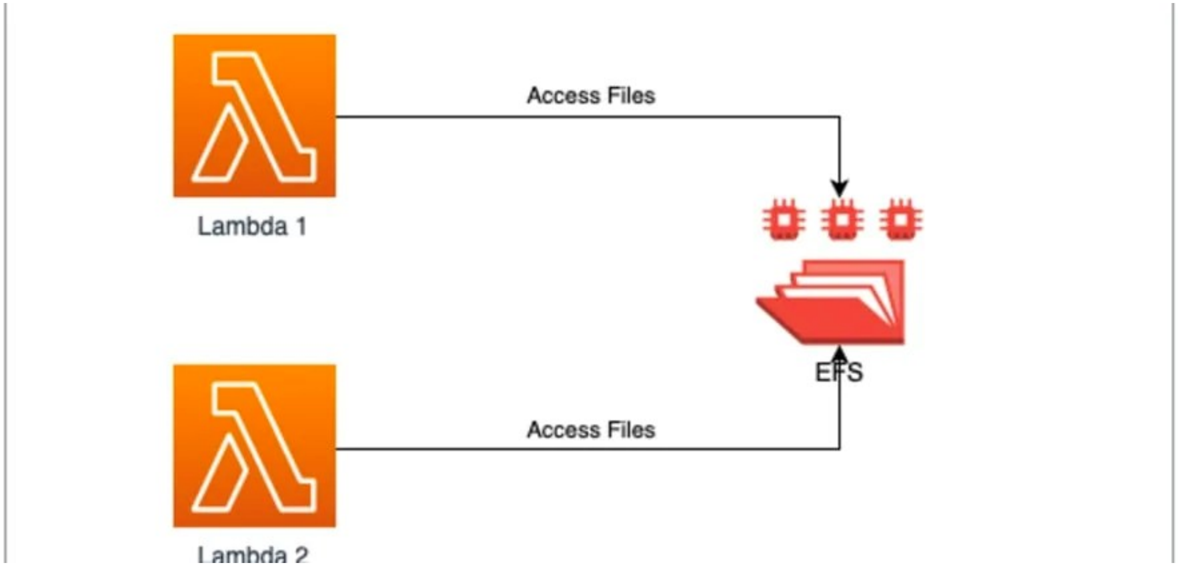


- Centralize, store, and search Lambda function log entries.



Stateful Lambda functions

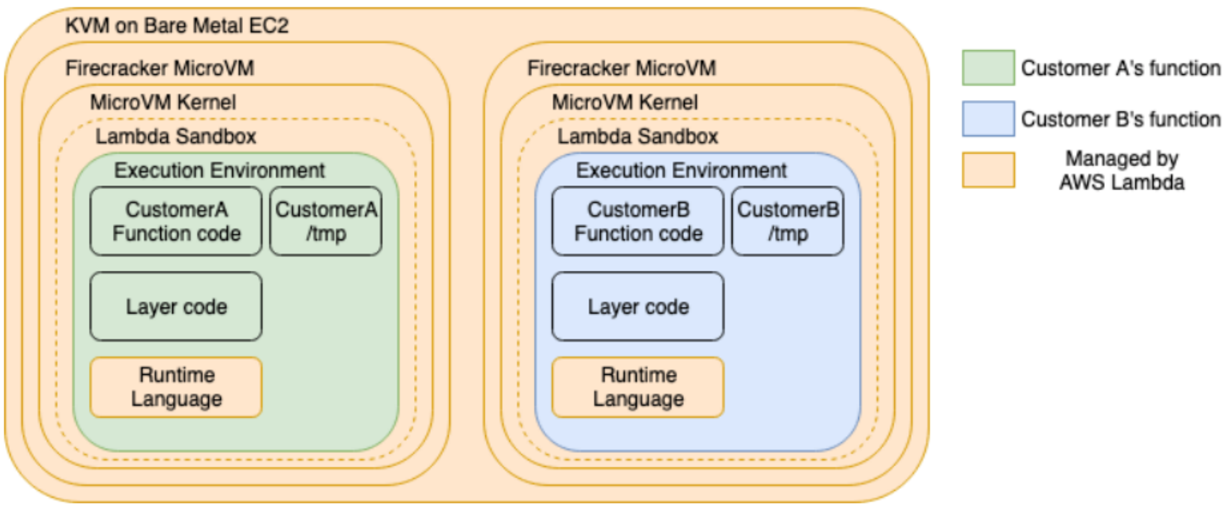
- AWS Lambda recently introduced support for Amazon EFS (NFS as a Service)
- Allows you to introduce persistence between Lambda functions.
- Scalable shared file system
- Serverless supercomputing.

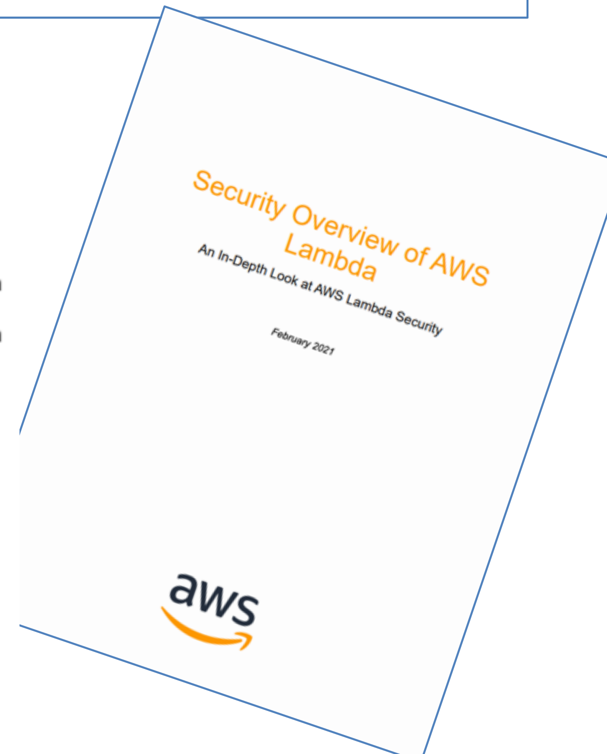


https://dev.to/imohd23/how-to-use-efs-with-aws-lambda-2057

- Potential use cases:
  - https://lumigo.io/blog/unlocking-more-serverless-use-cases-with-efs-and-lambda/

Peeking behind the curtains of AWS Lambda





FireCracker - https://firecracker-microvm.github.io/

https://d1.awsstatic.com/whitepapers/Overview-AWS-Lambda-Security.pdf

ARTICLE

Peeking behind the curtains of serverless platforms

Authors: Liang Wang, Mengyuan Li, Yinqian Zhang, Thomas Ristenpart, Michael Swift Authors Info & Affiliations

Publication: USENIX ATC '18: Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference • July 2018 • Pages 133–145

https://dl.acm.org/doi/10.5555/3277355.3277369

Serverless Application

- Combine serverless services to produce applications that have a very low TCO (Total Cost of Ownership).
-

Storage - Amazon S3

Compute - AWS Lambda

Database - Amazon DynamoDB

Gateways - Amazon API Gateway

Queues - Amazon SQS

Messaging - Amazon SNS

Internet of Things - AWS IoT

Streaming Analytics - Amazon Kinesis

User Management - Amazon Cognito

Machine Learning - Amazon Machine Learning

- Web application
- Alexa Skill
- Chatbot
- AWS IoT Button
- …

Exposing Functions to the Internet

https://es.slideshare.net/AmazonWebServices/building-apis-with-amazon-api-gateway

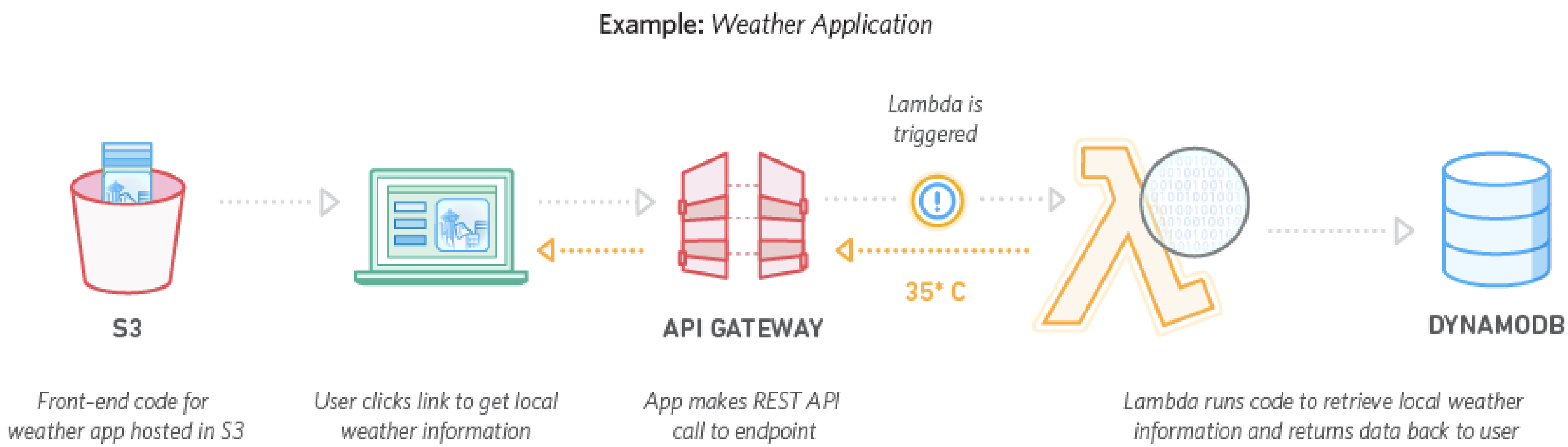
- Use an API to be able to invoke the function remotely. Challenges in:
  - Manage multiple API versions
  - Access authorization
  - Increases in invocation traffic

API Gateway

- API Gateway - https://aws.amazon.com/es/api-gateway/
  - Creation, publication, maintenance, monitoring, protection of APIs at any scale.
- Allows
  - Create a unified API for multiple microservices.
  - Protection against DDoS and throttling attacks to avoid back-end problems
  - AuthZ/AuthN requests using Cognito (and Lambda)
- Limit
  - 29 seconds maximum invocation time

Serverless Architecture Examples

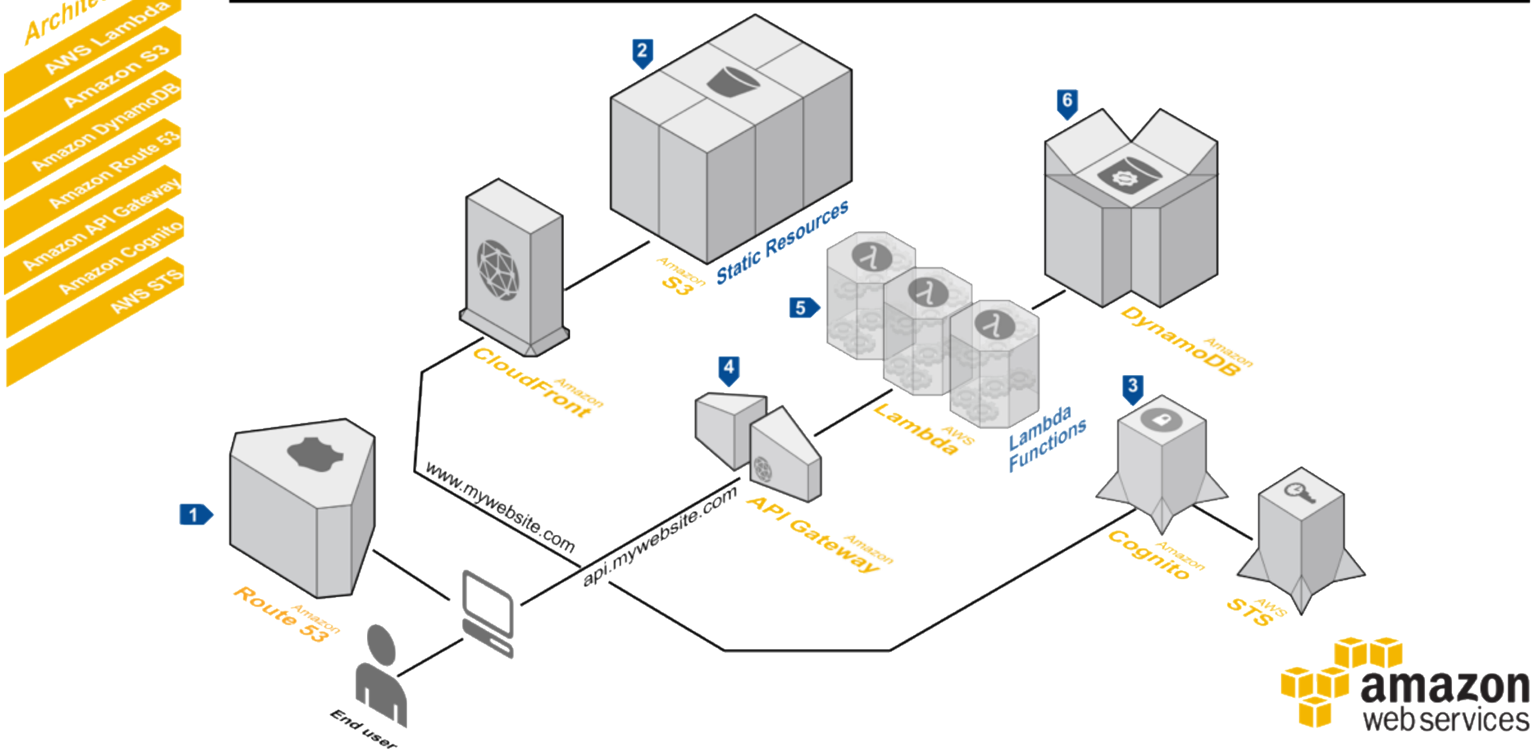
- Web application to obtain weather information stored in DynamoDB offering a REST API created with API Gateway.
  - The Lambda function is executed by invoking API methods (GET, POST, etc.)



Serverless Web Application Architecture

Serverless Blog Web Application Architecture

Sometimes developers want to just build their application. You don't want to deal with infrastructure or scaling. With AWS, you can build a scalable, highly available, sophisticated web application with zero servers to maintain. Serverless web applications provide an even faster time to market for your product.



System Overview

1 The user's DNS requests are served by **Amazon Route 53**, a highly available Domain Name System (DNS) service. Network traffic is routed to infrastructure running in Amazon Web Services.

2 Static content is delivered by **Amazon CloudFront,** a global network of edge locations. Requests are automatically routed to the nearest edge location. The static resources and content are stored in **Amazon Simple Storage Service (S3)**, a highly durable storage infrastructure designed for mission-critical and primary data storage. **S3** serves the static content for the website such as HTML, CSS, and Javascript files.

3 A user first authenticates (either using **Cognito User Pools**, an external identity provider, or your own custom developed authentication system), the client calls out to **Amazon Cognito** in order to obtain temporary credentials to call **Amazon API Gateway**. **Cognito** retrieves credentials from **AWS STS** to pass back to the user.

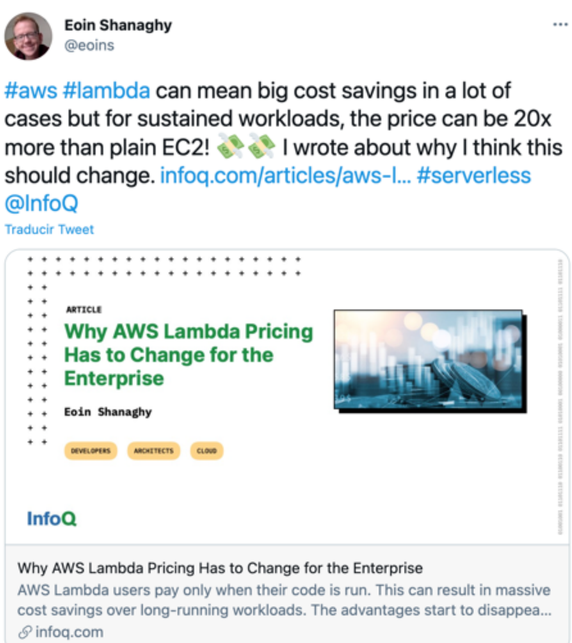
4 API requests are signed using the temporary credentials obtained from Amazon Cognito and sent to the **Amazon API Gateway** service which provides features such as security and throttling for your APIs. Requests are passed through the **API Gateway** where they canbe transformed and passed through to the backend service logic.

5 **AWS Lambda** provides the backend business logic for your web application. You do not run servers, but instead upload code to AWS which will be invoked when a request comes into your API. The service is highly scalable and removes the need to manage infrastructure.

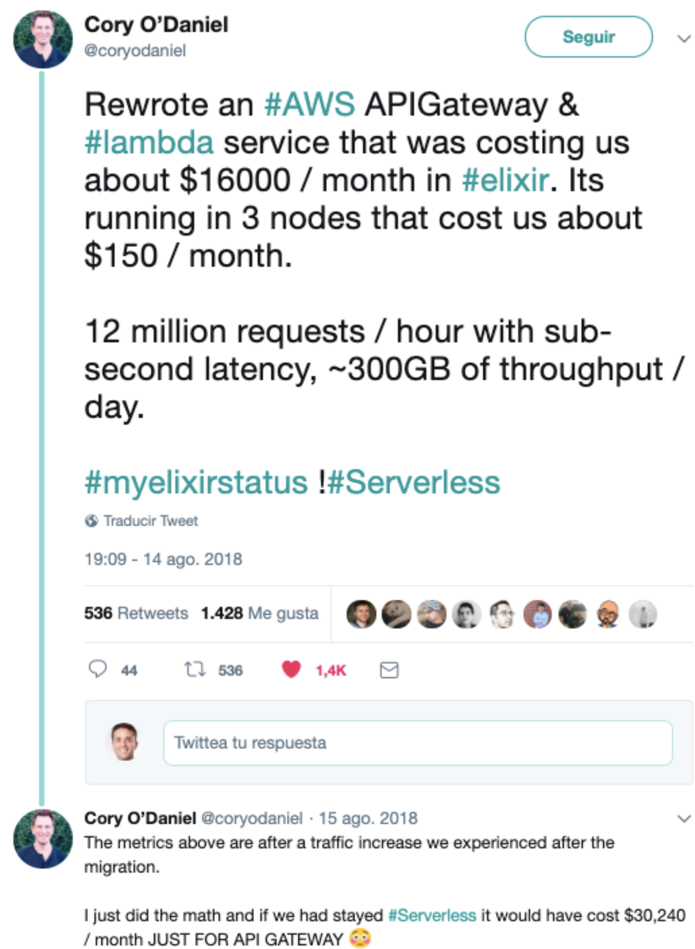
6 As a fully managed database solution, **Amazon DynamoDB** provides fast, consistent performance as the data layer for your web applications.

Beware of the costs

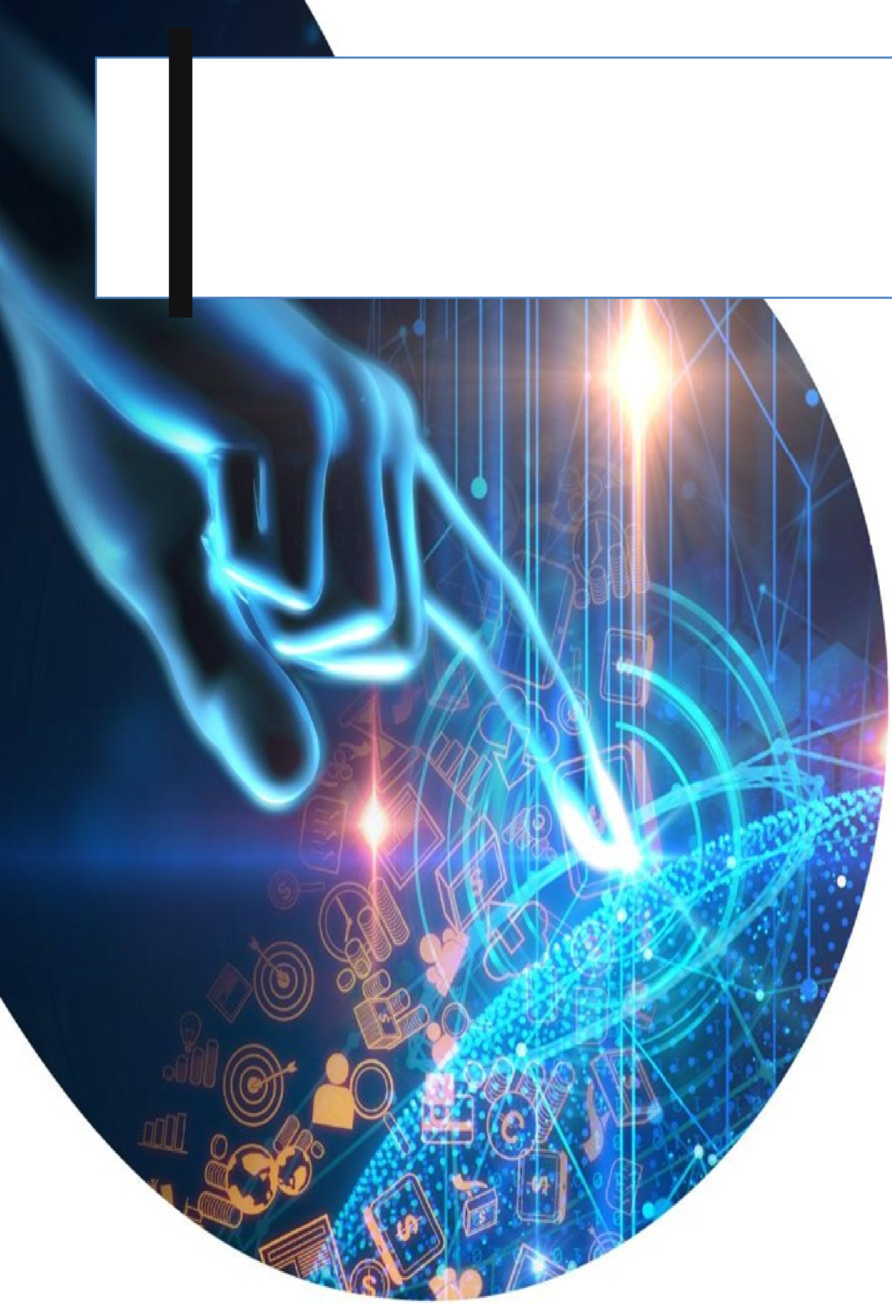
- For high service usage rates it may be interesting to use a traditional VM-based architecture
- Price reduction at the cost of reducing elasticity.



https://twitter.com/eoins/status/1381643758031273988



https://twitter.com/coryodaniel/status/1029414668681469952

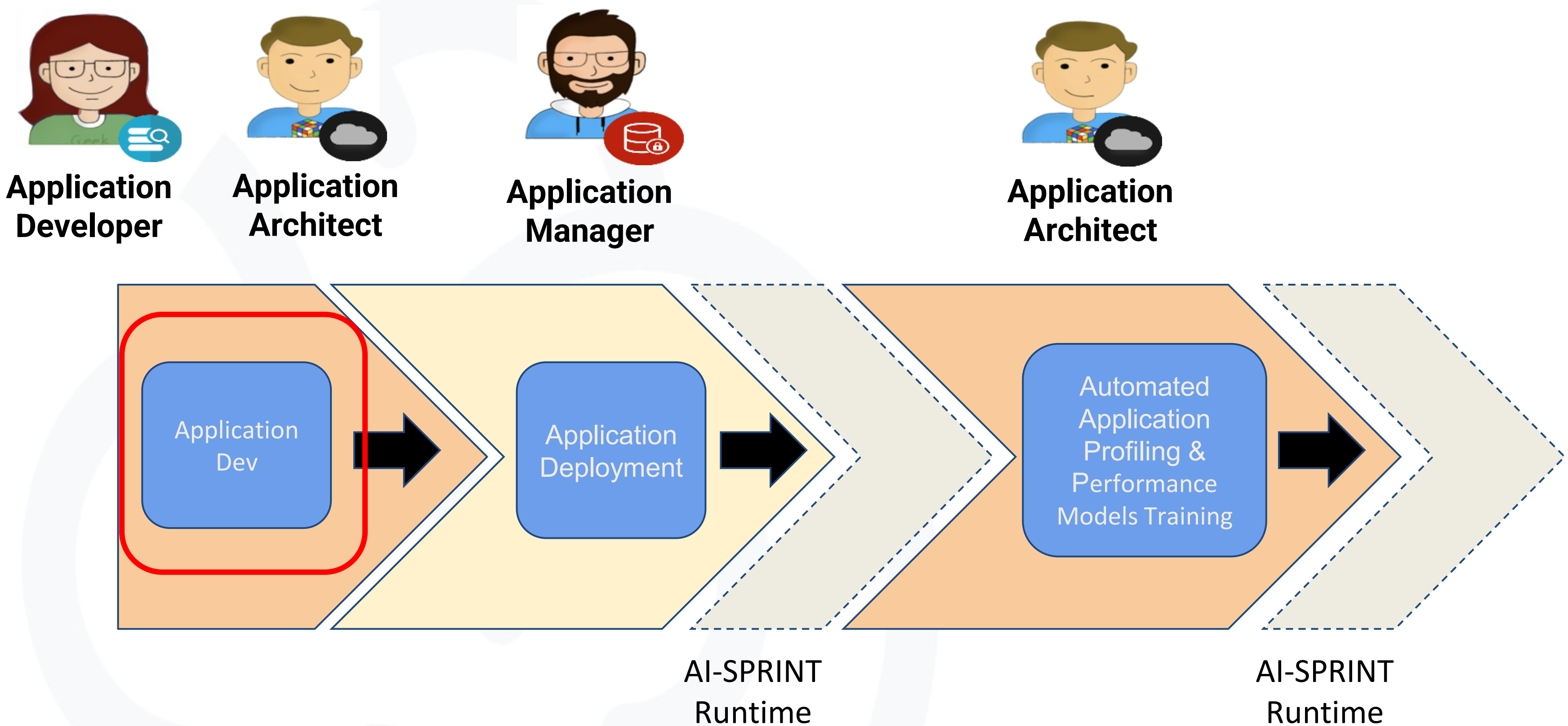


AI-SPRINT Design

Francesco Lattari

francesco.lattari@polimi.it

AI Application Design Workflow



AI-SPRINT Design

Problem solved:

- **Automatic generation of configuration files** for AI-SPRINT design and runtime tools
- **High-level abstractions** to constraint the application deployment
- **Advanced functionalities** to:
  - **Configure monitoring files**
  - Provide **automatic partitioning of AI models**
  - Provide **automatic generation of alternative deployments**
  - Generate AI-SPRINT Drift Detector component

Motivations:

- **Standardize the architectures of the applications**
- **Provide a simple interface** between the user and the AI-SPRINT framework
- Need for **advanced functionalities driven by AI requirements**

AI-SPRINT Design main contributions

Without AI-SPRINT

- The developer of the AI application must prepare the application organizing the files in a non-standard non-portable structure
- The developer must explicitly define each possible deployment
- No easy ways to define QoS constraints to drive the deployment
- The user must involve an AI expert to provide partitioned models and to manage them
- The user is not able to easily design detection algorithms for detecting data drift at runtime

AI-SPRINT Design main contributions

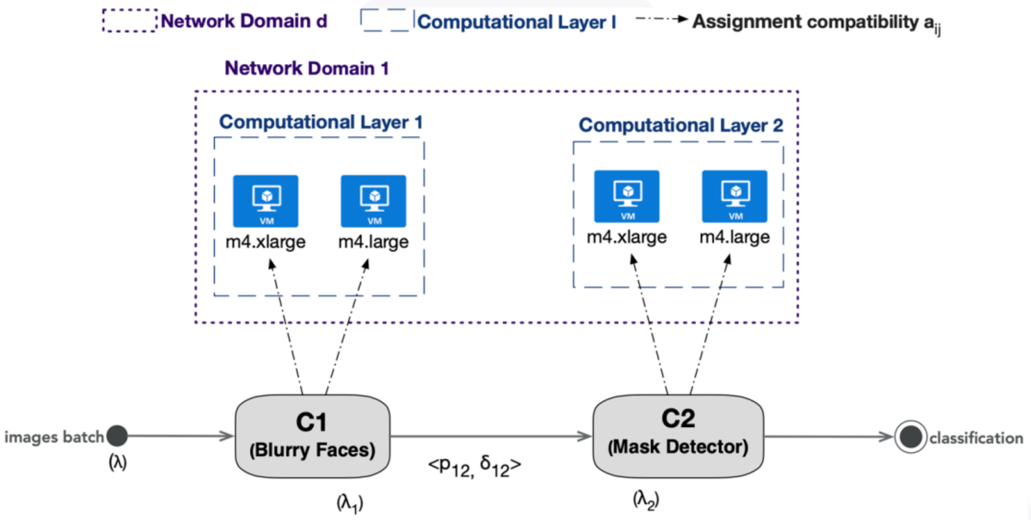
Without AI-SPRINT

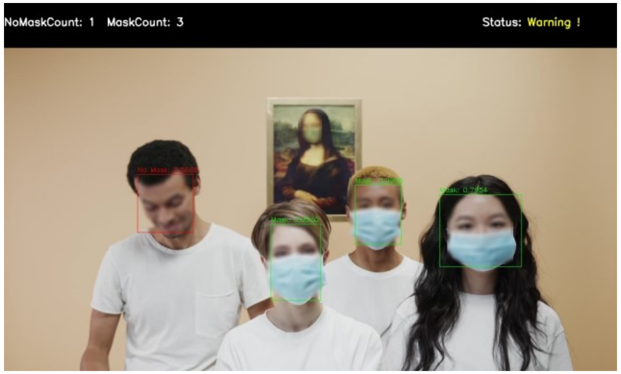
- The developer of the AI application must prepare the application organizing the files in a non-standard non-portable structure
- The developer must explicitly define each possible deployment
- No easy ways to define QoS constraints to drive the deployment
- The user must involve an AI expert to provide partitioned models and to manage them
- The user is not able to easily design detection algorithms for detecting data drift at runtime

With AI-SPRINT

- Simple interface to provide application components and workflow following a well-defined template
- The application architect easily defines the available resources
- Alternative deployments are automatically generated
- High-level abstractions to define QoS constraints
- SPACE4AI-D Partitioner allows automatic partitioning of neural networks
- Automatic design of runtime drift detection algorithms can be enabled

Mask detection application





Local execution time constraint:

Blurry Faces: 15 s

Global execution time constraint:

Blurry Faces + Mask Detector : 20 s

AI-SPRINT Design Input Files

Input Files

```
mask_detection_app/
├── aisprint
│   ├── deployments
│   ├── designs
│   └── logs
├── ams
├── common_config
│   ├── application_dag.yaml
│   ├── candidate_deployments.yaml
│   └── candidate_resources.yaml
├── im
├── oscar
├── oscarp
├── pycompss
├── space4ai-d
│   └── SPACE4AI-D.yaml
├── space4ai-r
└── src
    ├── blurry-faces-onnx
    │   ├── main.py
    │   ├── onnx
    │   │   └── version-RFB-640.onnx
    │   ├── requirements.sys
    │   ├── requirements.txt
    │   └── utils.py
    └── mask-detector-onnx
        ├── cfg
        │   └── obj.names
        ├── main.py
        ├── onnx
        │   └── yolov3-tiny.onnx
        ├── requirements.sys
        └── requirements.txt
```

AI-SPRINT Design Demo: Application Preparation and Design

Demo steps:

1. Generate new AI-SPRINT application named *mask_detection_app* using the available Docker image and application template. AI-SPRINT Design is available as part of the **AI-SPRINT Studio**.

1. Add components' implementation, application DAG and candidate resources files to the application project

1. Run AI-SPRINT Design

1. Inspect the generated files

AI-SPRINT Design Result

Output Files

```
mask_detection_app/
├── aisprint
│   ├── deployments
│   │   ├── base
│   │   │   ├── ams
│   │   │   │   ├── qos_constraints_L1.yaml
│   │   │   │   └── qos_constraints_L2.yaml
│   │   │   ├── application_dag.yaml
│   │   │   ├── im
│   │   │   ├── oscar
│   │   │   ├── oscarp
│   │   │   ├── pycompss
│   │   │   ├── space4ai-d
│   │   │   │   ├── qos_constraints.yaml
│   │   │   │   └── SPACE4AI-D.yaml
│   │   │   ├── space4ai-r
│   │   │   └── src
│   │   │       ├── blurry-faces-onnx -> ../../../designs/blurry-faces-onnx/base
│   │   │       └── mask-detector-onnx -> ../../../designs/mask-detector-onnx/base
│   │   ├── multi_cluster_qos_constraints.yaml
│   │   └── optimal_deployment
│   │       └── production_deployment.yaml
│   ├── designs
│   │   ├── blurry-faces-onnx
│   │   │   └── base
│   │   │       ├── main.py
│   │   │       ├── onnx
│   │   │       │   └── version-RFB-640.onnx
│   │   │       ├── requirements.sys
│   │   │       ├── requirements.txt
│   │   │       └── utils.py
│   │   ├── component_partitions.yaml
│   │   └── mask-detector-onnx
│   │       └── base
│   │           ├── cfg
│   │           │   └── obj.names
│   │           ├── main.py
│   │           ├── onnx
│   │           │   └── yolov3-tiny.onnx
│   │           ├── requirements.sys
│   │           └── requirements.txt
│   └── logs
├── ams
├── common_config
│   ├── annotations.yaml
│   ├── application_dag.yaml
│   ├── candidate_deployments.yaml
│   └── candidate_resources.yaml
├── im
├── oscar
├── oscarp
├── pycompss
├── space4ai-d
│   ├── qos_constraints.yaml
│   └── SPACE4AI-D.yaml
├── space4ai-r
└── src
    ├── blurry-faces-onnx
    │   ├── main.py
    │   ├── onnx
    │   │   └── version-RFB-640.onnx
    │   ├── requirements.sys
    │   ├── requirements.txt
    │   └── utils.py
    └── mask-detector-onnx
        ├── cfg
        │   └── obj.names
        ├── main.py
        ├── onnx
        │   └── yolov3-tiny.onnx
        ├── requirements.sys
        └── requirements.txt
```

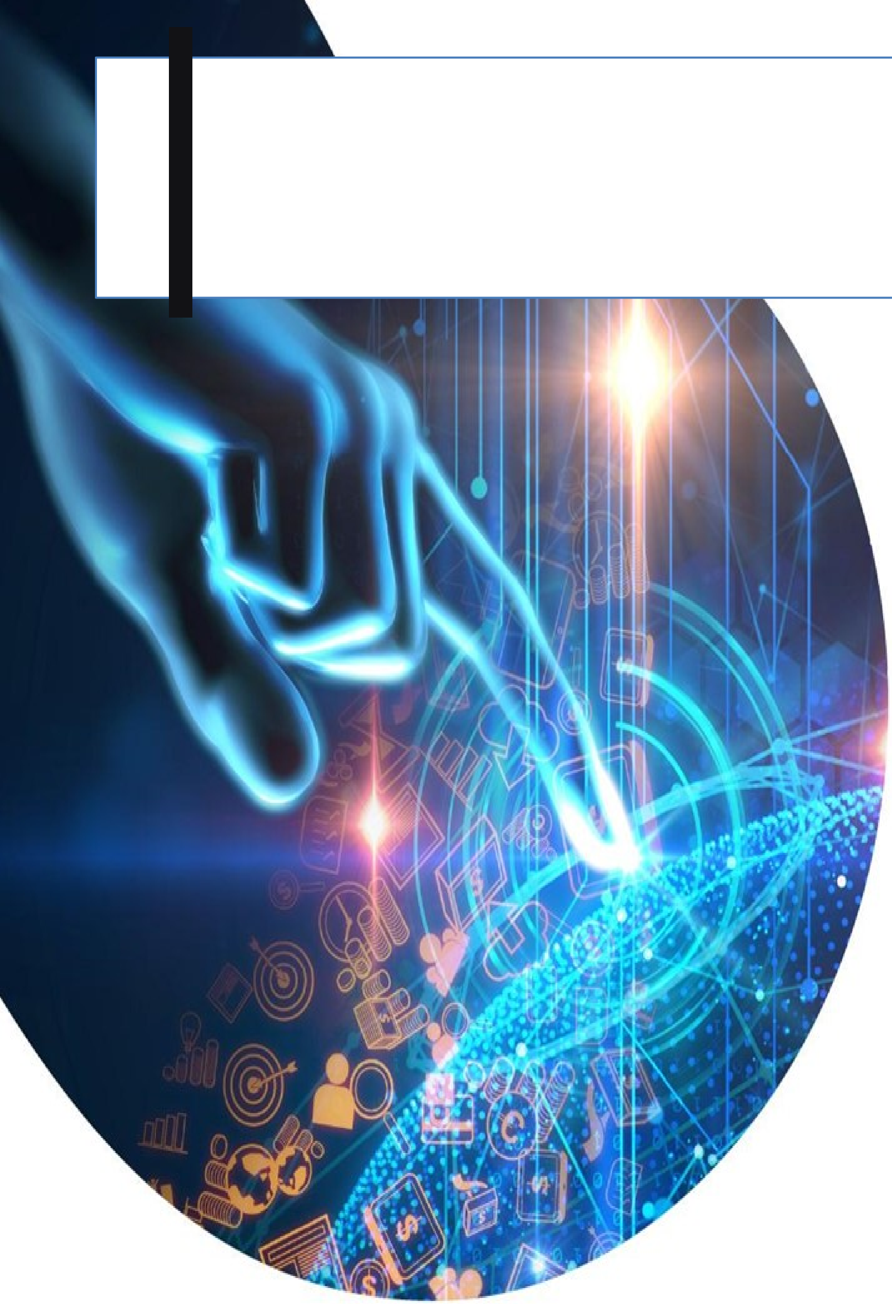
References & Links

Links:

GitLab repository for AI-SPRINT Studio: https://gitlab.polimi.it/ai-sprint/ai-sprint-studio

Docker container with AI-SPRINT Studio: registry.gitlab.polimi.it/ai-sprint/ai-sprint-studio

Link to source project files: https://gitlab.polimi.it/ai-sprint/ai-sprint-examples/-/tree/main/mask_detection_local_global_constraints

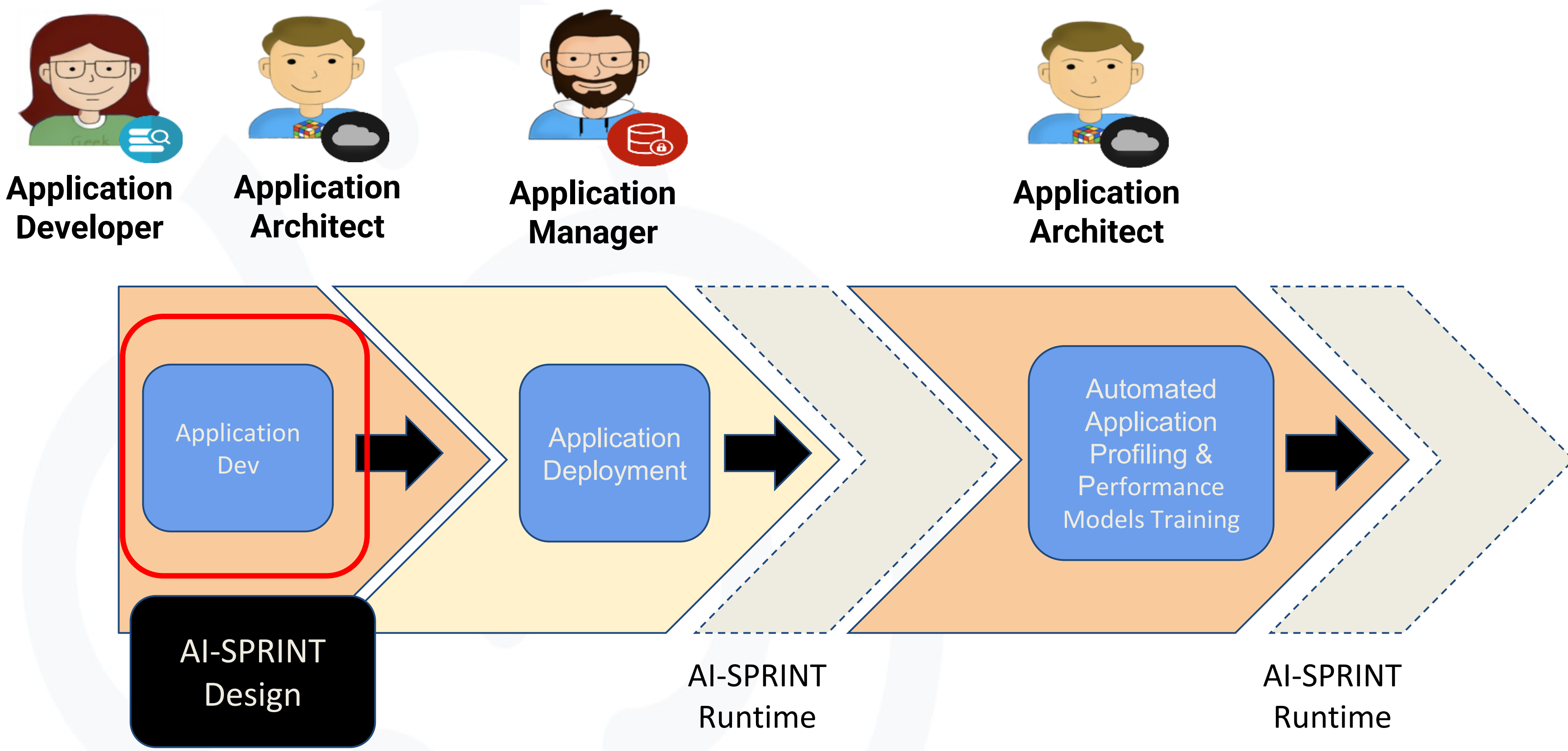


TOSCARIZER

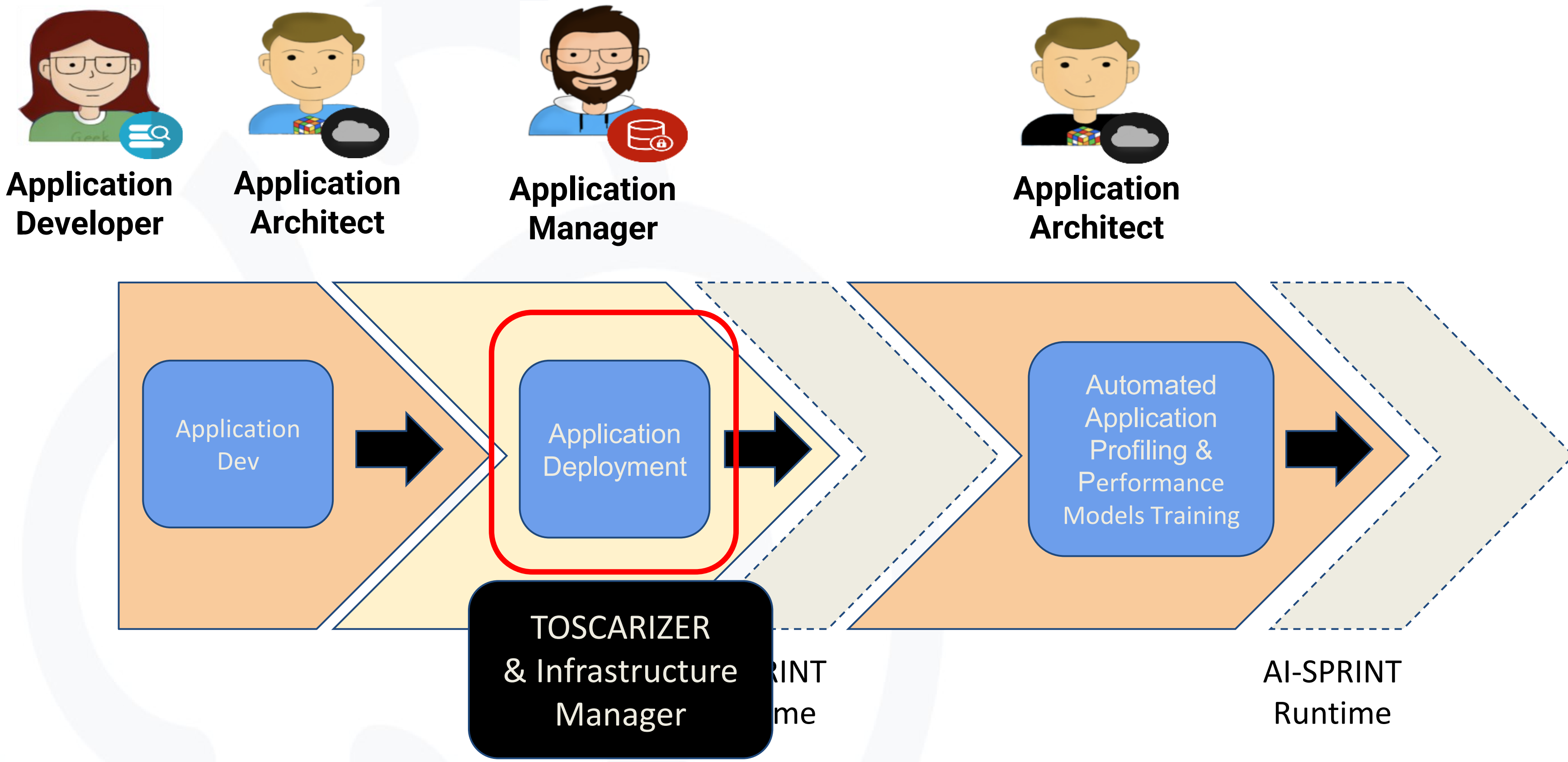
Miguel Caballer

micafer1@upv.es

AI Application Design Workflow



AI Application Design Workflow



Application Deployment

Problem solved:

- Help Application Manager to **generate component container images**.
- **Generate TOSCA templates** with the full description of the virtual infrastructures required by the application
- **Deployment of all application** components along the computing continuum

Motivations:

- **Provisioning and configuring** complex virtual infrastructures **is a complex task** due to the multiple API and configurations involved

TOSCARIZER main contributions

Without AI-SPRINT

- App. M. has to manually create Dockerfiles and build/push them for all the required architectures (AMD64, ARM64)
- App. M. has to manually create TOSCA templates to enable the deployment of application components that requires advanced knowledge on TOSCA standard.
- App. Manager has to access multiple Cloud back-ends with different interfaces (e.g. CLI, GUI, API).

TOSCARIZER main contributions

Without AI-SPRINT

- App. M. has to manually create Dockerfiles and build/push them for all the required architectures (AMD64, ARM64)
- App. M. has to manually create TOSCA templates to enable the deployment of application components that requires advanced knowledge on TOSCA standard.
- App. Manager has to access multiple Cloud back-ends with different interfaces (e.g. CLI, GUI, API).

With AI-SPRINT

- Container images are automatically built and pushed for all components only for the needed architectures used in the deployments.
- TOSCA templates are automatically generated for all the components with the exact requirements specified by the application developed in the application description files.
- Automated deployment/Undeployment of OSCAR services on pre-provisioned OSCAR clusters at the Edge of the network and provision whole OSCAR clusters on the available Cloud back-ends

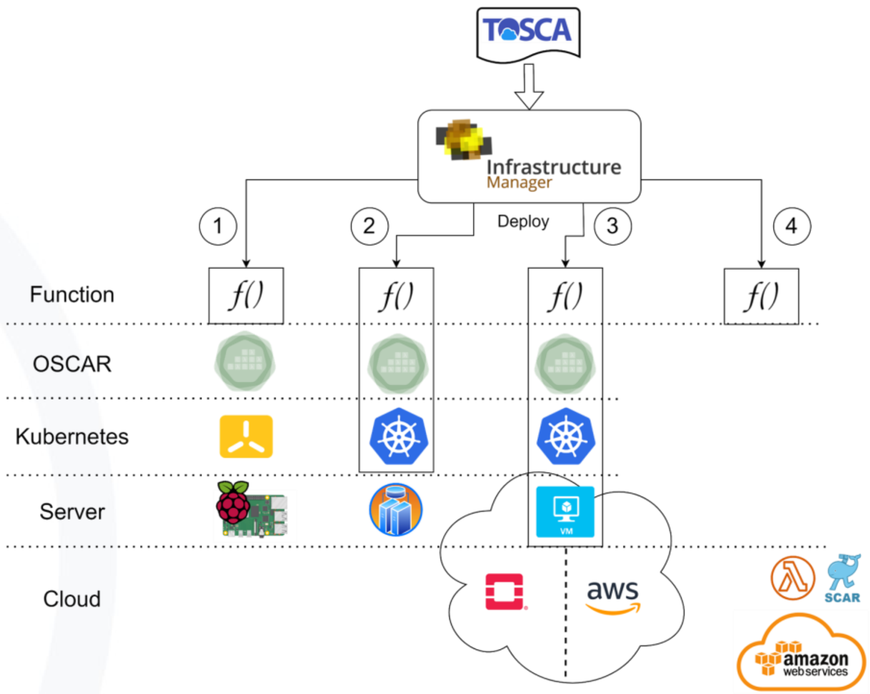
TOSCARIZER

- **New developed component** to help Application Manager to deploy Inference services.
  - **Creates the Docker images** for all application components considering all possible destination architectures (AMD64 and ARM64).
  - **Creates TOSCA templates** to deploy, not only the inference services on top of OSCAR clusters, but also all the needed underlying cloud infrastructure (VMs, K8s cluster, OSCAR …).
    - Deploys the full application workflow.
  - **Interacts with the IM** to finally deploy/undeploy all the inference infrastructure.

Cloud Continuum Support

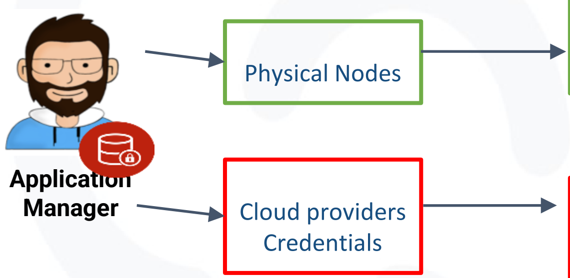
Type of deployments:

- (1) **Edge device**:
  - Only deploy OSCAR service on top of an existing cluster.
- (2) **Edge Node/Edge Device**:
  - Accessed via SSH.
  - K8s + OSCAR + OSCAR service.
- (3) **Cloud** (On-premises/Public)
  - Deploy VM + K8s + OSCAR + OSCAR service.
- (4) **AWS Lambda**
  - Deploy FaaS function.
  - Using SCAR.



Application and Infrastructure deployment

- It takes as input the output of the AI-SPRINT design tool +
  - Physical nodes:
    - MinIO credentials
      - In case of edge device
    - SSH credentials
      - In case of edge node
    - AWS S3 info
      - In case of Lambda
  - IM auth file:
    - Cloud Credentials



common_config/physical_nodes.yaml


```yaml
ComputationalLayers:
   computationalLayer1:
      number: 1
      Resources:
         resource1:
            name: RaspPi
            minio:
               endpoint: https://minio.oscar.net
               access_key: minio
               secret_key: pass
            oscar:
               name: oscar-test
```

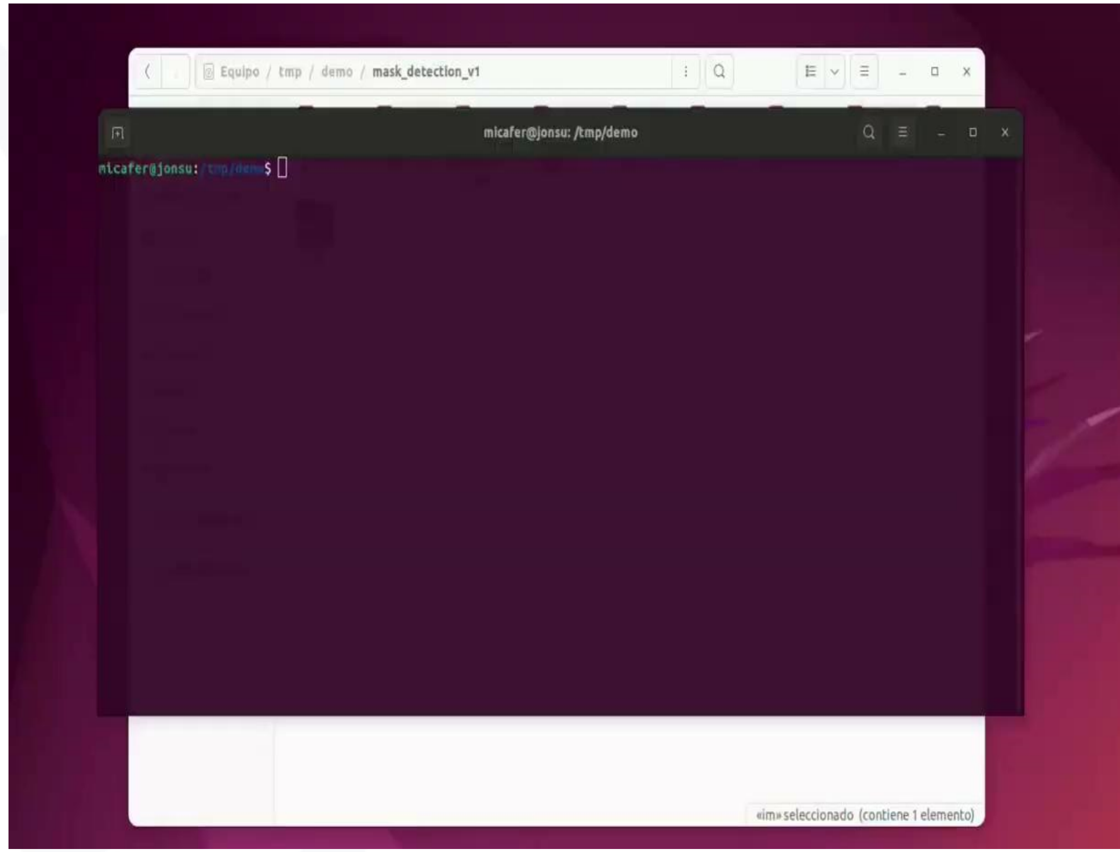

im/auth.dat


```
id = one; type = OpenNebula; host = server:2633; username = user; password = pass
id = oscar1; type = OSCAR; host = https://oscar.net; username = user; password = pass
type = InfrastructureManager; username = user; password = pass
type = EC2; username = AK; password = SK
```

Application and Infrastructure deployment

Demo steps:

1. **Build & push the Docker images** for all the components / partitions
2. **Create** the corresponding **TOSCA files** to deploy all the application components (base or optimal cases)
3. **Perform the deployment** through the IM
4. Test the application workflow
5. Undeploy infrastructures.



Refere[redacted]ces & Links

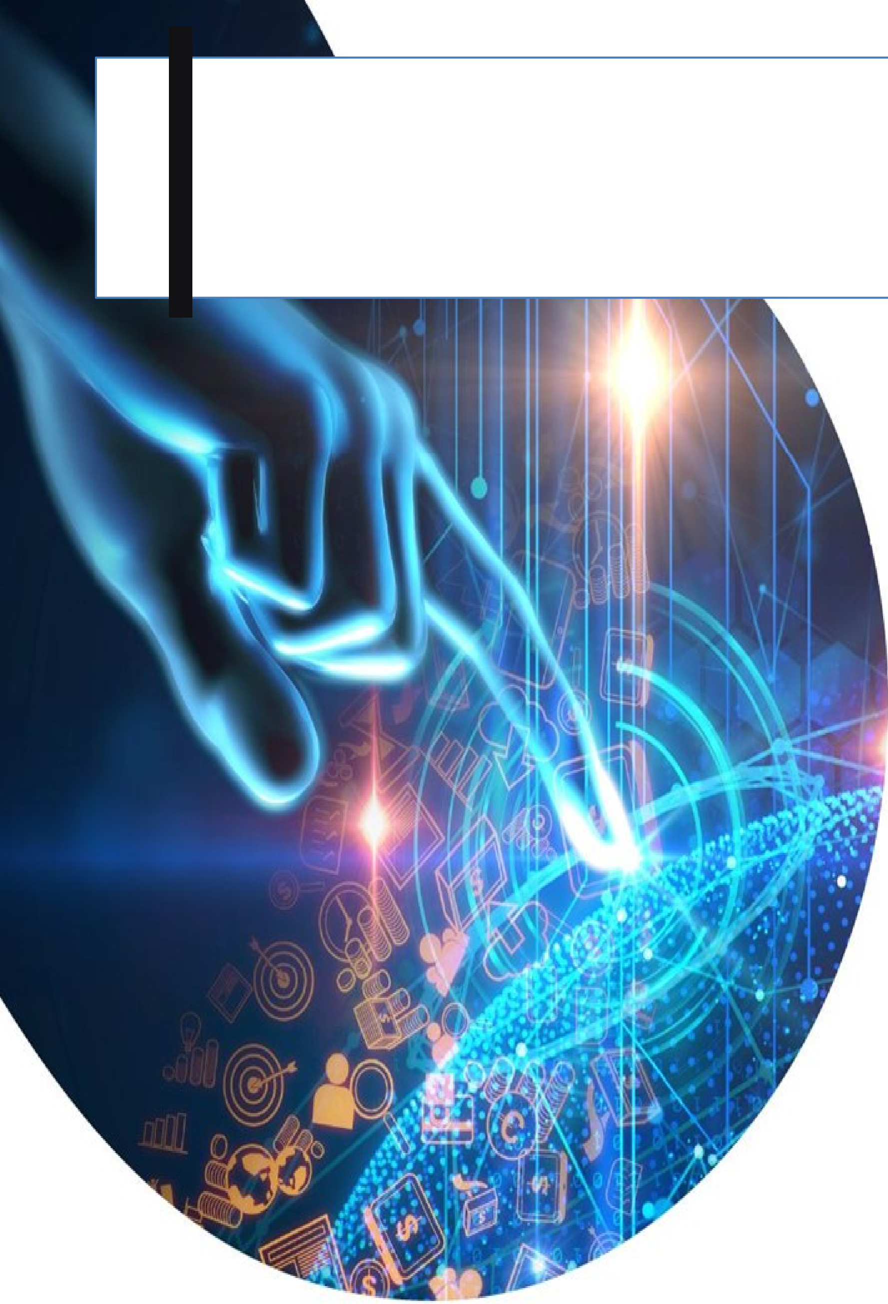
References:

[1] Miguel Caballer, Germán Moltó, Amanda Calatrava, and Ignacio Blanquer. Infrastructure Manager: A TOSCA-Based Orchestrator for the Computing Continuum. Journal of Grid Computing, 21:51, 9 2023. https://link.springer.com/article/10.1007/s10723-023-09686-7

Links:

GitLab repository: https://gitlab.polimi.it/ai-sprint/toscarizer

Integrated in docker AI-SPRINT Studio container: registry.gitlab.polimi.it/ai-sprint/ai-sprint-studio



Programming Distributed Computing Platforms with COMPSs and dislib

Daniele Lezzi
Workflows and Distributed Computing
Barcelona Supercomputing Center
daniele.lezzi@bsc.es

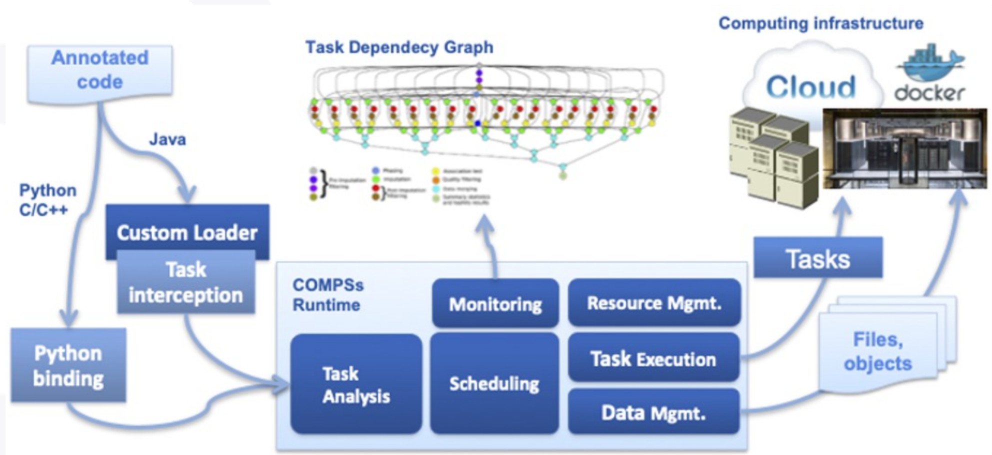
Outline

- COMPSs overview
- Dislib overview
- The ds-array data structure
- Supported methods
- Some results
- Machine learning basics
- Typical workflow in dislib
- Sample code: C-SVM
- Browsing the dislib website

[redacted]

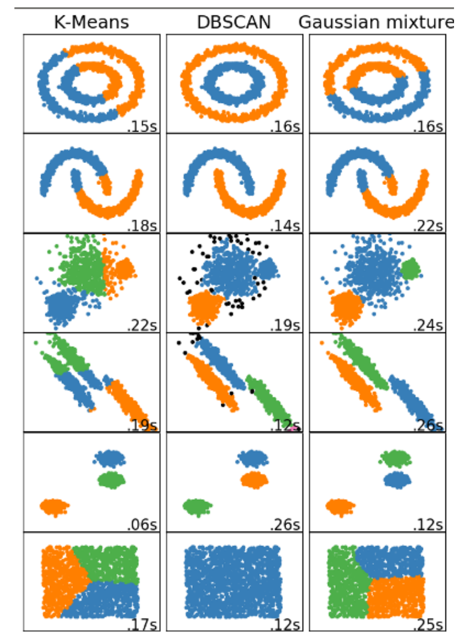
The application developer provides a sequential Python script whose functions are annotated through decorators; these **annotations** are used by the runtime to run those parts of code as asynchronous parallel tasks code.
These annotations describe the type of **parameters and constraints** on the resources. PyCOMPSs also provides a set of APIs to control the flow of the applications (fault tolerance and synchronisation points).
PyCOMPSs processes the information provided by the user through Python decorators and generates a **dependency graph.**

```python
@constraint(computing_units="${ComputingUnits}")
@task(x_list={Type: COLLECTION_IN, Depth: 2},
      y_list={Type: COLLECTION_IN, Depth: 2},
      id_list={Type: COLLECTION_IN, Depth: 2},
      returns=4)
def _train(x_list, y_list, id_list, random_state, **params):
    x, y, ids = _merge(x_list, y_list, id_list)

    clf = SVC(random_state=random_state, **params)
    clf.fit(X=x, y=y.ravel())

    sup = x[clf.support_]
    start, end = 0, 0
    sv = []

    for xi in x_list[0]:
        end += xi.shape[1]
        sv.append(sup[:, start:end])
        start = end

    sv_labels = y[clf.support_]
    sv_ids = ids[clf.support_]

    return sv, sv_labels, sv_ids, clf
```



Dislib: parallel machine learning

- dislib: Collection of machine learning algorithms
  - Unified interface, inspired in scikit-learn (fit-predict)
  - Based on a distributed data structure (ds-array)
  - Unified data acquisition methods
  - Parallelism transparent to the user – PyCOMPSs parallelism hidden
  - Open source, available to the community
- Provides multiple methods:
  - data initialization
  - Clustering
  - Classification
  - Model selection, ...

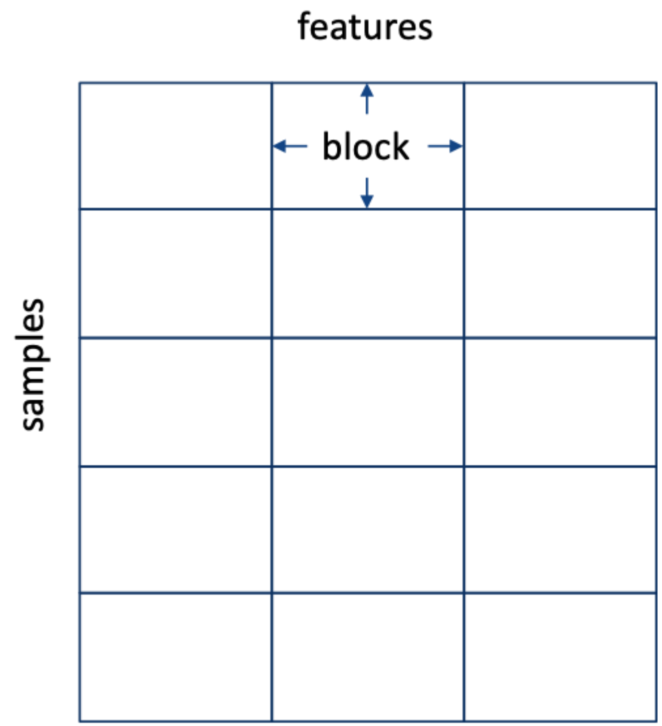


dislib.bsc.es

DISLIB | Distributed Computing Library

Distributed array (ds-array)

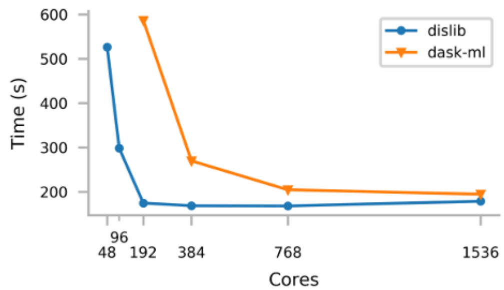
- 2-dimensional structure (i.e., matrix)
  - Divided in blocks (NumPy arrays)
- Works as a regular Python object
  - But not always stored in local memory!
- Methods for instantiation and slicing with the same syntax of numpy arrays:
  - Internally parallelized with PyCOMPSs:
  - Loading data (e.g. from a text file)
  - Indexing (e.g., x[3], x[5:10]
  - Operators (e.g., x.min(), x.transpose())
- ds-arrays can be iterated efficiently along both axes
- Samples and labels can be represented by independent distributed arrays
- Data not always in memory:
  - Inherent support for out-of-core operations, enabling large data-sets



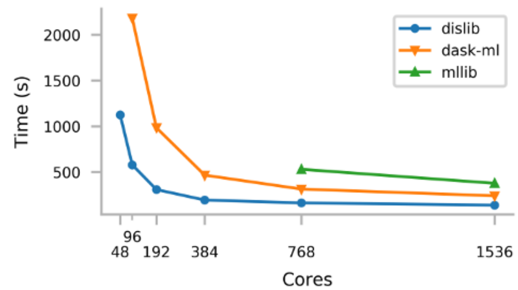
Supported methods

- Array creation routines
  - Multiple routines to create ds-arrays from random, existing data, files, ...
- Utilities to access arrays, scale, apply a function, ...
- Matrix decomposition:
  - Principal Component Analysis (PCA)
  - QR
  - TSQR
  - SVD
- Clustering:
  - DBSCAN
  - K-Means
  - Gaussian Mixture
  - Daura (Gromos)
- Classification
  - CascadeSVM
  - RandomForest classifier
  - DecisionTree classifier
- Recommendation
  - Alternating least squares (ALS)
- Regression
  - Linear regression
  - LASSO
  - RandomForest regressor
  - DecisionTree regressor
- Neighbour queries:
  - k-nearest neighbours
- Model selection:
  - GridSearch
  - RandomizedSearch
  - K-fold

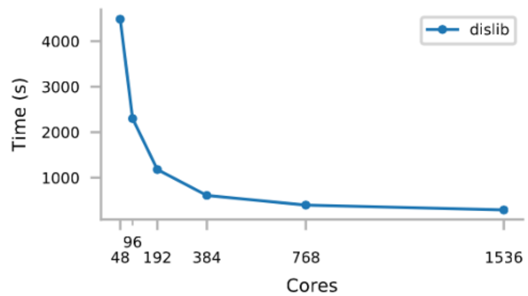
dislib sample results - K-means clustering



1 billion samples
50 features



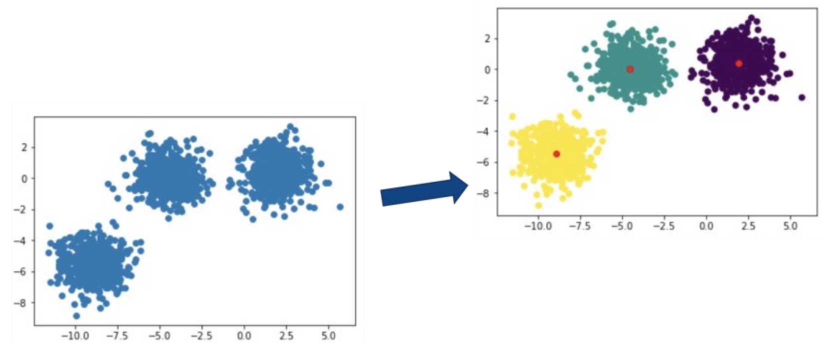
500 million samples
100 features



2 billion samples
100 features

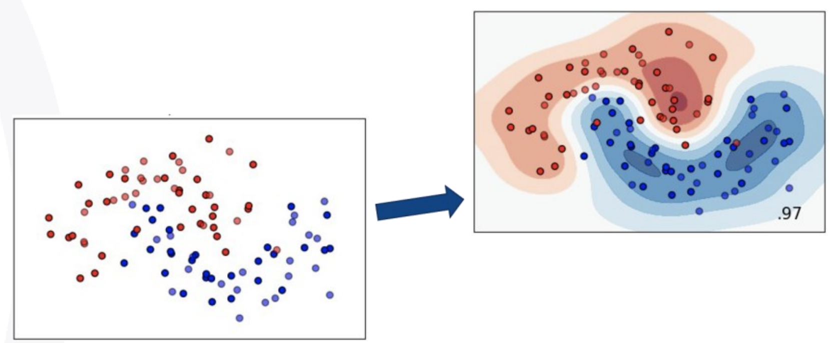
For very large sizes, dislib can obtain results while MLlib and dask fail to finish the execution

- Unsupervised
  - Find unknown patterns in (unlabelled) data
  - Example: clustering



Unlabelled data

- Supervised
  - Learn a decision function from a labelled data
  - Example: classification



Labeled data

Estimators

- Based on scikit-learn
- Estimator = anything that learns from data (labelled or unlabelled)
- Two main methods:
  - fit → learns something from data (e.g., a decision function)
  - predict → provides new information based on a fitted model (e.g., labels data based on the computed decision function)

Typical workflow

Block size

1. Read input data from file/s

2. Instantiate estimator with parameters

3. Fit estimator with training data

4. Make predictions on test data

```
x = load_txt_file("train.csv", (10, 780))
x_test = load_txt_file("test.csv", (10, 780))

kmeans = KMeans(n_clusters=10)

kmeans.fit(x)

kmeans.predict(x_test)
```

Internals: ds-array implementation

- Implemented as an object, with main parameters:
  - Block size: shape of a regular block
  - Blocks: list of lists of NumPy ndarray (or spmatrix)
  - Sparse: whether the block is sparse or not
- Methods
  - Most of the methods for array creation or transformation are parallelized with PyCOMPSs:

```
@task(returns=np.array)
def _random_block(shape, seed):
    np.random.seed(seed)
    return np.random.random(shape)
```

```
@task(blocks={Type: COLLECTION_IN, Depth: 2}, returns=np.array)
def _block_apply_axis(func, axis, blocks, *args, **kwargs):
...
```

```
...
for block in x._iterator(axis=(not axis)):
    out = _block_apply_axis(func, axis, block._blocks, *args, **kwargs)
    out_blocks.append(out)
...
```

```
x = ds.random_array((100, 100), block_size=(25, 25))
mean = ds.apply_along_axis(np.mean, 0, x)
```

Samp[redacted]e code: C-SVM

```
""x : ds-array, shape=(n_samples, n_features)
            Training samples.
y : ds-array, shape=(n_samples, 1)
            Class labels of x.""
while not self._check_finished():
    self._do_iteration(x, y, ids_list)
    if self.check_convergence:
        self._check_convergence_and_update_w()
        self._print_iteration()
    return se
```

```
def _do_iteration(self, x, y, ids_list):
    ...
    # first level
    for partition, id_bk in zip(_paired_partition(x, y), ids_list):
        x_data = partition[0]._blocks
        y_data = partition[1]._blocks
        ...
        _tmp = _train(x_data, y_data, ids, self.random_state, **pars)
        sv, sv_labels, sv_ids, self._clf = _tmp
        q.append((sv, sv_labels, sv_ids))

    # reduction
    while len(q) > arity:
        x_data = q[:arity]
        ...
        _tmp = _train(x_data, y_data, ids, self.random_state, **pars)
    ...
```

Set of tuples (x_data, y_data) that are partitions of x and y horizontally with parts of both samples.

Sample code: C-SVM

PyCOMPSs collections

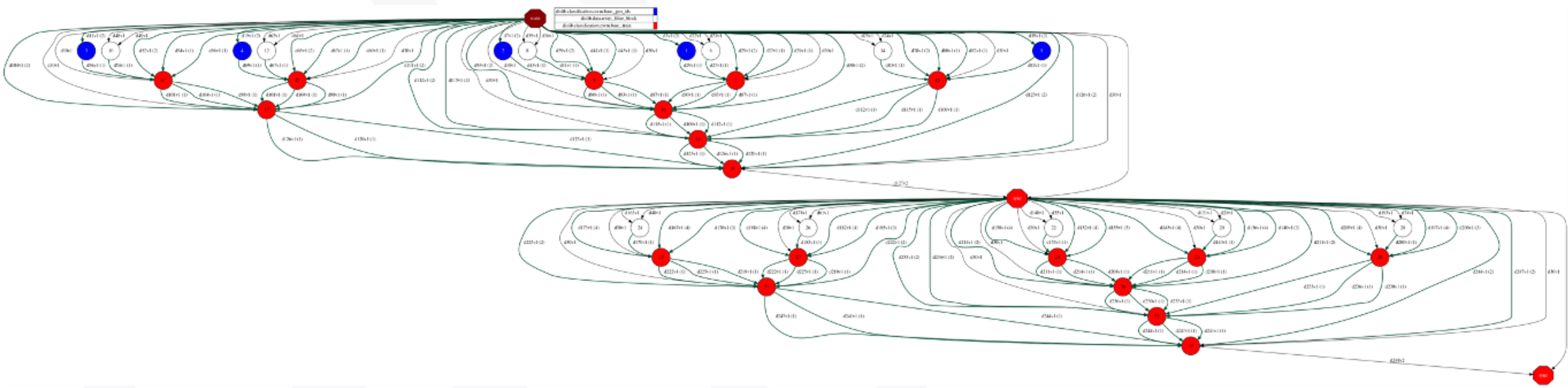
leverages Scikit-learn

```
from sklearn.svm import SVC
@task(x_list={Type: COLLECTION_IN, Depth: 2},
      y_list={Type: COLLECTION_IN, Depth: 2},
      id_list={Type: COLLECTION_IN, Depth: 2},
      returns=4)
def _train(x_list, y_list, id_list, random_state, **params):
    x, y, ids = _merge(x_list, y_list, id_list)

    clf = SVC(random_state=random_state, **params)
    clf.fit(X=x, y=y.ravel())

    sup = x[clf.support_]
    start, end = 0, 0
    sv = []

    for xi in x_list[0]:
        end += xi.shape[1]
        sv.append(sup[:, start:end])
        start = end

    sv_labels = y[clf.support_]
    sv_ids = ids[clf.support_]

    return sv, sv_labels, sv_ids, clf
```

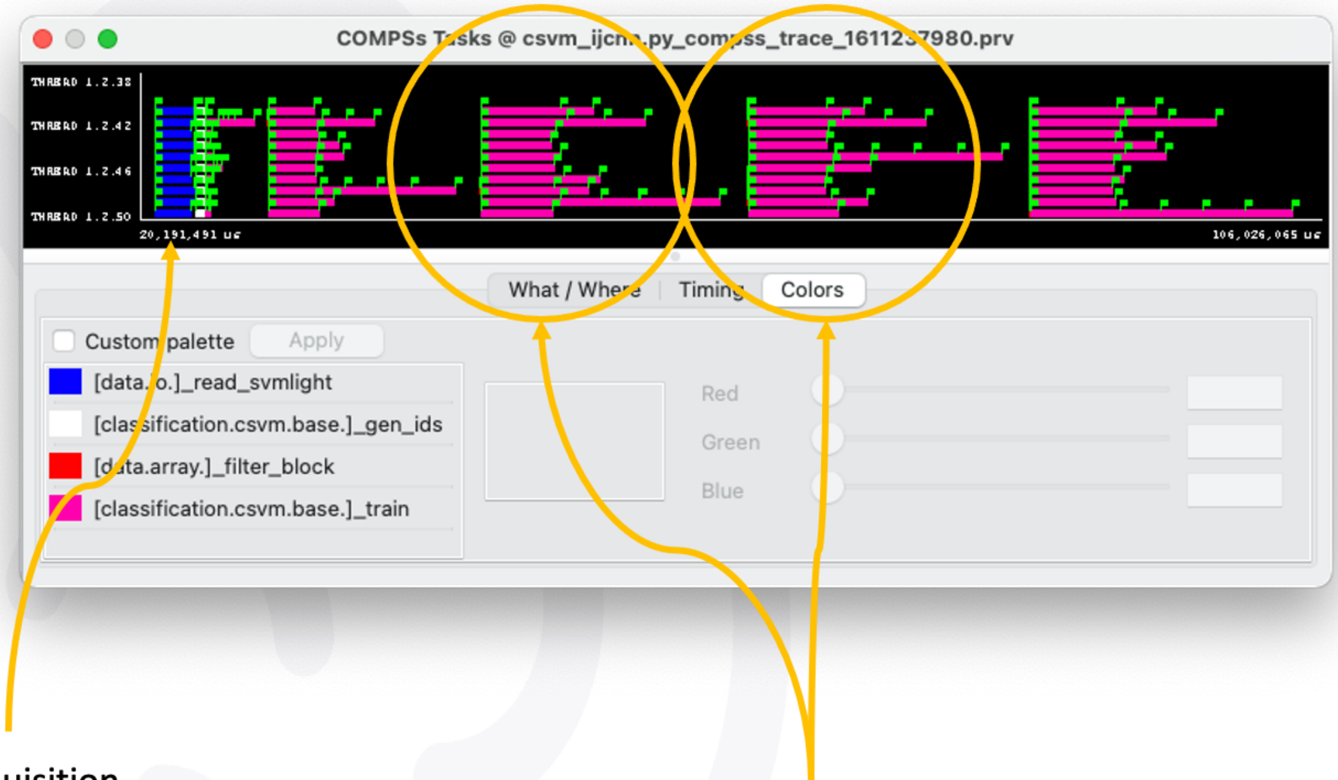
Sample user code: C-SVM

```
import dislib as ds
from dislib.classification import CascadeSVM
from dislib.utils import shuffle

def main():
    x_ij, y_ij = ds.load_svmlight_file("./C-SVM/datasets/train",
        block_size=(5000, 22), n_features=22, store_sparse=True)

    csvm = CascadeSVM(c=10000, gamma=0.01)

    csvm.fit(x_ij, y_ij)

if __name__ == "__main__":
    main()
```

C-SVM Task graph

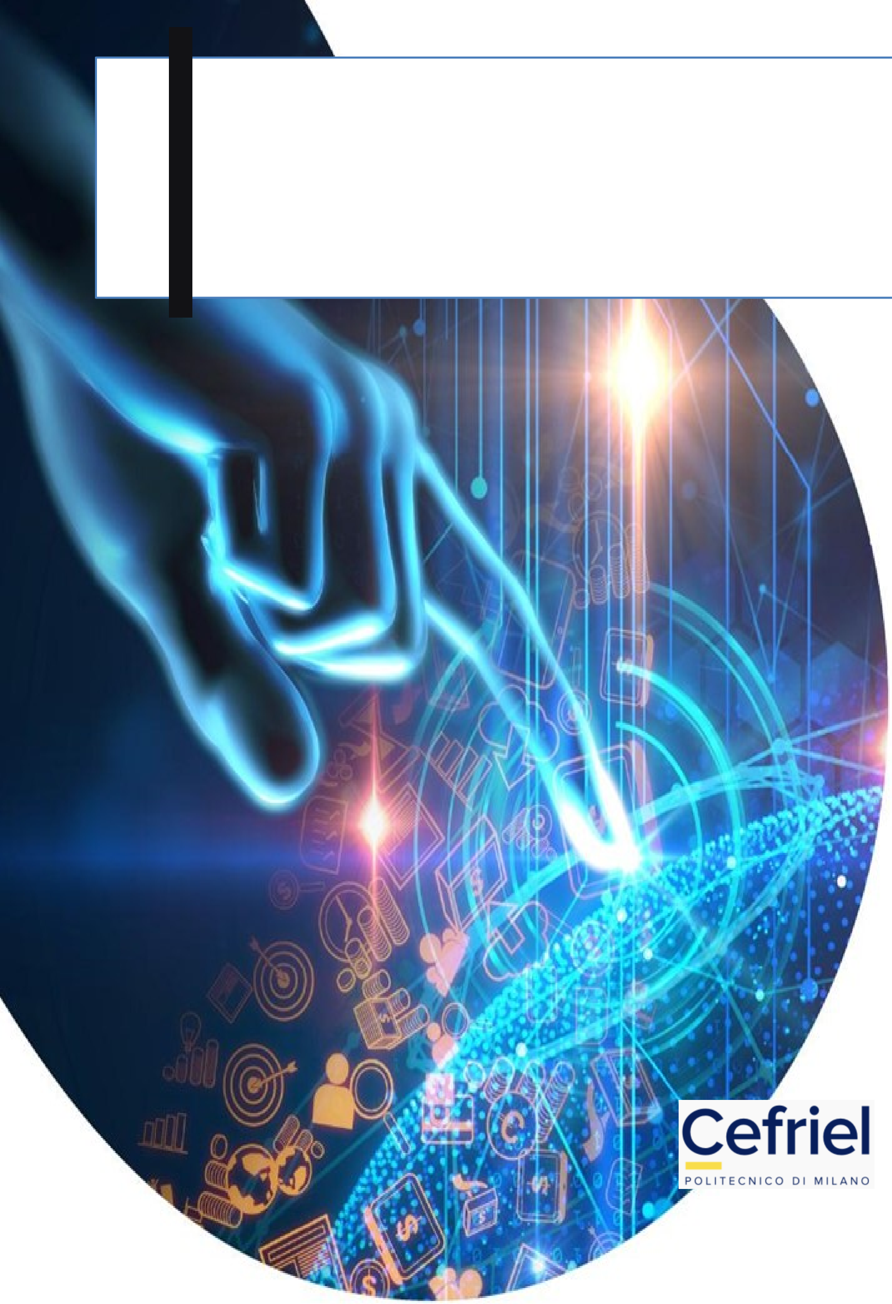


C-SVM Tracefile



Data acquisition

iterations



Optimization and runtime management of AI applications

Federica Filippini, Hamta Sedghani, Enrico Galimberti

Politecnico di Milano, Italy

{name}.{lastname}@polimi.it

Giuseppe Caccia

Cefriel, Italy

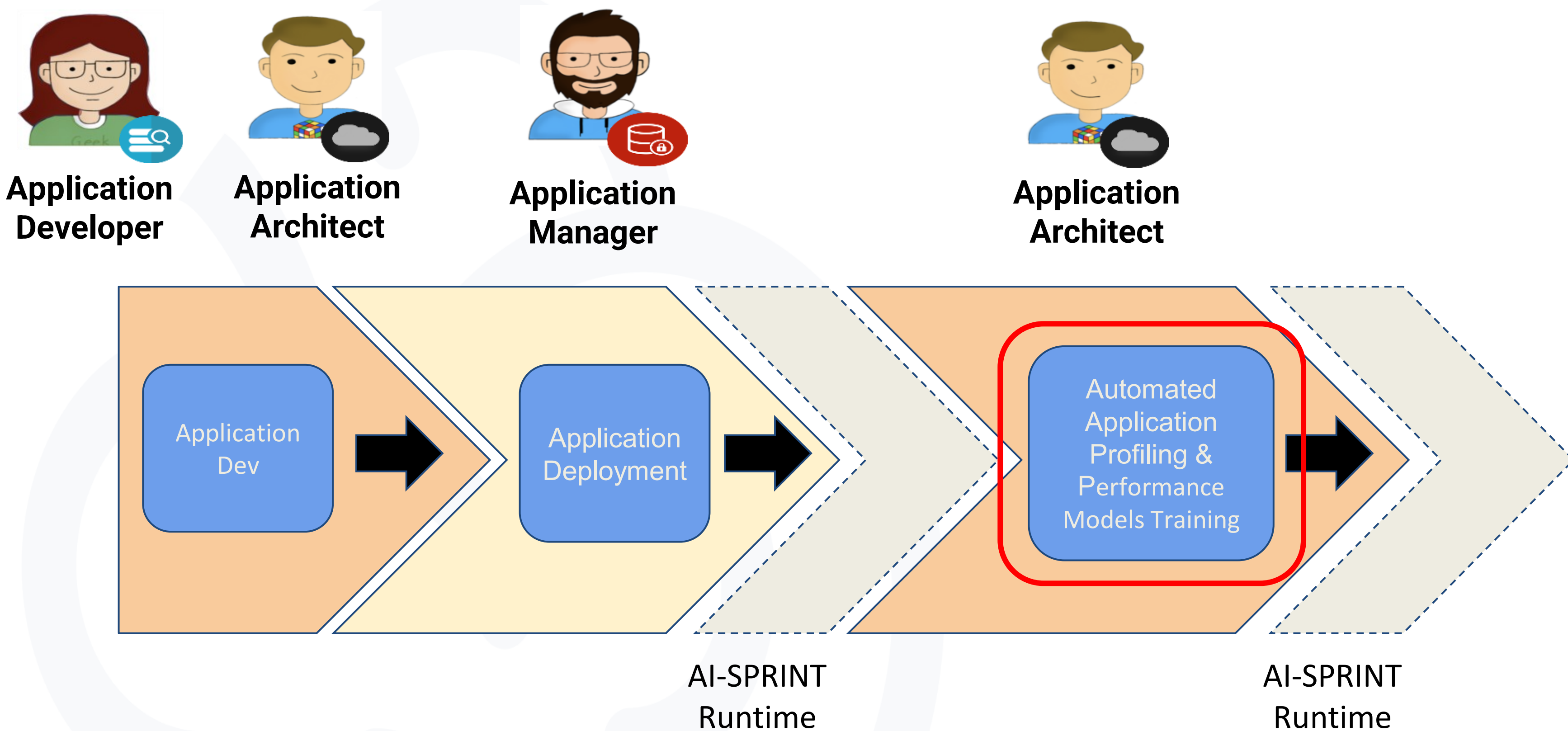
caccia@cefriel.it

Cefriel POLITECNICO DI MILANO

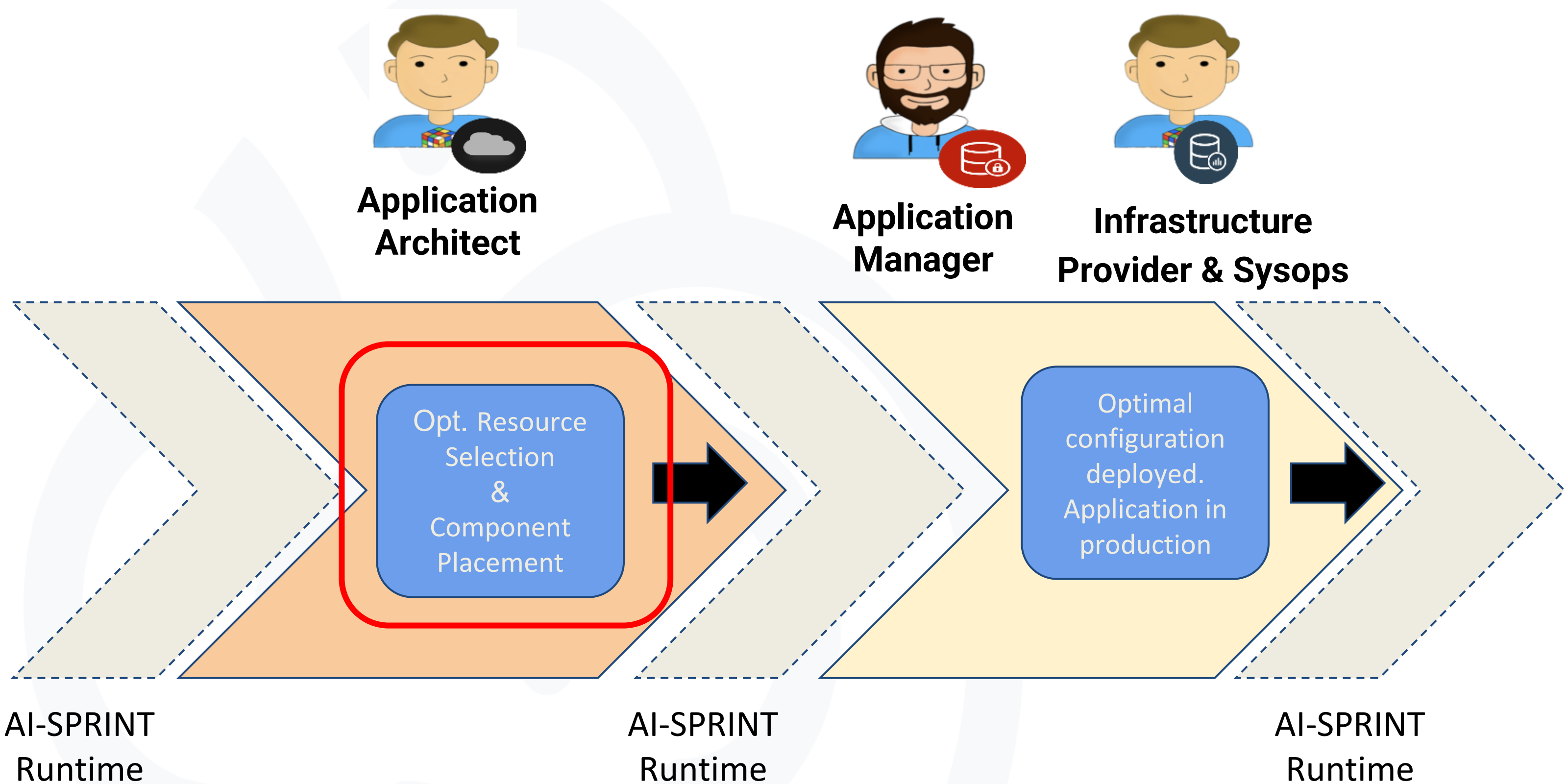
POLITECNICO MILANO 1863

AI SPRINT AI IN SECURE PRIVACY-PRESERVING COMPUTING CONTINUUM

AI Application Design Workflow



AI Application Design Workflow



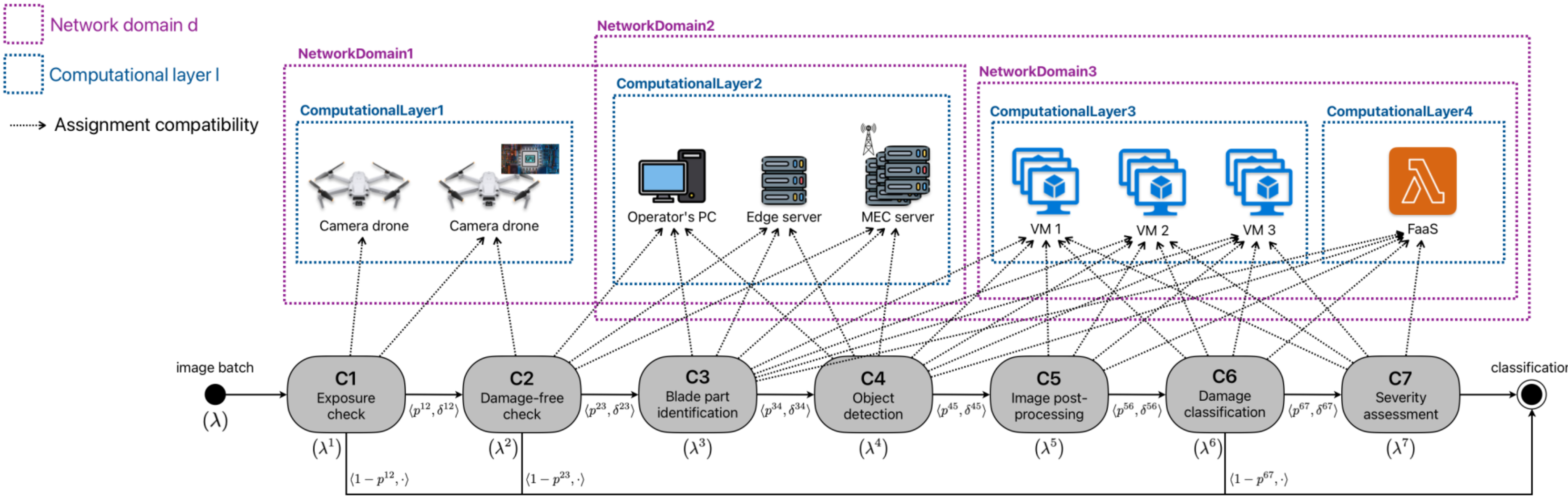
- **Goal:** predicting the response time of application components on the candidate resource configurations

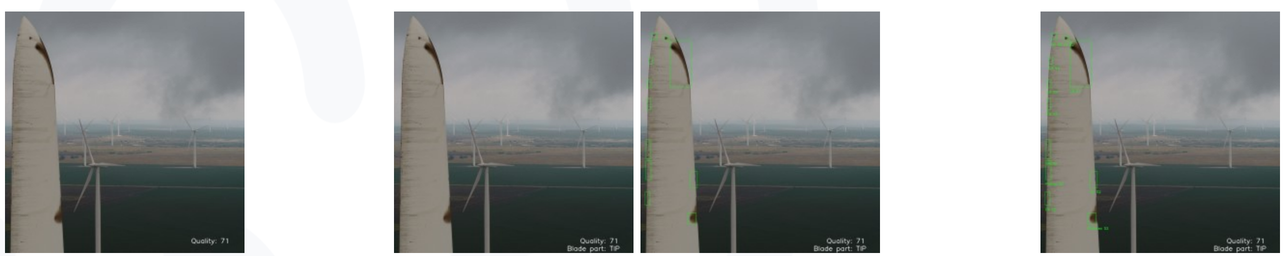
- **Why?** To support the selection of the optimal placement, minimizing costs and guaranteeing performance constraints

- **How?** Several strategies:
  - Analytical models (e.g., M/M/1, M/G/1...)
  - Machine Learning-based models

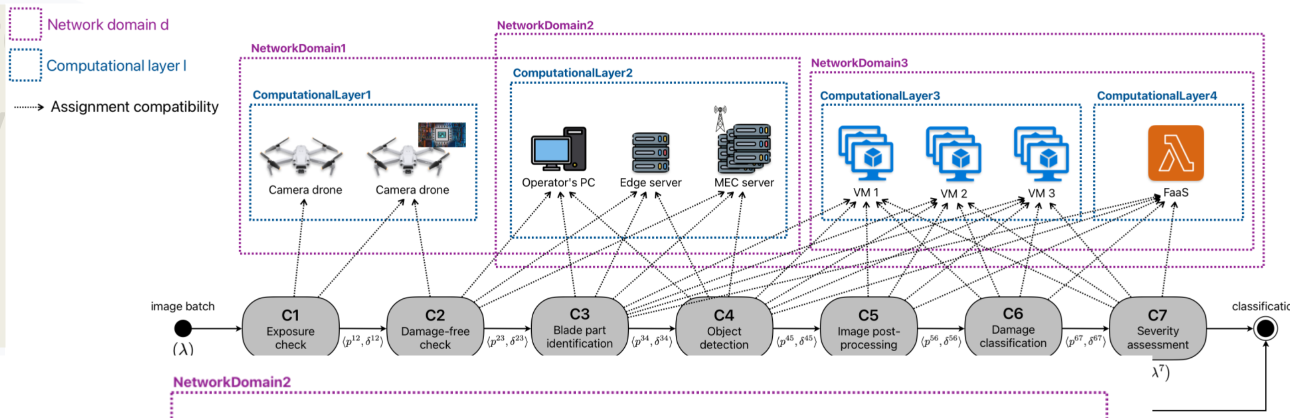
▲ Accurate
▲ Limited profiling
independently on
required
theoretical
▲ No training time
assumptions
▲ Fast execution
▲ Specific for the current
component/resource
▼ Based on assumptions
pair
that may not hold in
▼ Need to be periodically
practical scenarios
re-trained

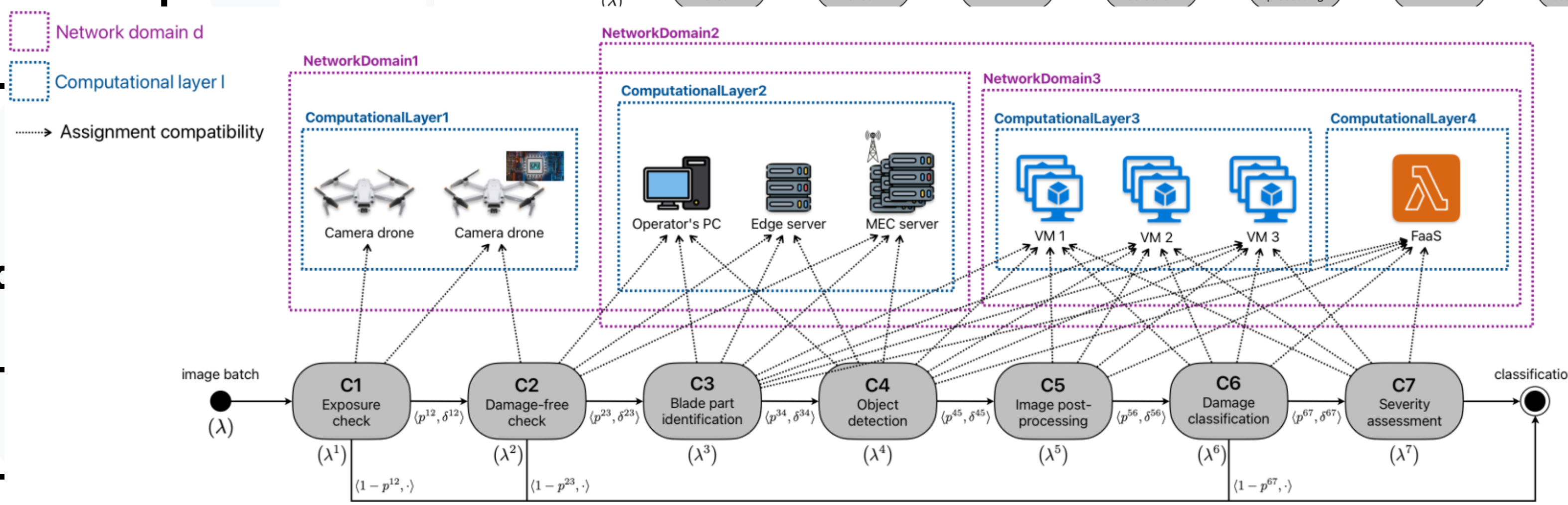
Sample use-case application





- **Resource selection & component placement problem**
  - Which resour
    - How many V
  - Which neural
    component
  -
- at c
  -
  -

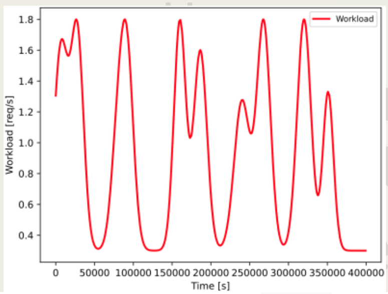


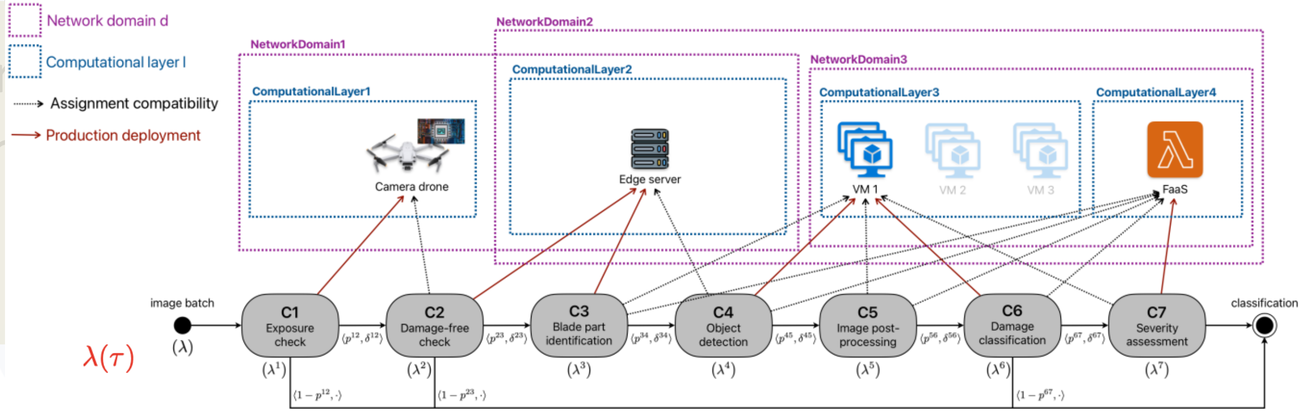


violations

Design-time and runtime optimization

- **Resource selection & component placement** problem
  – Which resources to use at each computational layer





- at **design time**…
  – Based on the **expected input workload**
  – To dimension the resources & avoid QoS constraints violations

- …and at **runtime**!
  – In response to **workload variations** that induce resource saturation/underutilization

OSCAR-P

PERFORMANCE MODELS

+

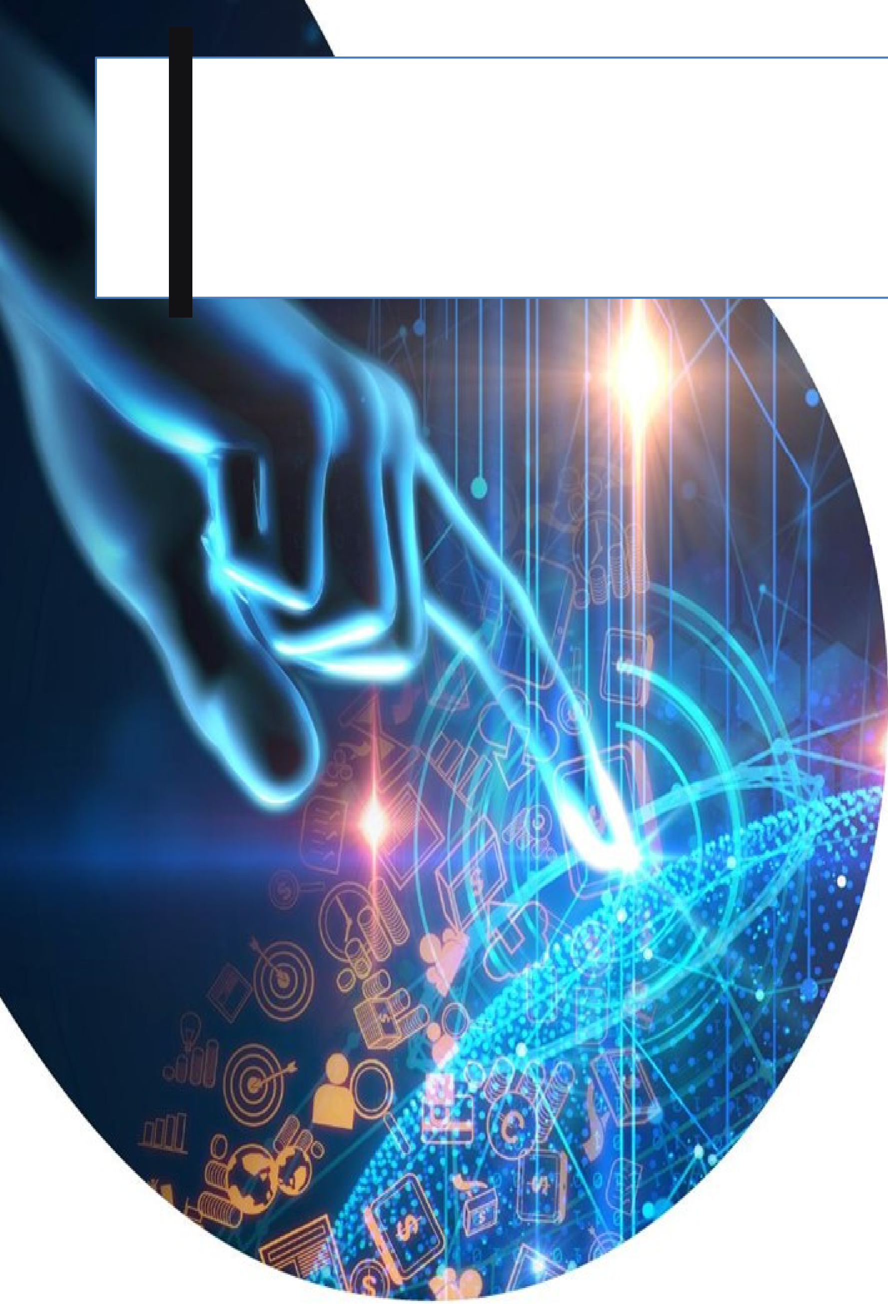
SPACE4AI-D

“Profiling and Predicting the Performance of Function as a Service-based Applications in Computing Continua”

“A Design-time Tool for AI Applications Resource Selection in Computing Continua”

SPACE4AI-R

“A Runtime Management Tool for AI Applications Component Placement and Resource Scaling in Computing Continua”



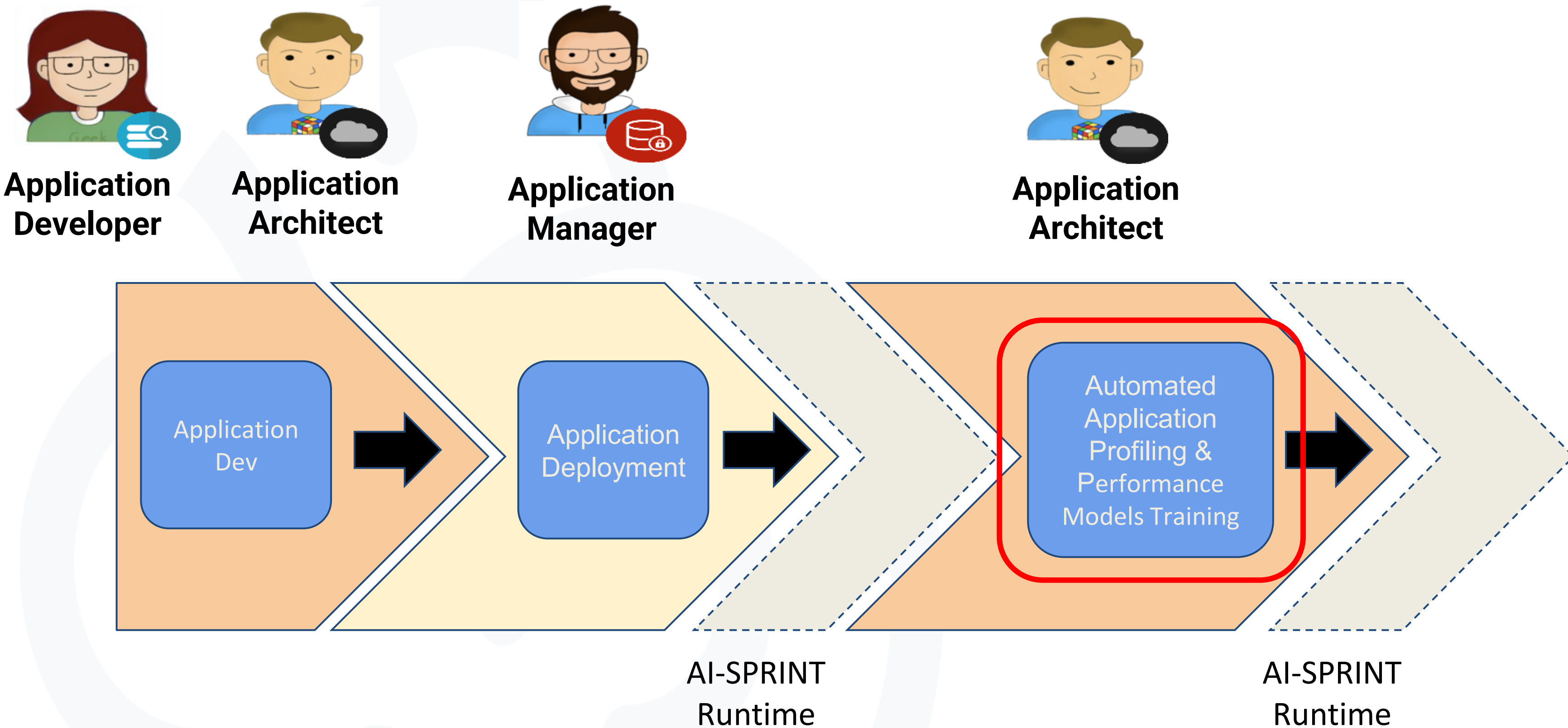
OSCAR-P

and Performance Models generation

Federica Filippini, Enrico Galimberti

{name}.{lastname}@polimi.it

AI Application Design Workflow



Aut[redacted]tic Application Profiling and ML Models Training

Proble[redacted] solved:

- **Automatic** application **performance profiling**, with parameters set **declaratively** in a **configuration file**
- Infrastructures are **automatically deployed and configured**, thanks to the integration with IM
- **Dataset preparation and ML models training** is also **fully automated**

Motivations:

- Deploying and configuring multiple infrastructures is a **complex task**
- Profiling an application on multiple configurations manually is **extremely time consuming**

State Of The Art

- SeBS [1] (Serverless Benchmark Suite) aims at being the first comprehensive benchmarking tool that systematically supports a wide array of applications and cloud resources, including commercial providers such as AWS, Azure, and Google Cloud.
- EdgeBench [2] instead analyzes two of them, Amazon AWS Greengrass and Microsoft Azure IoT Edge, using different performance metrics, and also compares the performance of the edge frameworks to the respective cloud-only implementations.
- DeFog [3] presents a benchmarking tool that focuses on testing an application across a cloud-only, edge-only and cloud-edge, by comparing the performance across the different deployments allows to gain insight on potential improvements. The tool collects metric on the latency of the application, both for communication and computation, under normal conditions and under stress, with the aim of understanding how the services that make up an application can be better distributed across the computing continuum.

[1] M. Copik, G. Kwasniewski, M. Besta, et al., Sebs: A serverless benchmark suite for function-as-a-service computing, in: ICM, 2021, pp. 64–78.

[2] A. Das, S. Patterson, M. Wittie, Edgebench: Benchmarking edge computing platforms, in: UCC, IEEE, 2018, pp. 175–180.

[3] J. McChesney, N. Wang, A. Tanwer, E. de Lara, B. Varghese, Defog: fog computing benchmarks, in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, 2019, pp. 47–58.

State Of The Art

- SuanMing[4] is an integrated framework for learning regressors using different algorithms (Random Forest, Nearest Neighbor Regression, Ridge Regression, and Support Vector Regression) of microservice-based systems running in public and private clouds, with the end goal of identifying potential sources of performance loss in complex applications.
- Another … hmoudi et al. [5], w… etrics by conside… the requests…

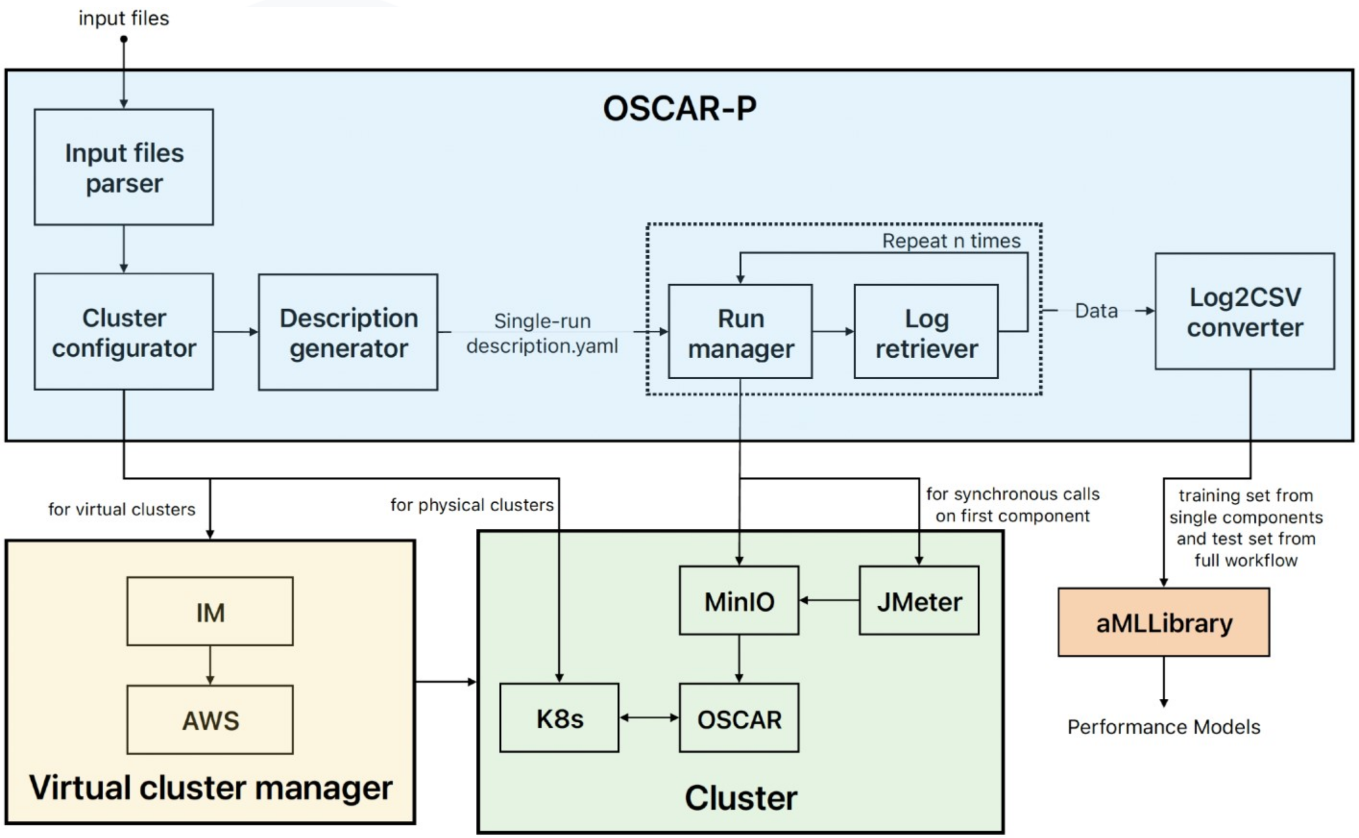
[4] J. Grohmann, M. … n: ICPE, 2021, pp. 165–176.

[5] N. Mahmoudi, H…

OSCAR-P is **focused on** benchmarking the **OSCAR** framework, which can be deployed on top of any commercial cloud → it is **cloud provider agnostic**

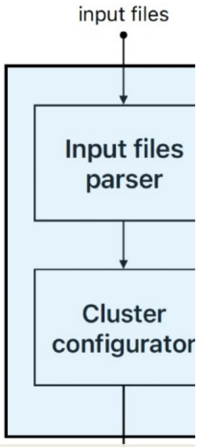
OSCAR-P can provide the average execution time of an application workflow with **acceptable precision** (MAPE lower than 25%) even for **unseen configurations** and with a **limited testing campaign**

OSCAR-P



input files
OSCAR-P
Input files parser
Cluster configurator
Description generator
Single-run description.yaml
Repeat n times
Run manager
Log retriever
Data
Log2CSV converter
for virtual clusters
for physical clusters
for synchronous calls on first component
training set from single components and test set from full workflow
IM
AWS
Virtual cluster manager
MinIO
JMeter
K8s
OSCAR
Cluster
aMLLibrary
Performance Models

OSCAR-P



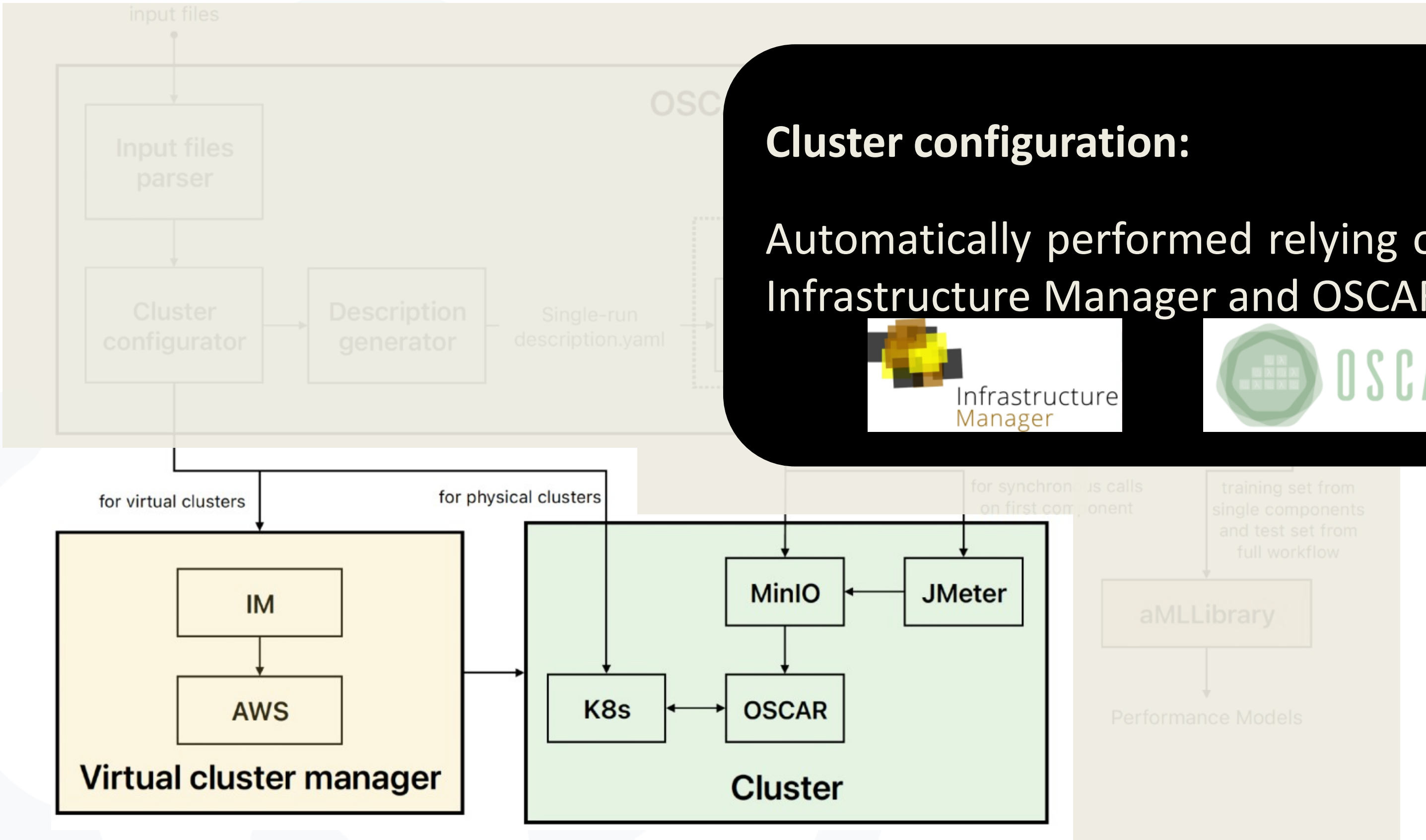

```
aisprint
├── deployments
├── designs
└── logs
ams
common_config
├── annotations.yaml
├── application_dag.yaml
├── candidate_deployments.yaml
└── candidate_resources.yaml
im
└── auth.dat
oscar
oscarp
└── run_parameters.yaml
py...
space4ai-d
├── qos_constraints.yaml
└── SPACE4AI-D.yaml
space4ai-r
src
├── blurry-faces-onnx
└── mask-detector-onnx
```


```yaml
input_files:
  storage_bucket: "storage"
  filename: "input-video.mp4"
asynchronous:
  batch_size: 1
  number_of_batches: 1
  distribution: "deterministic"
  inter_upload_time: 30
synchronous:
  number_of_pre_allocated_pods: 2
  connect_timeout_seconds: 30
  request_timeout_seconds: 300
  worker_nodes: 4
  intervals:
    - throughput: 2
      number_of_threads: 2
      duration_seconds: 600
      ramp_up_seconds: 5
components:
  component1:
    parallelism: [ 1 ]
  component2:
    parallelism: [ 1 ]
    distribution: "deterministic"
run:
  test_synchronously: False
  test_single_services: False
  train_models: False
  campaign_dir: "test"
  repetitions: 1
  cooldown_time: 60
other:
  time_correction: 0
  domain_name: "polimi-aisprint.click"
  clean_infrastructures_before_testing: False
  clean_infrastructures_after_testing: False
```


Required input:

- Physical and virtual resources description
- Application components description
- Application parameters and input data
- Machine Learning models hyperparameters

OSCAR-P

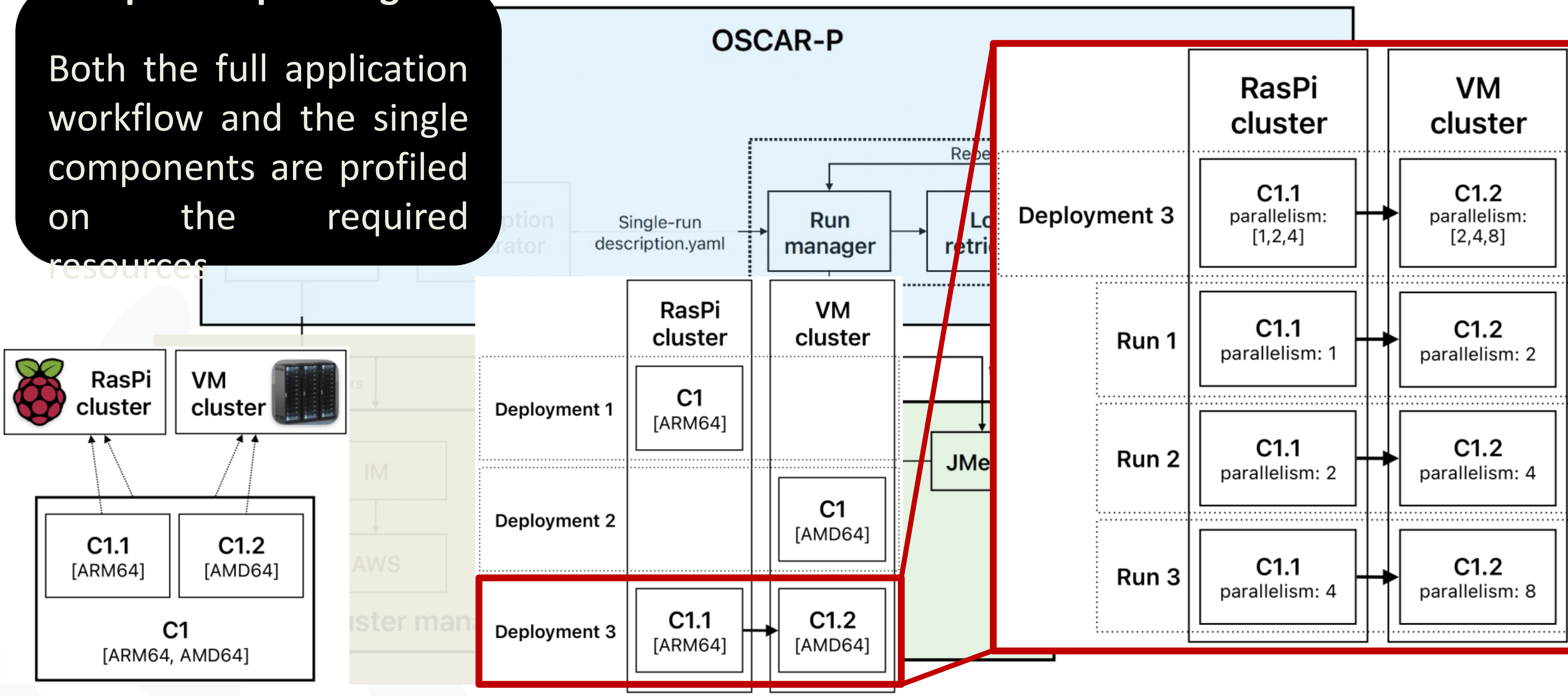


Cluster configuration:

Automatically performed relying on the Infrastructure Manager and OSCAR

Component profiling:

Both the full application workflow and the single components are profiled on the required resources



OSCAR-P

ML-based performance models generation:

Regression models:

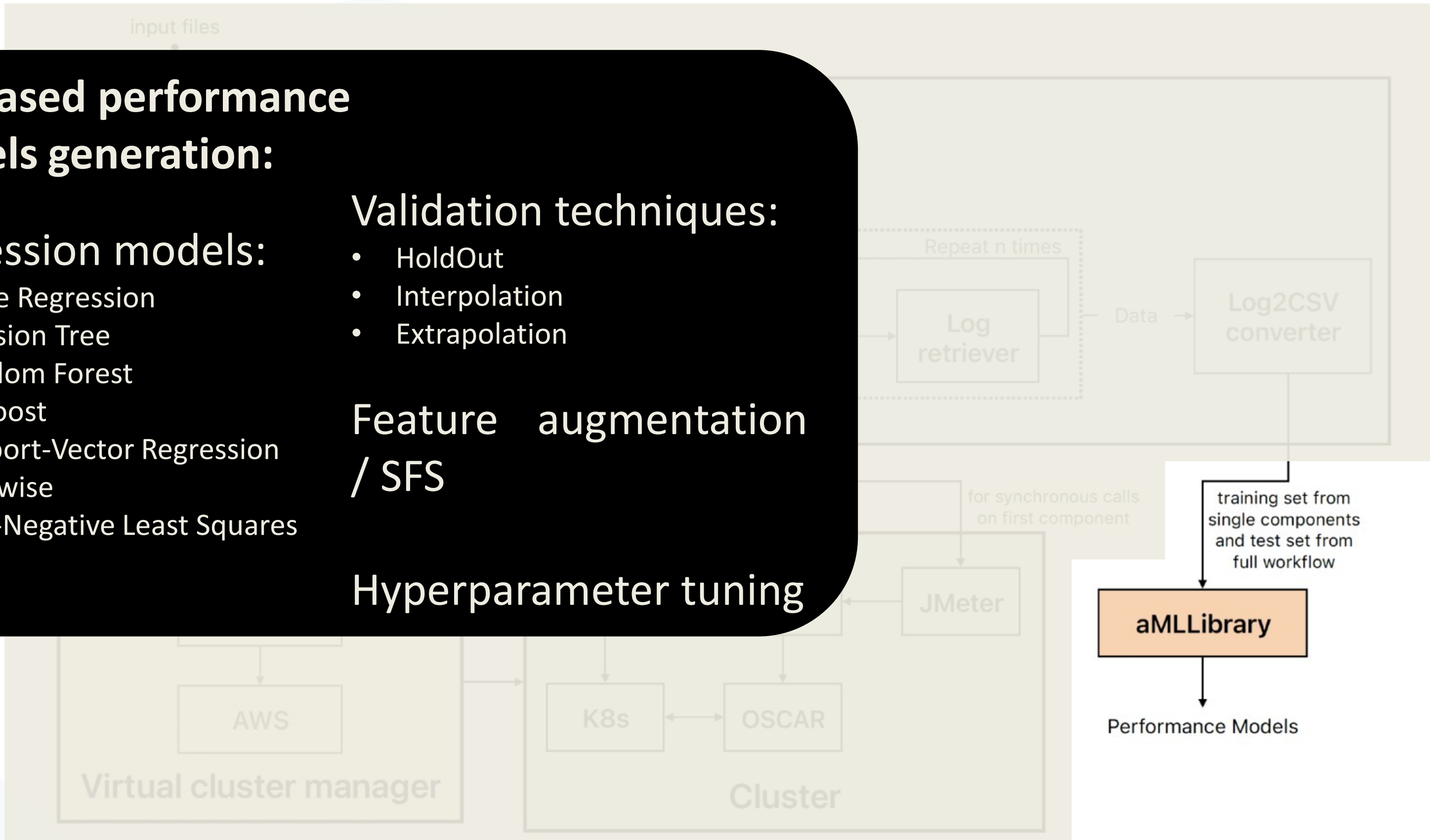
- Ridge Regression
- Decision Tree
- Random Forest
- XGBoost
- Support-Vector Regression
- Stepwise
- Non-Negative Least Squares

Validation techniques:

- HoldOut
- Interpolation
- Extrapolation

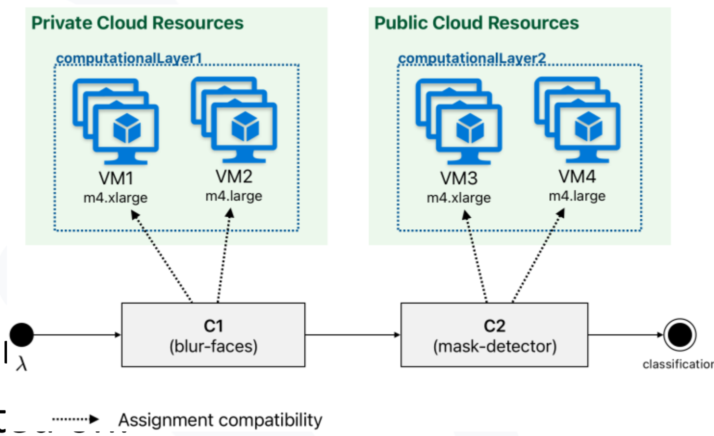
Feature augmentation / SFS

Hyperparameter tuning



Experimental setup

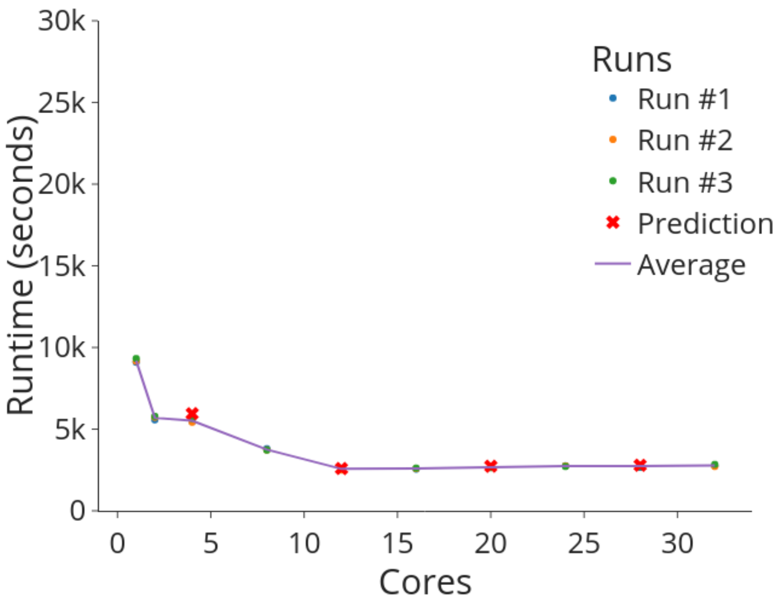
- **2-c[redacted]mponents** application:



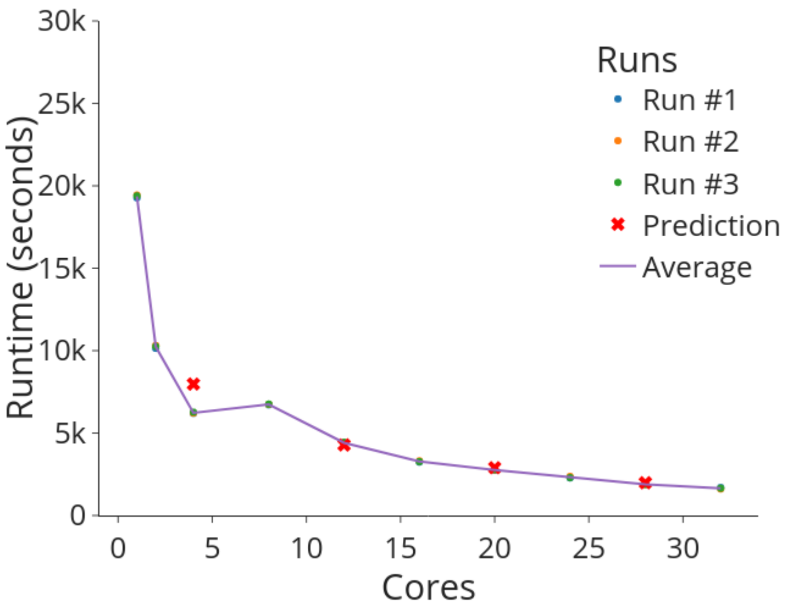
- **Profiling data** collected o **e workflow**
- **Performance models** test
  – **interpolation** and **extrapolation** capabilities
  – **predicting** the application response time given the components data

Experimental results: Interpolation

- blur-faces: MAPE = 3.34%

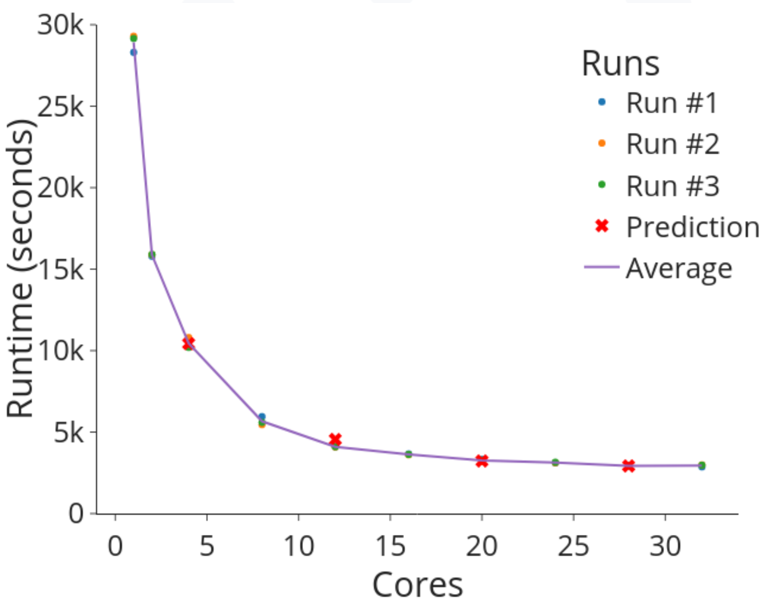


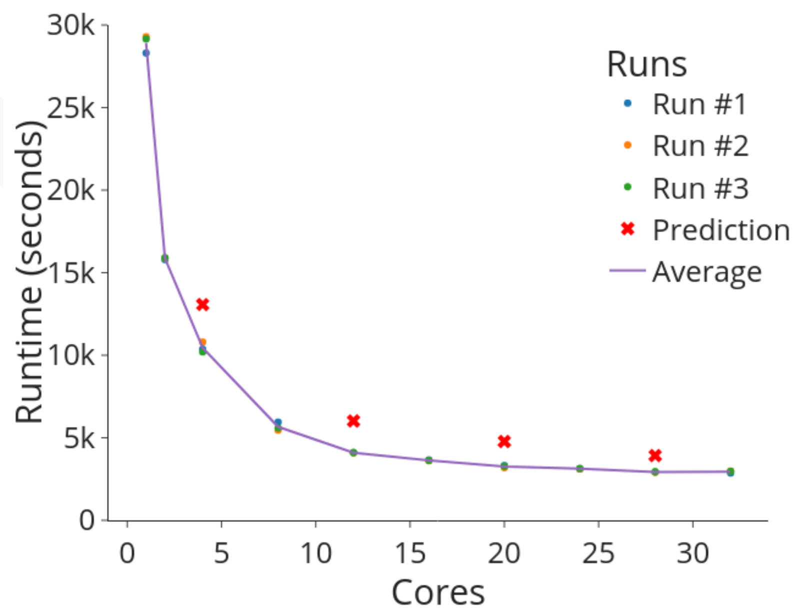
- mask-detector: MAPE = 10.14%



Experimental results: Interpolation

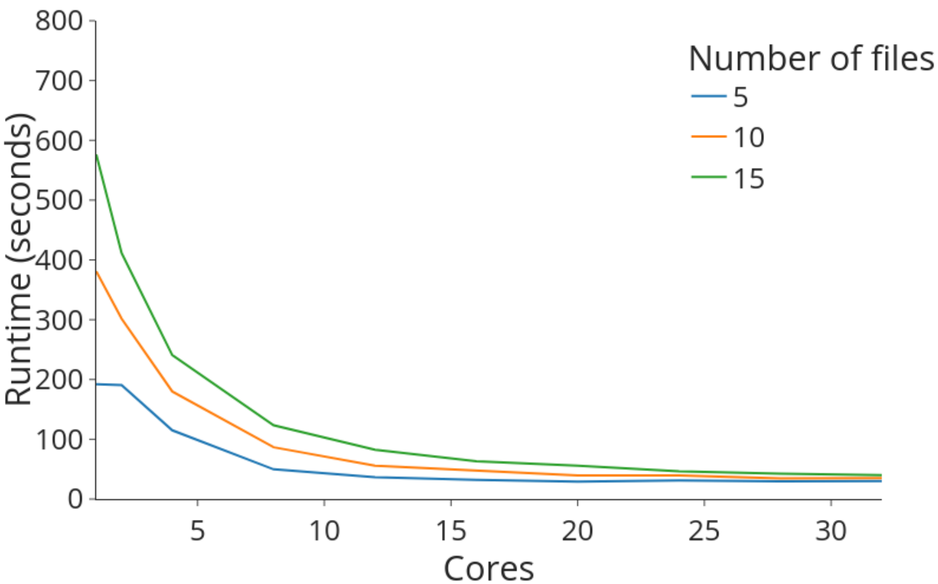
- Full workflow: MAPE = 3.13%
- Combined models: MAPE = 17.08%



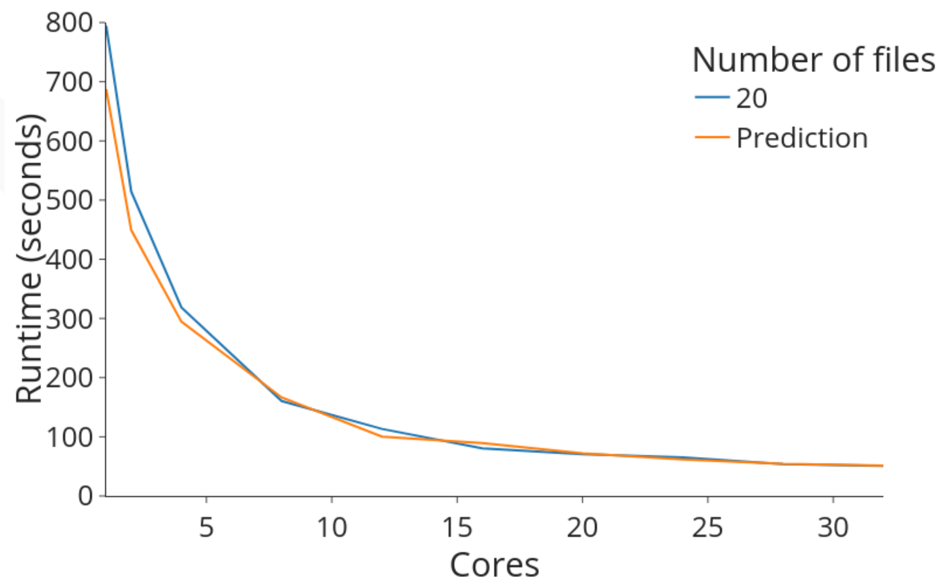


Experimental results: Extrapolation

- Training set



- Predictions: MAPE = 9.75%



References & Links

[1] M. Copik[redacted] G. Kwasniewski, M. Besta, et al., Sebs: A serverless benchmark suite for function-as-a-service computing, in: ICM, 2021, pp. 64–78.

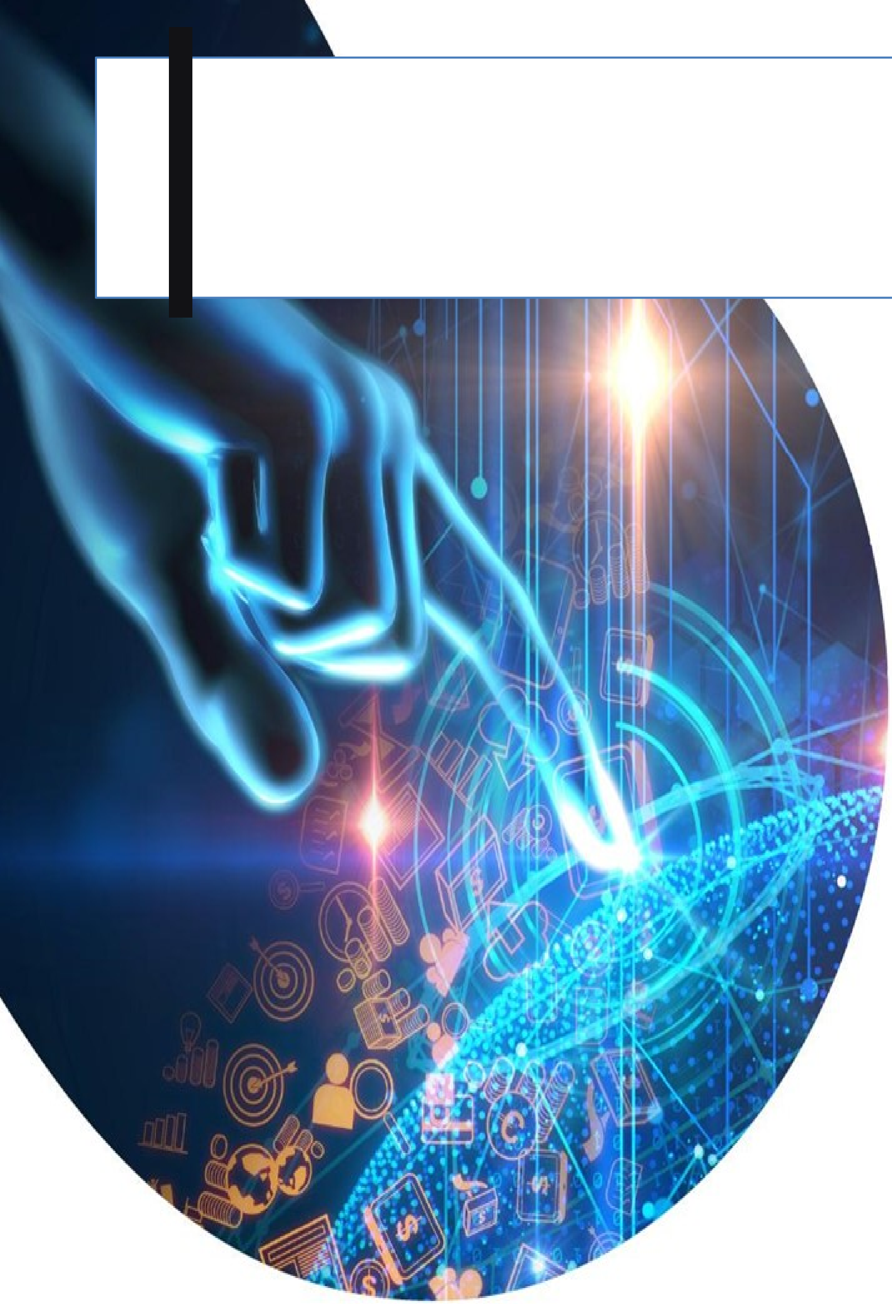
[2] A. Das, S[redacted]. Patterson, M. Wittie, Edgebench: Benchmarking edge computing platforms, in: UCC, IEEE, 2018, pp. 175–180.

[3] J. McChesney, N. Wang, A. Tanwer, E. de Lara, B. Varghese, Defog: fog computing benchmarks, in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, 2019, pp. 47–58.

[4] J. Grohmann, M. Straesser, A. Chalbani, et al., Suanming: Explainable prediction of performance degradations in microservice applications, in: ICPE, 2021, pp. 165–176.

[5] N. Mahmoudi, H. Khazaei, Temporal performance modelling of serverless computing platforms, in: WoSC, 2020, pp. 1–6.

[6] E. Galimberti, B. Guindani, F. Filippini, et al., “OSCAR-P and AMLLibrary: Performance Profiling and Prediction of Computing Continua Applications,” in Companion of the 2023 ACM/SPEC International Conference on Performance Engineering, ser. ICPE ’23 Companion, Coimbra, Portugal: Association for Computing Machinery, 2023, pp. 139–146, isbn: 9798400700729. doi: 10.1145/3578245.3584941.

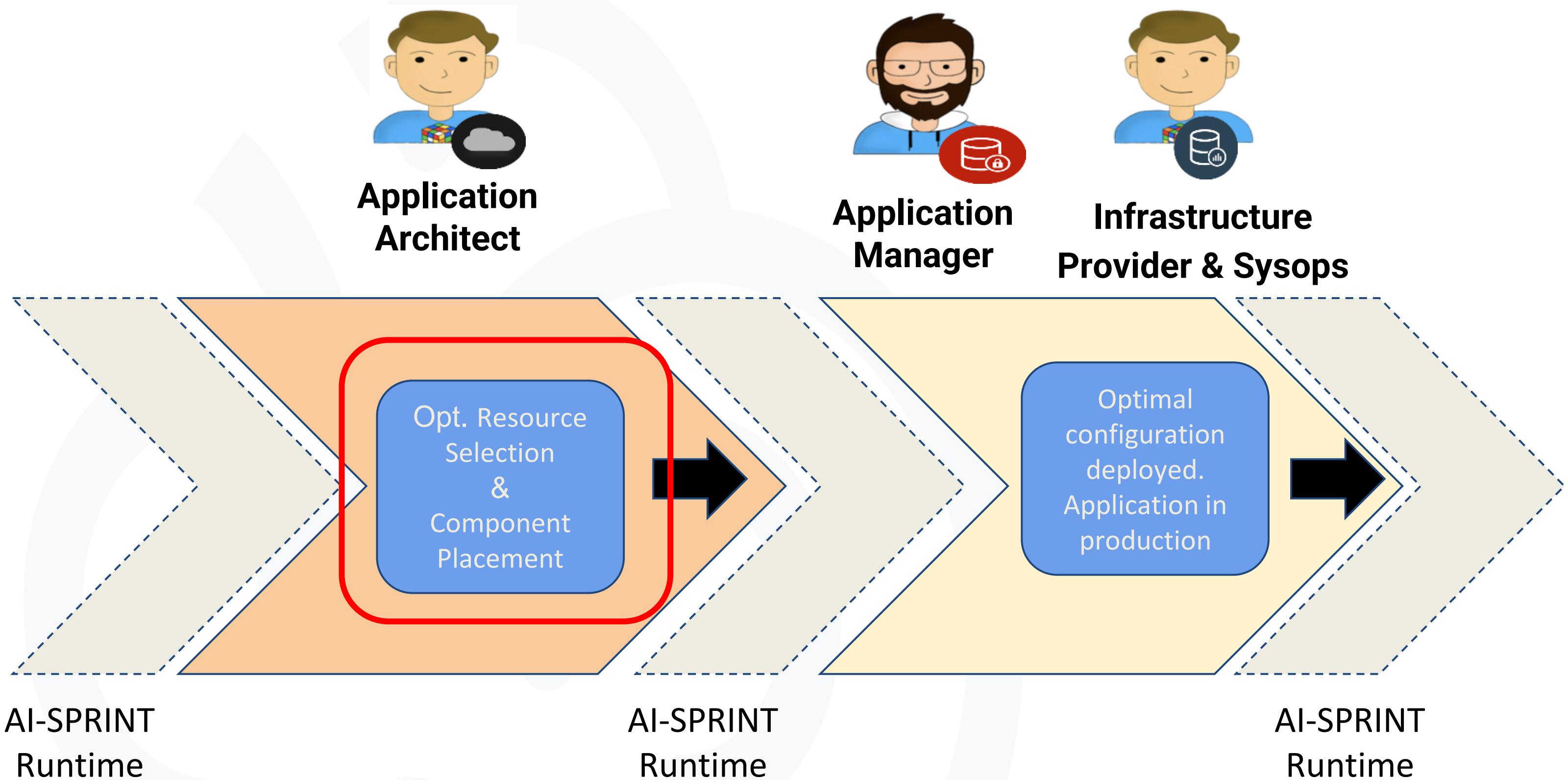


SPACE4AI-D

System PerformAnce and Cost Evaluation on Cloud for AI applications Design

Federica Filippini, Hamta Sedghani
{name}.{lastname}@polimi.it

AI Application Design Workflow



Problem solved:

- **Automatic exploration** of design alternatives to **minimize costs**
- Cope with **technology constraints**, **performance** and **privacy requirements**
- Identify **optimal resources and component placement** at each layer of the computing continuum

Motivations:

- Computing resources are **heterogeneous**
- **Efficient** component placement and resource allocation are crucial to **orchestrate** at best the continuum resources

State Of The Art

- [1] defines a serverless application workflow as a Directed Acyclic Graph (DAG) and proposes two heuristic algorithms to solve two optimization problems: (i) optimize the cost of serverless applications with DAG structure under performance constraint, and (ii) optimize the performance under a budget constraint
- [2] deve … llocation problem … mption while gu …
- [3] tackl … d computa … etermine the min … nts, and formaliz …

SPACE4AI-D is **one of the first proposals** to consider **resource contention** in determining the optimal component placement **for AI applications**

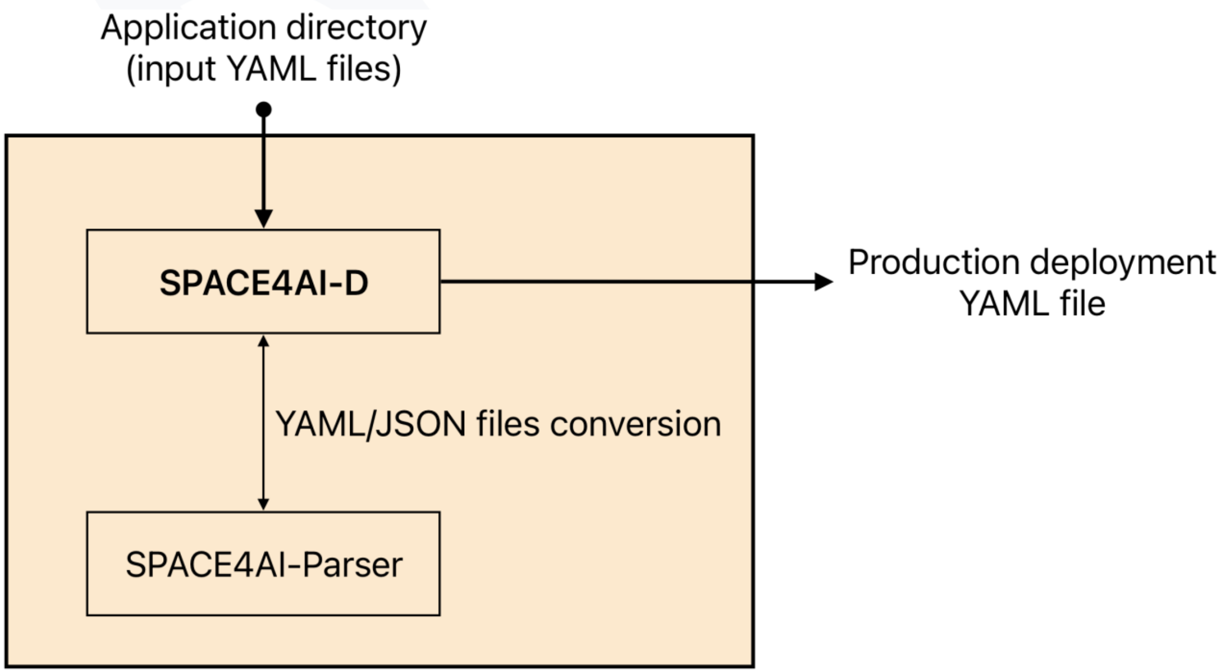
SPACE4AI-D considers **multiple candidate neural network deployments** given by the possibility of **partitioning components at different layers** according to **network and load conditions**

[1] C. Lin and H. Khazaei, "Modeling and Optimization of Performance and Cost of Serverless Applications," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 3, pp. 615-632, 1 March 2021, doi: 10.1109/TPDS.2020.3028841.

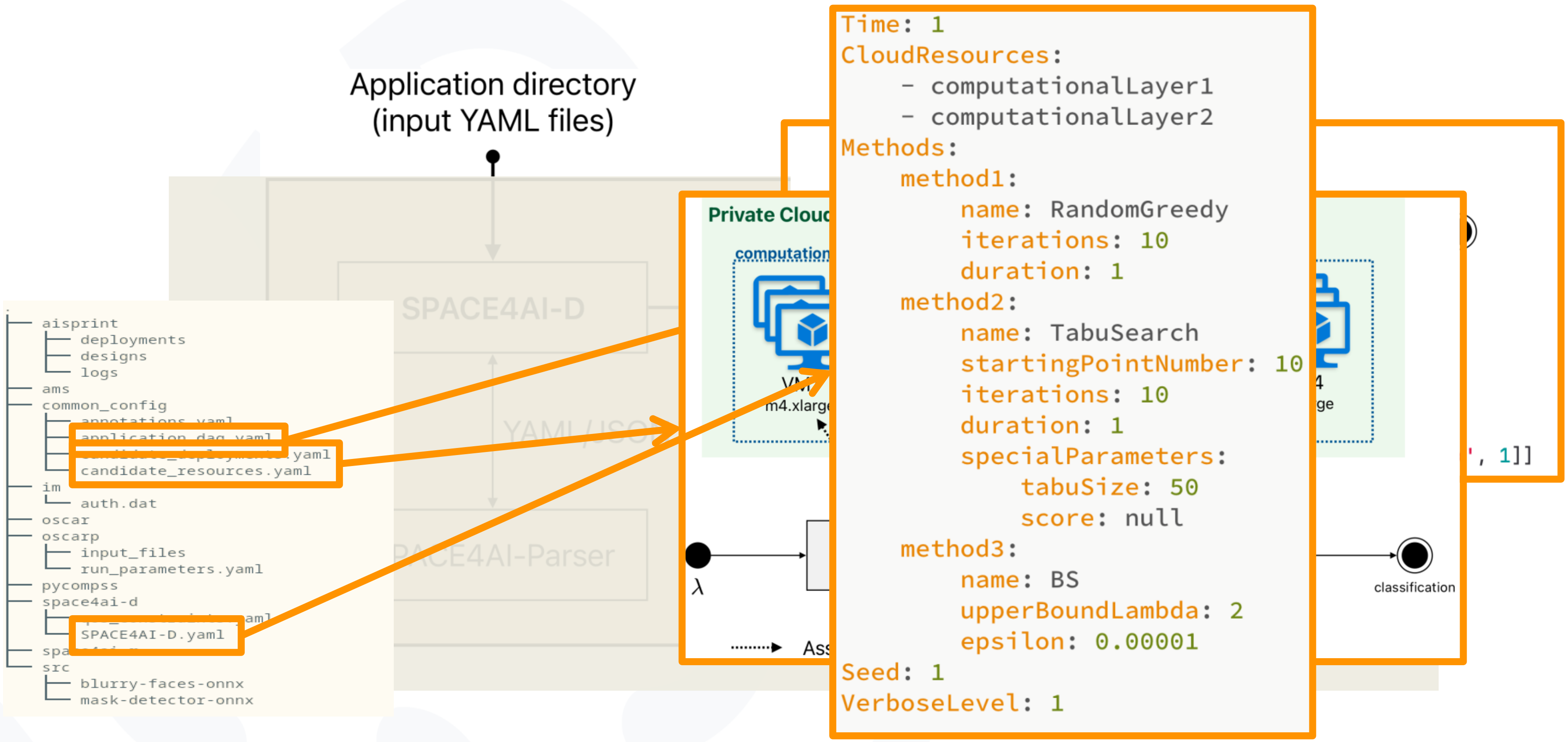
[2] B. Kopras, B. Bossy, F. Idzikowski, P. Kryszkiewicz, and H. Bogucka. Task allocation for energy optimization in fog computing networks with latency constraints. IEEE Transactions on Communications, 70(12):8229–8243, 2022.

[3] I. Cohen, C. F. Chiasserini, P. Giaccone, and G. Scalosub. Dynamic service provisioning in the edge-cloud continuum with bounded resources. IEEE/ACM Transactions on Networking, pages 1–16, 2023.

SPACE4AI-D

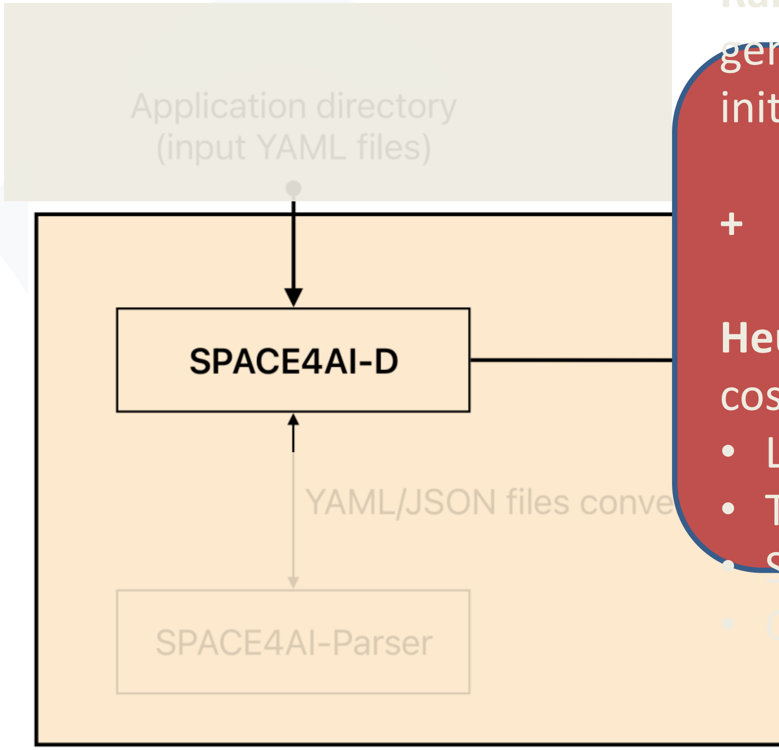

Application directory
(input YAML files)
SPACE4AI-D
Production deployment
YAML file
YAML/JSON files conversion
SPACE4AI-Parser

SPACE4AI-D



Application directory
(input YAML files)
SPACE4AI-D
YAML / JSON
SPACE4AI-Parser
aisprint
deployments
designs
logs
ams
common_config
application_dag.yaml
candidate_resources.yaml
im
auth.dat
oscar
oscarp
input_files
run_parameters.yaml
pycompss
space4ai-d
SPACE4AI-D.yaml
src
blurry-faces-onnx
mask-detector-onnx
Private Cloud
computation
m4.xlarge
λ
classification
Time: 1
CloudResources:
- computationalLayer1
- computationalLayer2
Methods:
method1:
name: RandomGreedy
iterations: 10
duration: 1
method2:
name: TabuSearch
startingPointNumber: 10
iterations: 10
duration: 1
specialParameters:
tabuSize: 50
score: null
method3:
name: BS
upperBoundLambda: 2
epsilon: 0.00001
Seed: 1
VerboseLevel: 1

SPACE4AI-D



Random Greedy (RG) algorithm to generate a pool of good-quality initial solutions

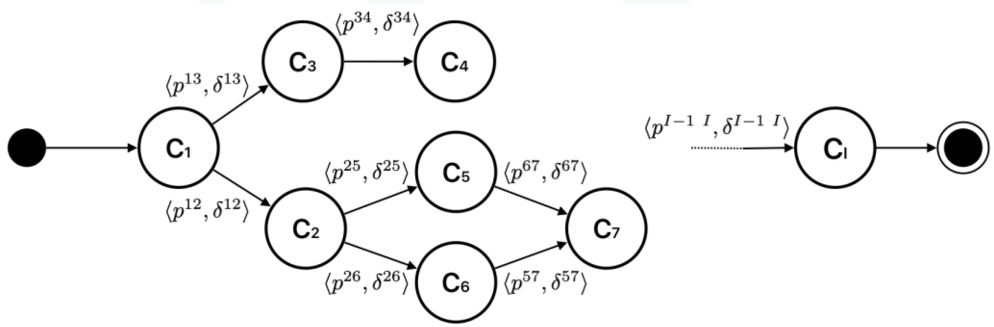
+

Heuristic algorithms to reduce the costs:

- Local Search (LS)
- Tabu Search (TS)
- Simulated Annealing (SA)
- Genetic Algorithms (Gas)

Application components model and QoS requirements

AI applications are modeled as **Directed Acyclic Graphs**



Node: AI application component

Node label: incoming load

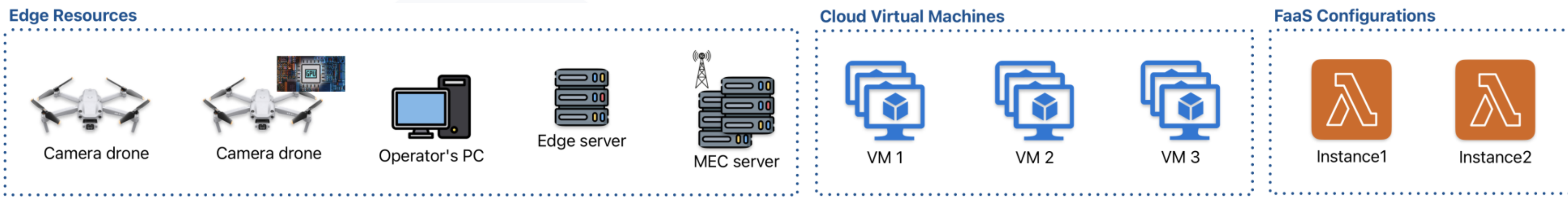
Edge: precedence relation between components

Edge label: <transition probability, data transfer>

Performance metric: response time

- **Local QoS constraints:** related to single components
- **Global QoS constraints:** related to sequences of consecutive components

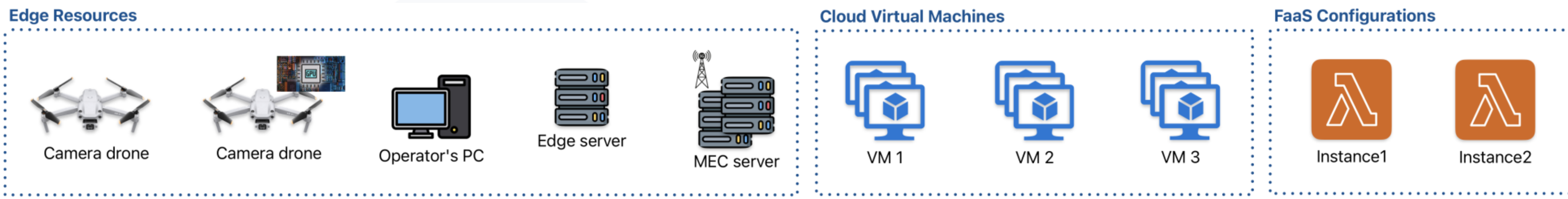
Resources model and system costs



- Edge devices, Cloud VMs and FaaS configurations, grouped in **computational layers** and characterized by different **memory capacity**
- Communications happen through **network domains** with different **access delay & bandwidth**

- **Edge costs:** amortized investment costs
- **Cloud VM costs:** per-second costs according to Cloud providers pricing models
- **FaaS costs:** GB-second costs depending on memory size, functions duration, total number of invocations

Resources model and system costs



Response time computation:

- **Edge & Cloud VMs:** demanding time without resource contention & individual **M/G/1** models
- **FaaS:** average execution time for each component according to [4]

OR

- **Machine Learning-based performance models**

+ Network delays due to data transmissions

Percentage error between 10% and 30%

[4] N. Mahmoudi and H. Khazaei, "Performance Modeling of Serverless Computing Platforms," in IEEE Transactions on Cloud Computing, vol. 10, no. 4, pp. 2834-2847, 1 Oct.-Dec. 2022, doi: 10.1109/TCC.2020.3033373.

Experimental setup

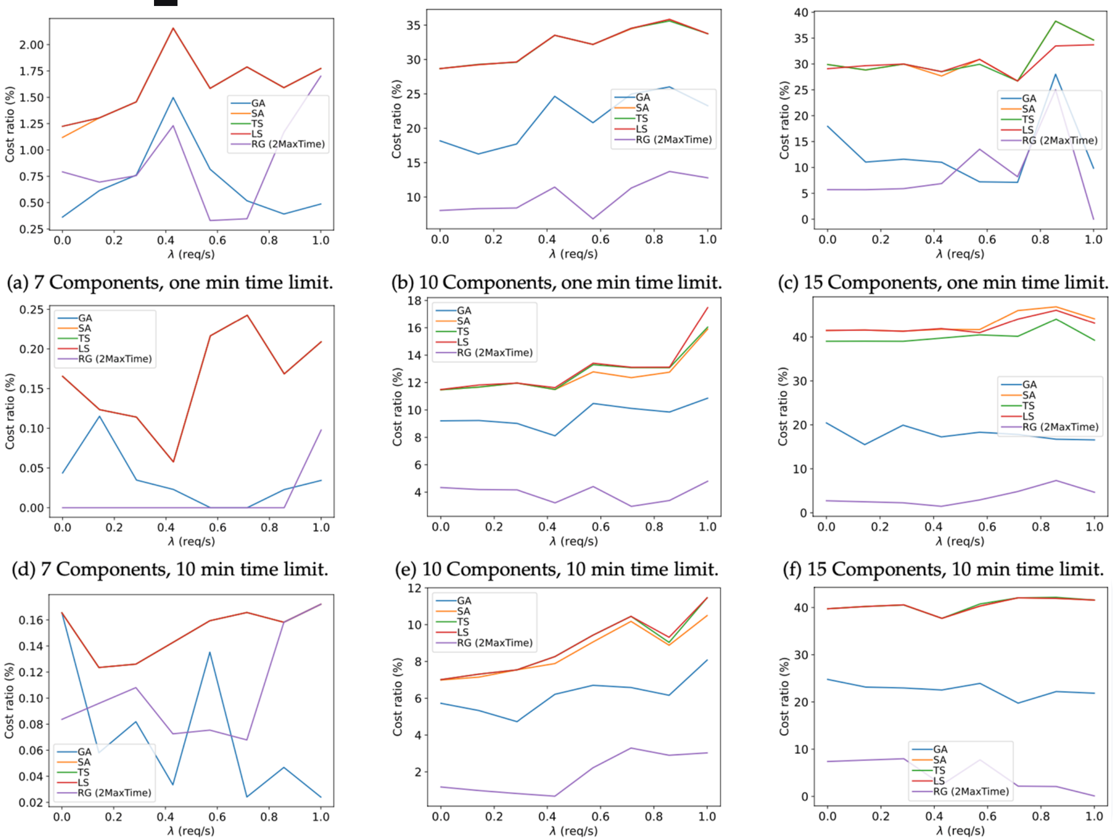
- Comparison between heuristic methods
- Comparison with the state of the art
- 3 scenarios at different scales:

Scenario	#Components	#Nodes in Computational Layers (CL)							#Local and global constraints
		CL_1	CL_2	CL_3	CL_4	CL_5	CL_6	CL_7	
1	7	Edge: 1	Edge: 3	VM: 4	VM: 4	FaaS: 3	-	-	3, 3
2	10	Edge: 1	Edge: 4	Edge: 4	VM: 4	VM: 4	FaaS: 4	-	4, 4
3	15	Edge: 1	Edge: 4	Edge: 5	VM: 5	VM: 5	VM: 5	FaaS: 5	5, 5

- Average percentage cost ratio over 10 random instances:

$$Cost\ ratio = \frac{SpecifiedMethod\ cost\ - OtherMethod\ cost}{OtherMethodcost} \times 100$$

Experimental results: Comparison between heuristic methods



(a) 7 Components, one min time limit. (b) 10 Components, one min time limit. (c) 15 Components, one min time limit.
(d) 7 Components, 10 min time limit. (e) 10 Components, 10 min time limit. (f) 15 Components, 10 min time limit.
(g) 7 Components, 30 min time limit. (h) 10 Components, 30 min time limit. (i) 15 Components, 30 min time limit.

***Heuristics*:**
Run RG (MaxTime) + Heuristic (MaxTime)

$$Cost\ ratio = \frac{RG\ cost(MaxTime)\ - Heuristics\ cost}{Heuristics\ cost} \times 100$$

LS, TS and SA obtain similar or better results compared with the RG and GA. LS is the best on average.

Experimental results: Comparison with a state-of-the-art method

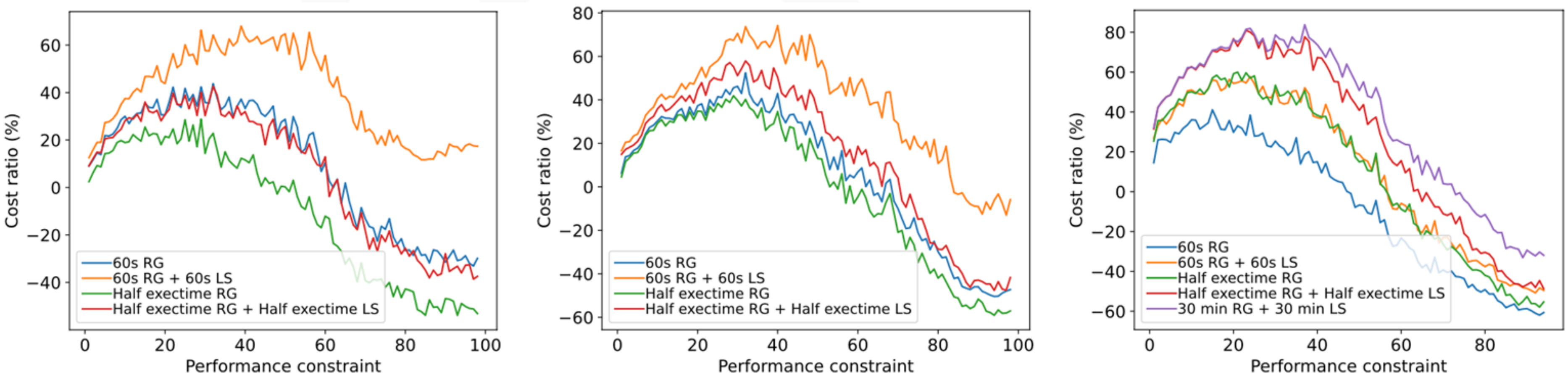
Algorithm: Best Cost Under Performance Constraint (BCPC)[1]

$$Cost\ ratio = \frac{\text{BCPC}cost\ \ - LS\ cost}{LS\ cost} \times 100$$

[1]C. Lin and H. Khazaei, "Modeling and Optimization of Performance and Cost of Serverless Applications," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 3, pp. 615-632, 1 March 2021, doi: 10.1109/TPDS.2020.3028841.

In the worst case, LS gains:

- time limit = exec time of BCPC: 27%
- time limit = one hour: 36%



(a) 7 components. (b) 10 components. (c) 15 components.

References & Links

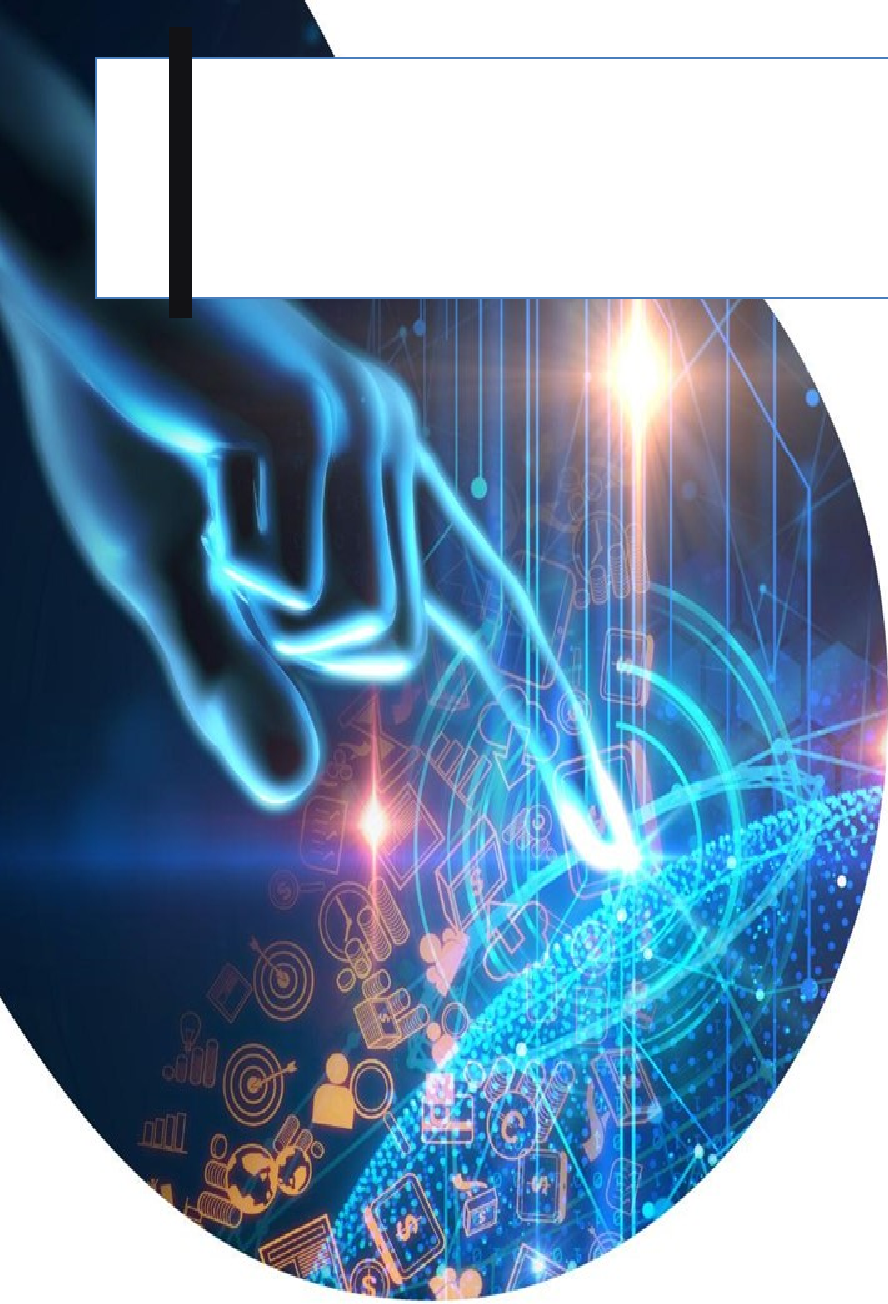
[1] C. Lin an[redacted]H. Khazaei, "Modeling and Optimization of Performance and Cost of Serverless Applications," in IEEE Transactions on Parallel and Distributed Systems, vo[redacted]32, no. 3, pp. 615-632, 1 March 2021, doi: 10.1109/TPDS.2020.3028841.

[2] B. Kopras, B. Bossy, F. Idzikowski, P. Kryszkiewicz, and H. Bogucka. Task allocation for energy optimization in fog computing networks with latency constraints. IEEE Transactions on Communications, 70(12):8229–8243, 2022.

[3] I. Cohen, C. F. Chiasserini, P. Giaccone, and G. Scalosub. Dynamic service provisioning in the edge-cloud continuum with bounded resources. IEEE/ACM Transactions on Networking, pages 1–16, 2023.

[4] N. Mahmoudi and H. Khazaei, "Performance Modeling of Serverless Computing Platforms," in IEEE Transactions on Cloud Computing, vol. 10, no. 4, pp. 2834-2847, 1 Oct.-Dec. 2022, doi: 10.1109/TCC.2020.3033373.

[5] H. Sedghani, F. Filippini, and D. Ardagna, “A Random Greedy based Design Time Tool for AI Applications Component Placement and Resource Selection in Computing Continua,” in IEEE International Conference on Edge Computing, EDGE 2021, Chicago, IL, USA, September 5-10, 2021, IEEE, 2021, pp. 32–40. doi: 10.1109/EDGE53862.2021.00014.



SPACE4AI-R

System PerformAnce and Cost Evaluation on Cloud for AI applications Runtime

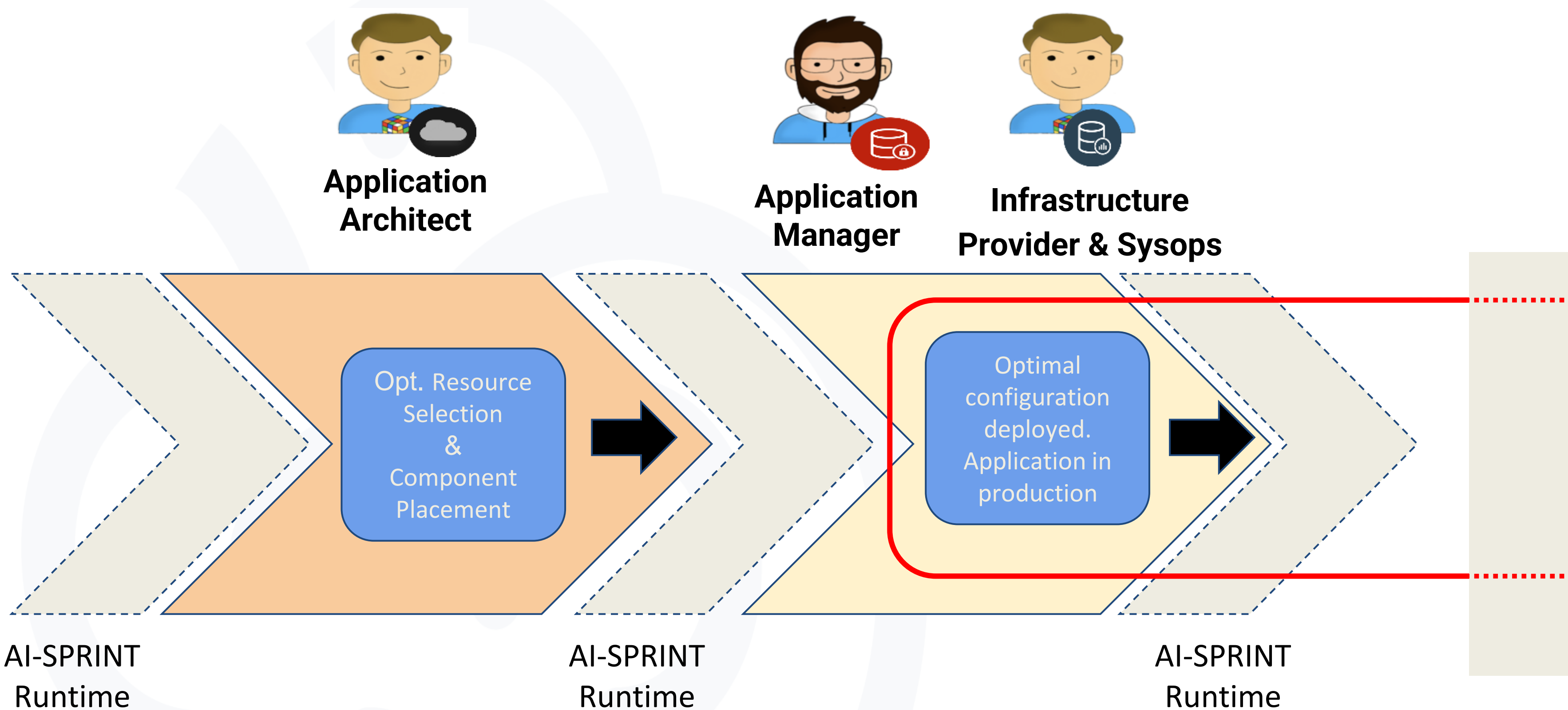
Federica Filippini

federica.filippini@polimi.it

Giuseppe Caccia

caccia@cefriel.it

AI Appl[redacted]cation Runtime Management



Optimal Component Placement & Resource Selection

Problem solved:

- **Automatic runtime reconfiguration** of resources and components placement to **minimize costs** and follow **workload fluctuations**
- Cope with **technology constraints**, **performance** and **privacy requirements**

Motivations:

- **Workload fluctuations** lead to resources **saturation or underutilization**
- The current **production deployment** needs to be continuously monitored and **adapted at runtime**

State Of The Art

- [1] presents an ML-based auto-scaling system that can behave proactively or reactively to adjust the number of Edge nodes in response to workload changes
- [2] addresses the service offloading and placement in the Computing Continuum through a greedy algorithm based on the online demands prediction
- [3] propo … loads in differen … system states a …
- [4] prop … ion of AI workflo …

SPACE4AI-R is **one of the first proposals** to consider **resource contention** in determining the optimal component placement **for AI applications**

SPACE4AI-R considers **multiple candidate neural network deployments** given by the possibility of **partitioning components at different layers** according to **network and load conditions**

[1] Thiago Pereira …

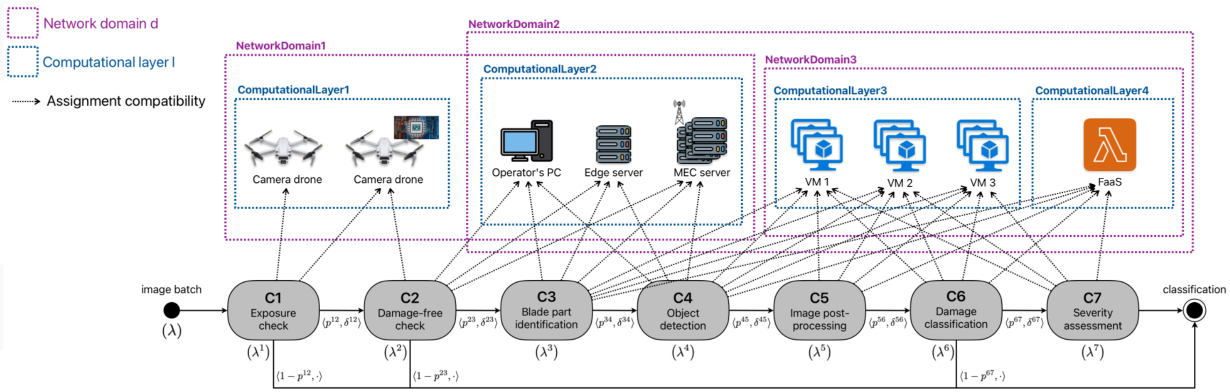
[2] Yeting Guo et a … 37, 11, 8523–8547.

[3] Xun Shao et al. 20 … 02, 927– 940.

[4] Qianlin Liang et al. 2023. Model-driven cluster resource management for ai workloads in edge clouds. ACM Trans. Auton. Adapt. Syst., 18, 1, Article 2.

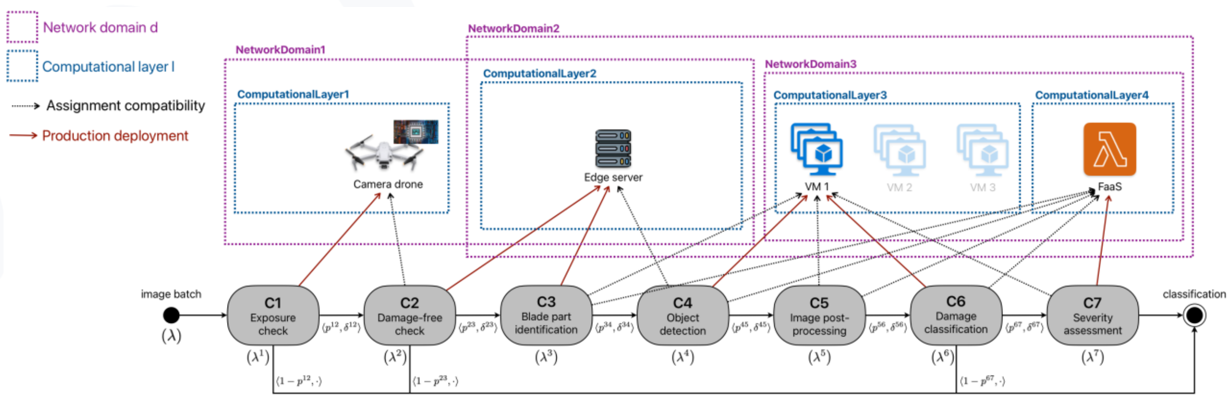
Resource selection and component placement problem at design-time:

- (maximum) expected workload
- Edge devices, Cloud VMs, FaaS
- minimum-cost solution
- performance guarantees



Runtime adaptation:

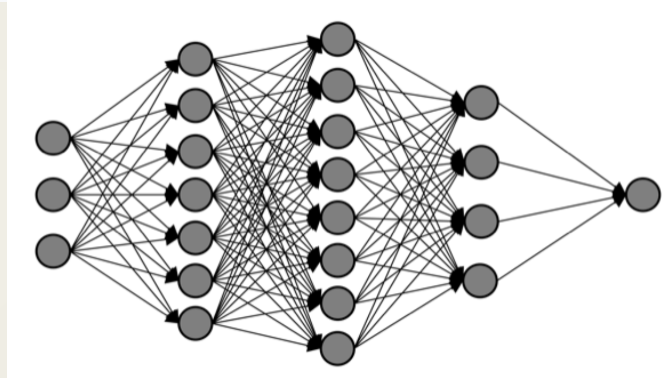
- varying workload profile
- resource scaling
- component migration
- periodic execution



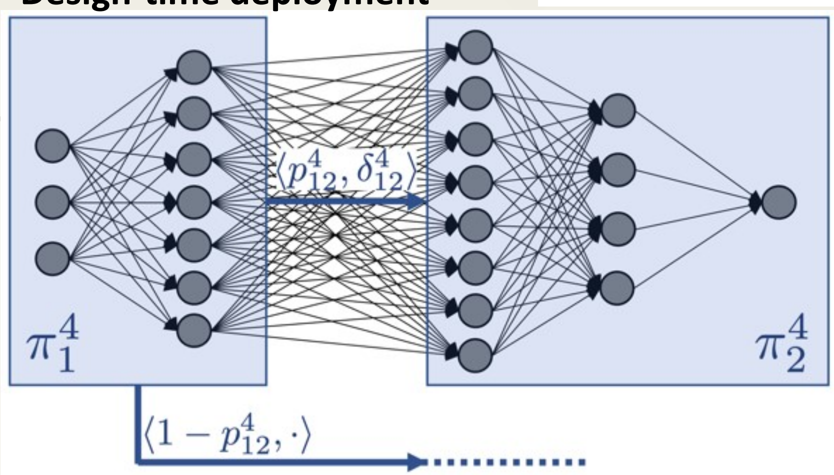
Random Search & Stochastic Local Search

Resource selection and component placement problem at design-time:

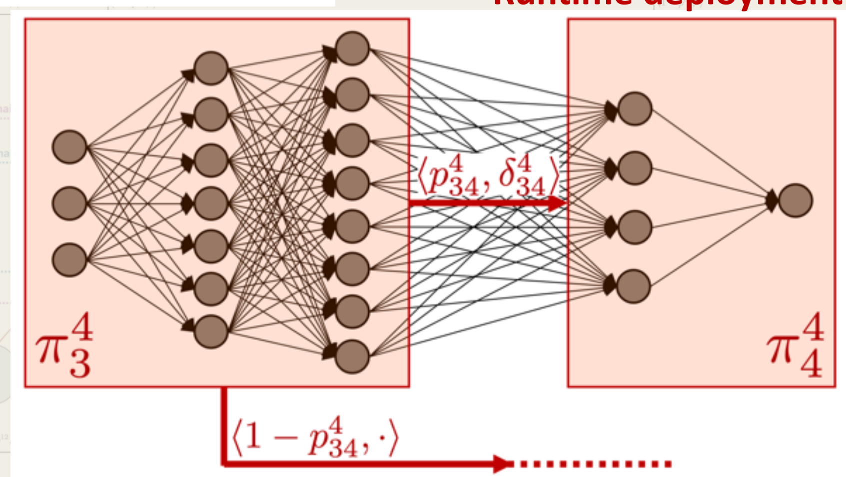
- (maximum) expected workload
- Edge devices, Cloud VMs,



Design-time deployment

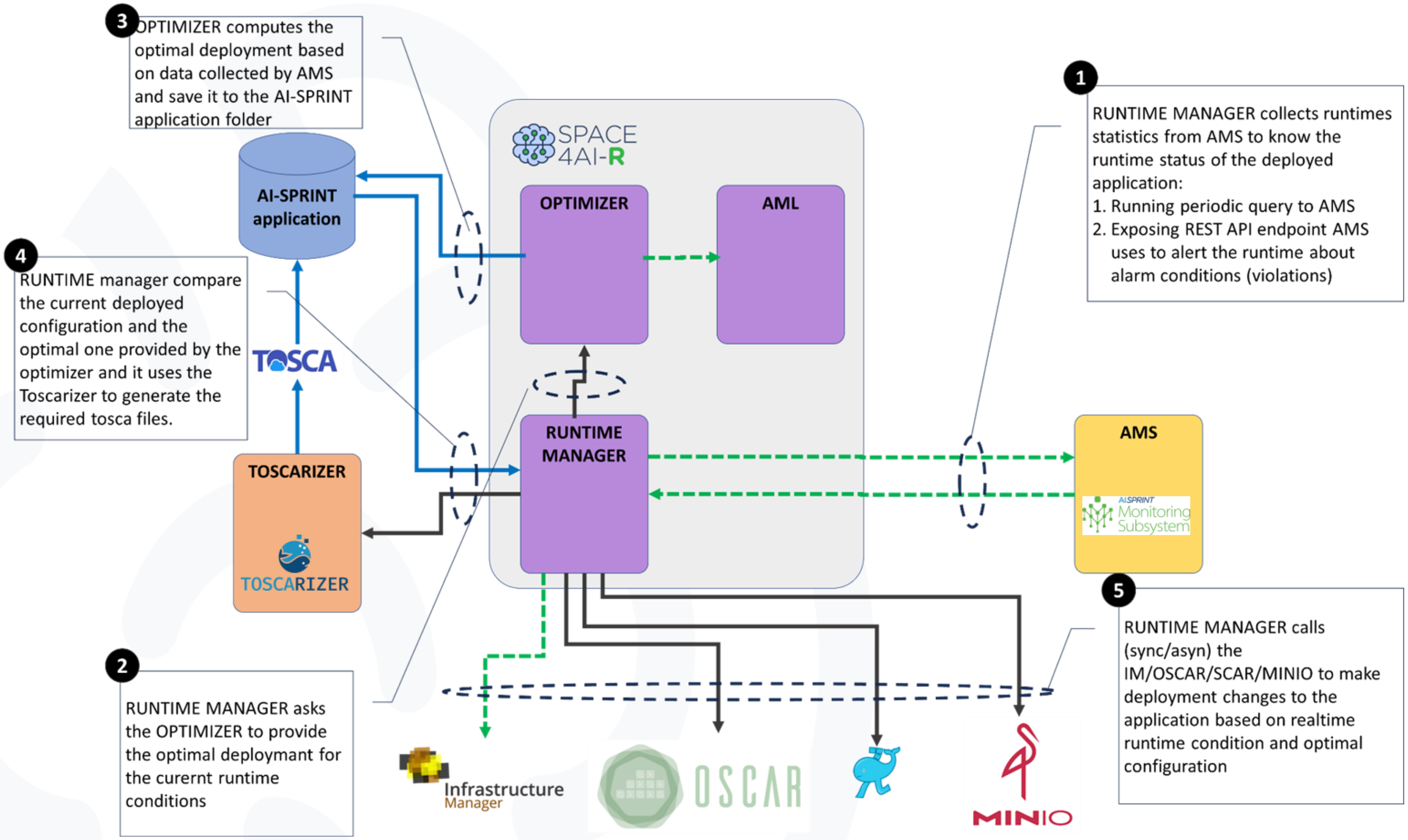


Runtime deployment



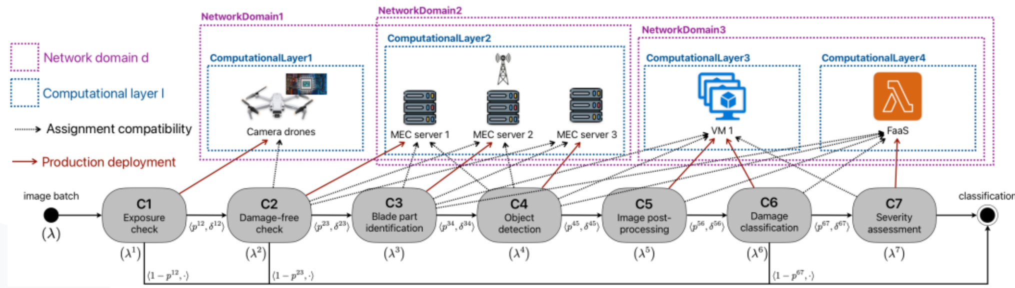
Stochastic Local Search

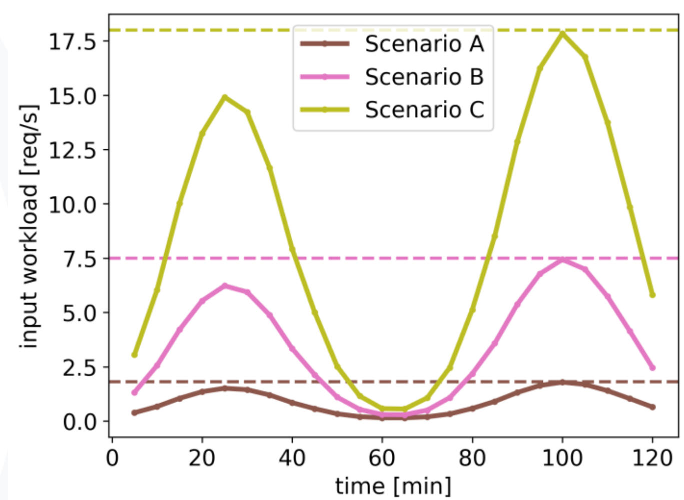
AI-SPRINT runtime architecture



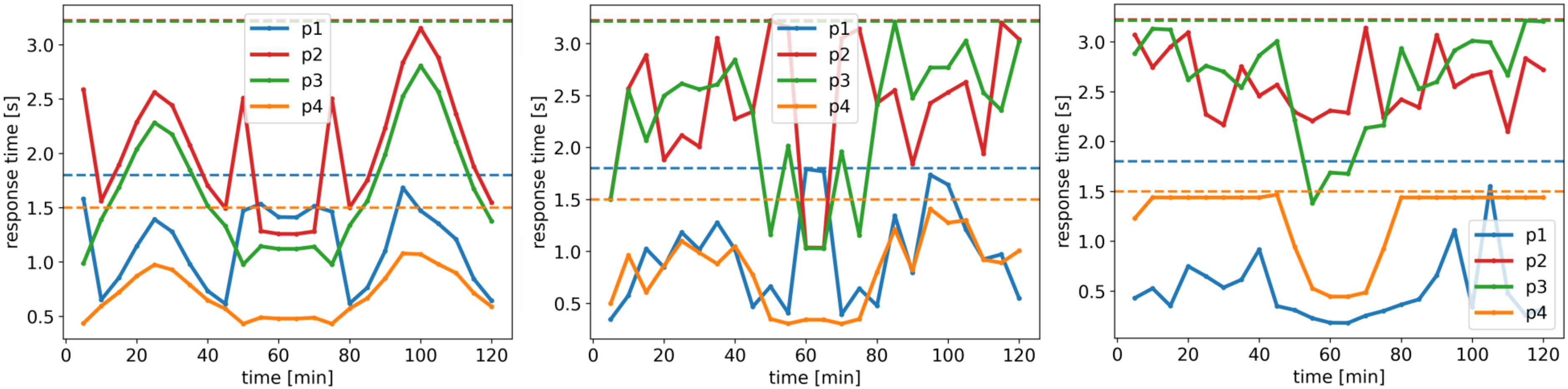
- **Mainteinance & inspection use-case**
- **7** components; **4** computational layers
- **Three scenarios:**
  - user's PC at the second computational layer; max workload = 1.8 req/s
  - 2 servers in the user's van; max workload = 7.5 req/s
  - 3 Mobile Edge Computing servers accessed from 5G tower; max workload = 18 req/s
- **Cloud resou**

Name	Cost [$/h]	Number of Instances
VM1	0.41	n=4
VM2	1.53	n=3
VM3	1.99	n=3
VM4	3.16	n=3





Experimental results: Use-case analysis



Scenario A Scenario B Scenario C

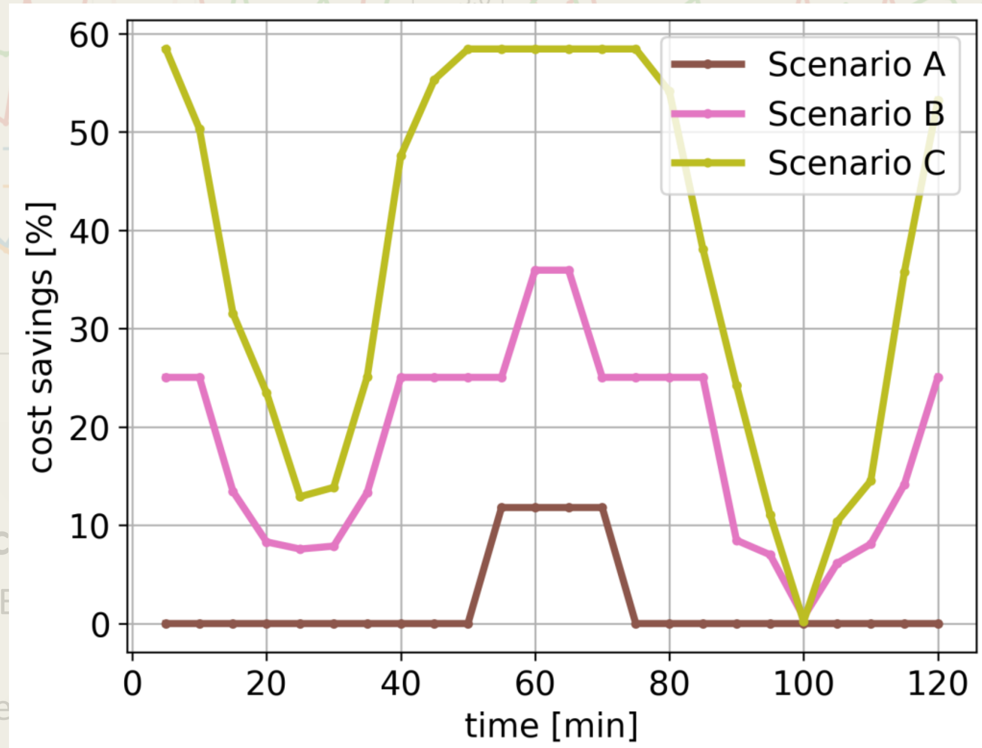
- In each scenario, **the four global QoS constraints are always satisfied**
  - **A:** when the workload is minimum, C_1 and C_2 run on the drone, increasing its utilization
  - **B:** it is more difficult for SPACE4AI-R to determine feasible solutions due to the higher workload
  - **C:** the response times are more stable; only the fourth path is always closer to the threshold

Experimental results: Use-case analysis

Cost saving of dynamic reconfigurations over a static placement keeping fixed the design-time solution for the entire application execution:

- SPACE4AI-R solution is always at least good as the design-time one
- **Up to 60%** cost reduction when the workload is at minimum

Average time to solution between 0.39 and 0.43 seconds



Experimental setup: Scalability analysis

- Three scenarios
- Variable number of components and resources

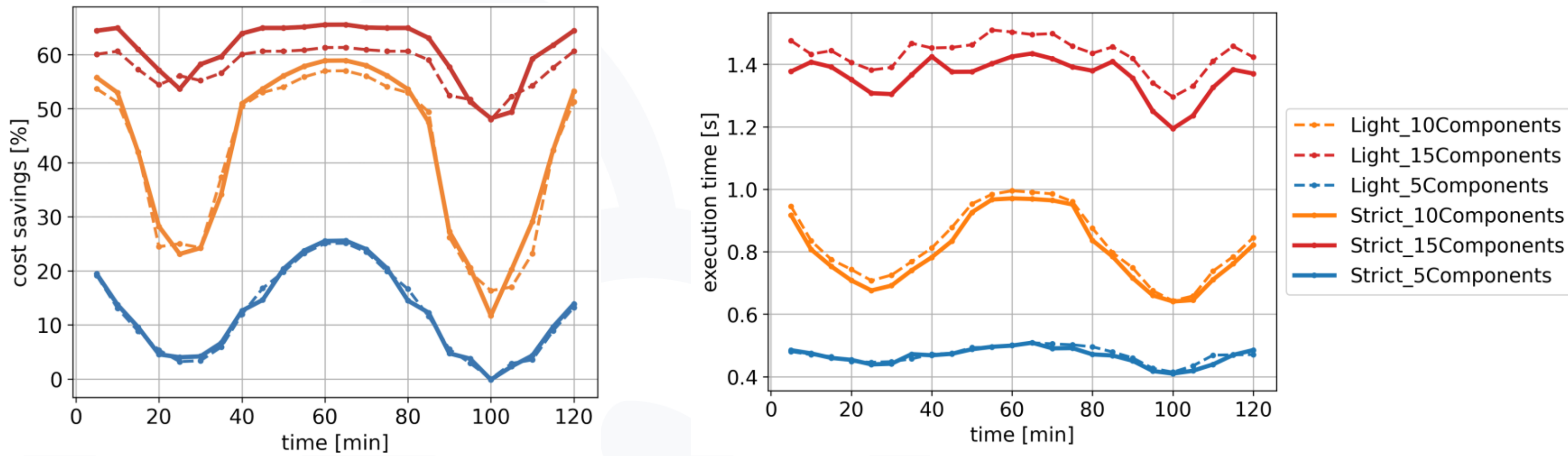
- Randomly-generated service demands:
  - in [1, 5]s for Edge resources
  - in [0.5, 2]s for Cloud VMs
  - in [2, 5]s for cold and warm FaaS requests

- Variable number of (light or strict) local and global constraints
  - light: in [50, 100]s and [200, 300]s
  - strict: in [7, 10]s and [20, 25]s

Scenario		1	2	3
Number of components		5	10	15
Type and number of resources in each layer	CL_1	Drone: 1	Drone: 1	Drone: 1
	CL_2	Edge: 2	Edge: 4	Edge: 5
	CL_3	VM: 3	Edge: 4	Edge: 5
	CL_4	FaaS: 2	VM: 4	VM: 5
	CL_5	-	VM: 4	VM: 5
	CL_6	-	FaaS: 4	VM: 5
	CL_7	-	-	FaaS: 5
Number of local, global constraints		3,3	4,4	5,5

10 randomly-generated DAGs with branches in each scenario

Experimental results: Scalability analysis



- SPACE4AI-R always guarantees lower costs than the static placement
- **cost savings > 60%** for larger systems

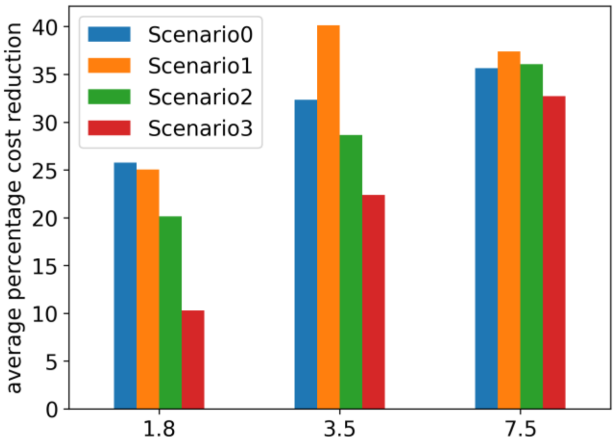
- The **time to solution is between 0.4 and 1.5s**; almost **100x faster** than SPACE4AI-D

Experimental setup: Comparison with a state-of-the-art method

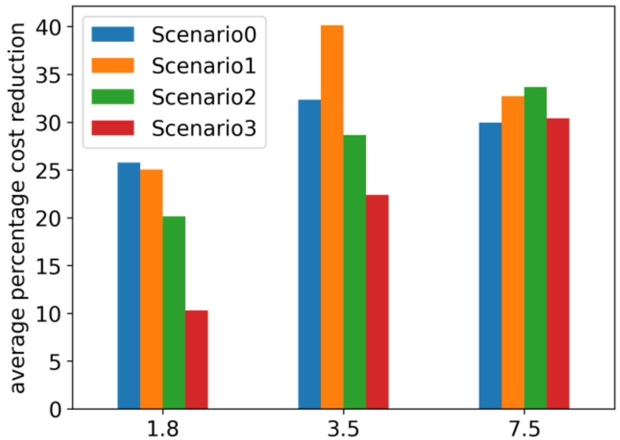
- Comparison with **Utilization Heuristic** (UHEUR) [5,6,7]
  - **Scaling** actions to keep the utilization within **[U_{min}, U_{max}]**
- **Four scenarios**
  - 2, 4, 8 or 10 components
  - reserved Edge and Cloud VMs with 9 or 10 available instances
  - service demand in [0.2, 0.6] *s* on Edge and [0.1, 0.55] *s* on Cloud
  - a local constraint on each component, threshold between *2x* and *2.5x* the demand
- **Maximum workload:** 1.8 *req/s*, 3.5 *req/s*, 7.5 *req/s*
- **Utilization intervals:**
  - [U_{min}, U_{max}] = [40, 50]%
  - [U_{min}, U_{max}] = [50, 60]%
  - [U_{min}, U_{max}] = [60, 80%]

[5] A. Wolke and G. Meixner. "Twospot: A cloud platform for scaling out web ap- plications dynamically". In Towards a Service-Based Internet: Third European Conference, ServiceWave 2010, Ghent, Belgium, December 13-15, 2010. Proceedings 3, pages 13–24. Springer, 2010.
[6] X. Zhu, D. Young, et al. "1000 islands: an integrated approach to resource management for virtualized data centers". Cluster Computing, 12:45–57, 2009.
[7] AWS Elastic Beanstalk. https://aws.amazon.com/elasticbeanstalk/, 2023. Accessed: (24/10/2023).

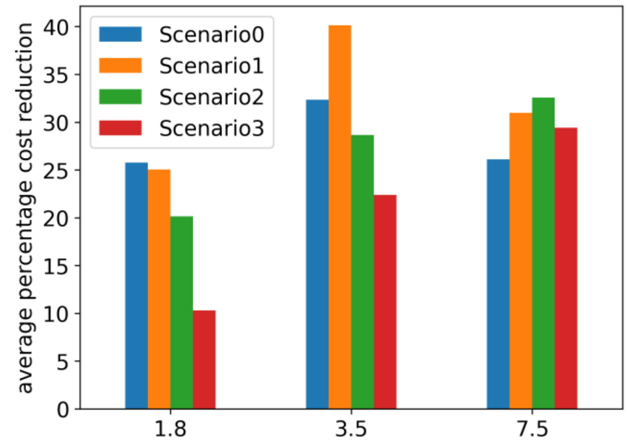
Expe[redacted]imental results: Comparison with a state-of-the-art method



$[U_{min}, U_{max}] = [40, 50]\%$



$[U_{min}, U_{max}] = [50, 60]\%$



$[U_{min}, U_{max}] = [60, 80]\%$

UHEUR incurs in a number of response times constraints violations between 1.8% and 35% when the load is high

References & Links

[1] Thiago Pereira da Silva et al. 2022. “Online machine learning for auto-scaling in the edge computing”. Pervasive Mob., 87, 101722.

[2] Yeting Guo et al. 2022. “PARA: Performability-aware resource allocation on the edges for cloud-native services”. Int. J. Intell. Syst., 37, 11, 8523–8547.

[3] Xun Shao et al. 2023. An Online Orchestration Mechanism for General- Purpose Edge Computing. IEEE Trans. Serv. Comput., 16, 02, 927– 940.

[4] Qianlin Liang et al. 2023. Model-driven cluster resource management for ai workloads in edge clouds. ACM Trans. Auton. Adapt. Syst., 18, 1, Article 2.

[5] A. Wolke and G. Meixner. “Twospot: A cloud platform for scaling out web ap- plications dynamically”. In Towards a Service-Based Internet: Third European Conference, ServiceWave 2010, Ghent, Belgium, December 13-15, 2010. Proceedings 3, pages 13–24. Springer, 2010.

[6] X. Zhu, D. Young, et al. “1000 islands: an integrated approach to resource management for virtualized data centers”. Cluster Computing, 12:45–57, 2009.

[7] AWS Elastic Beanstalk. https://aws.amazon.com/elasticbeanstalk/, 2023. Accessed: (24/10/2023).

[8] F. Filippini, H. Sedghani, and D. Ardagna, “SPACE4AI-R: Runtime Management Tool for AI Applications Component Placement and Resource Selection in Computing Continua,” in 2023 IEEE/ACM 16th International Conference on Utility and Cloud Computing (UCC ’23), (to appear) 2023, pp. 1–7, isbn: 979-8-4007-0234-1/23/12. doi: 10.1145/3603166.3632560.