fog guru

TRAINING THE NEXT GENERATION OF EUROPEAN FOG COMPUTING EXPERTS

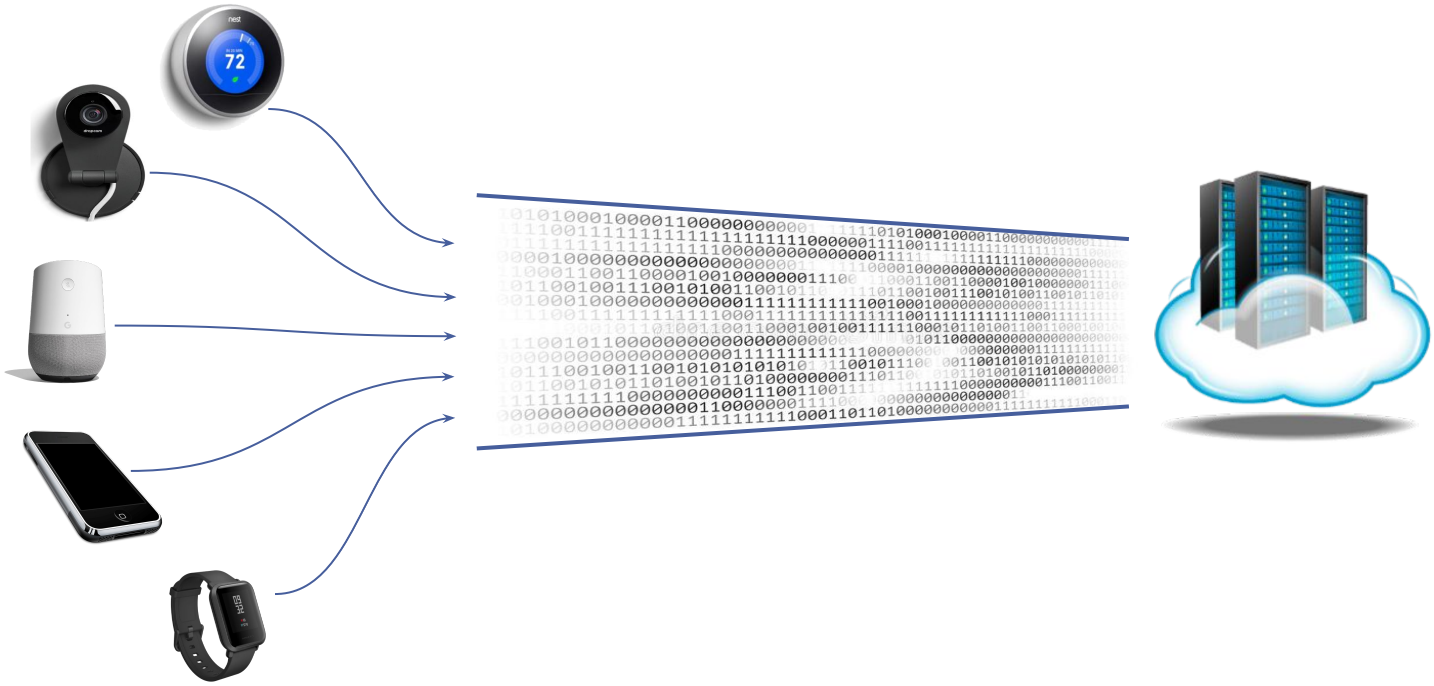
Easy-to-setup Fog computing testbed based on a RaspberryPi cluster for running data stream processing applications

HamidReza Arkian
Supervisor: Prof. Guillaume Pierre

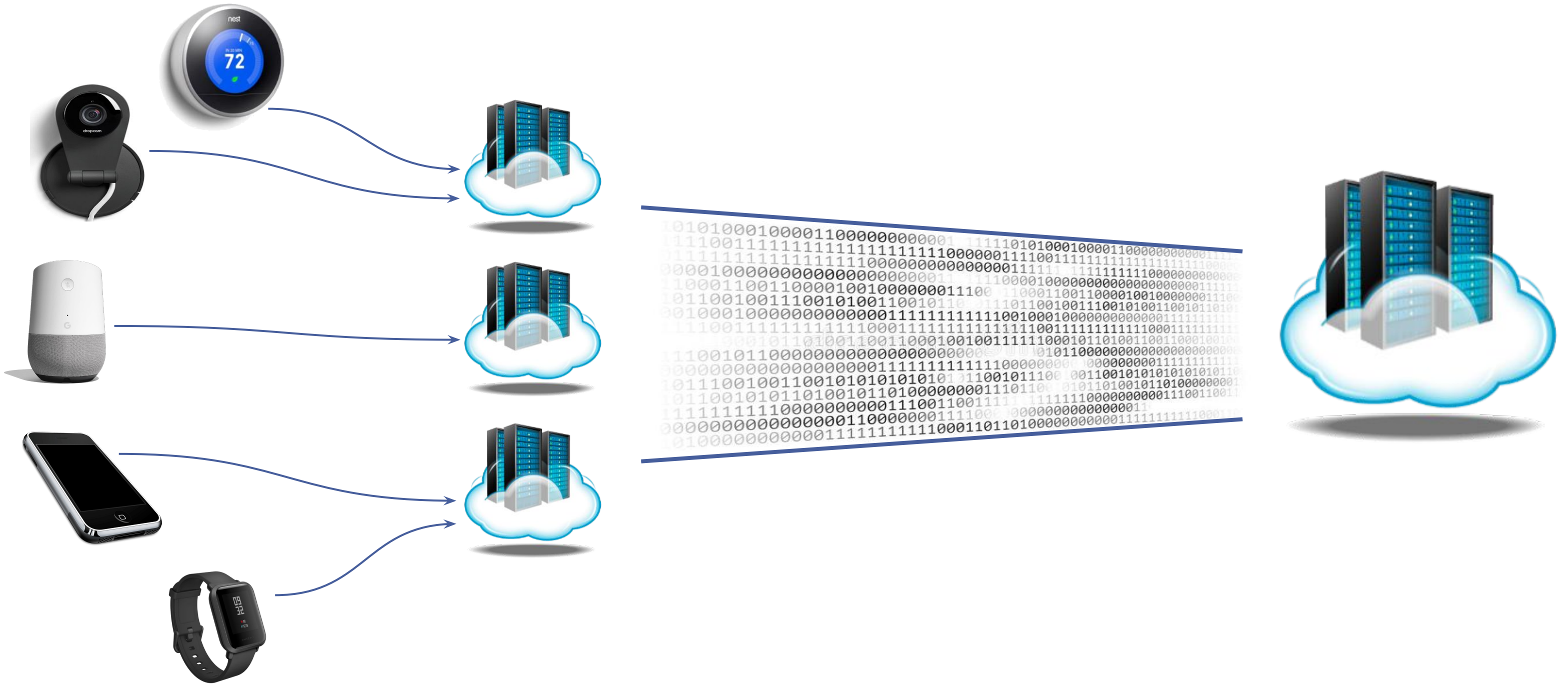
December 2020

European Commission

IoT-to-Cloud basic architecture



Fog-based architecture

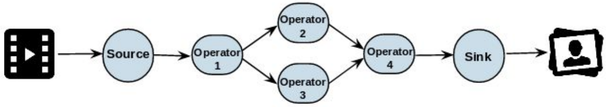


Data stream processing in Fog computing environments

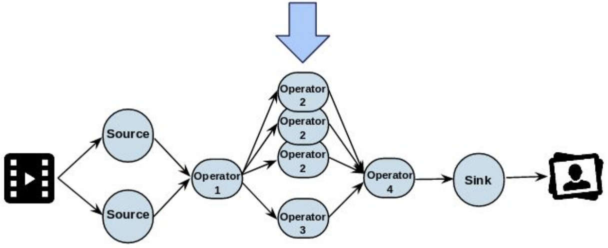
- Stream processing engines

STORM Flink Spark

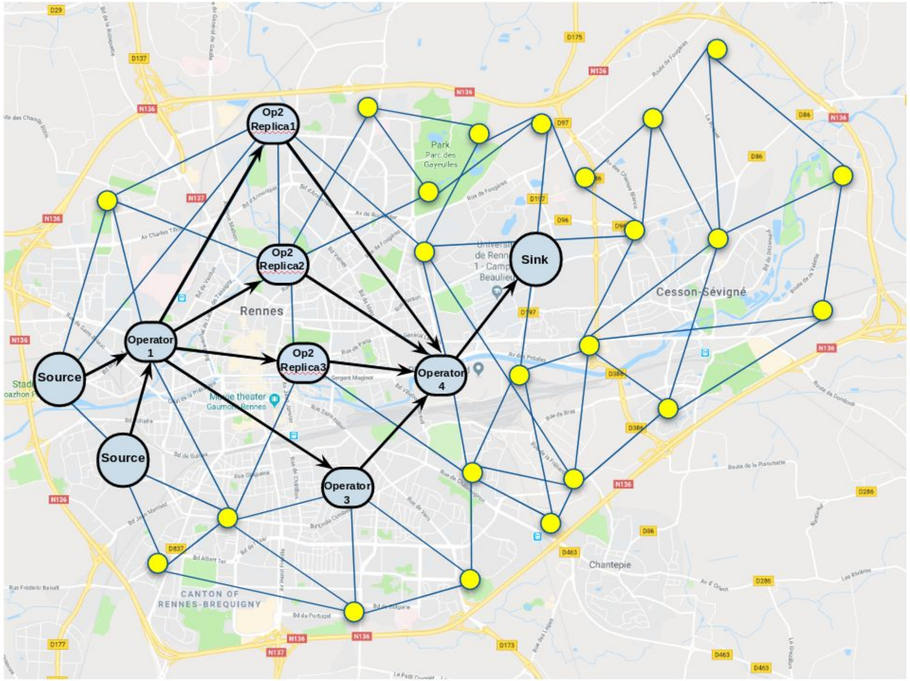
- Stream processing applications



Logical graph of DSP

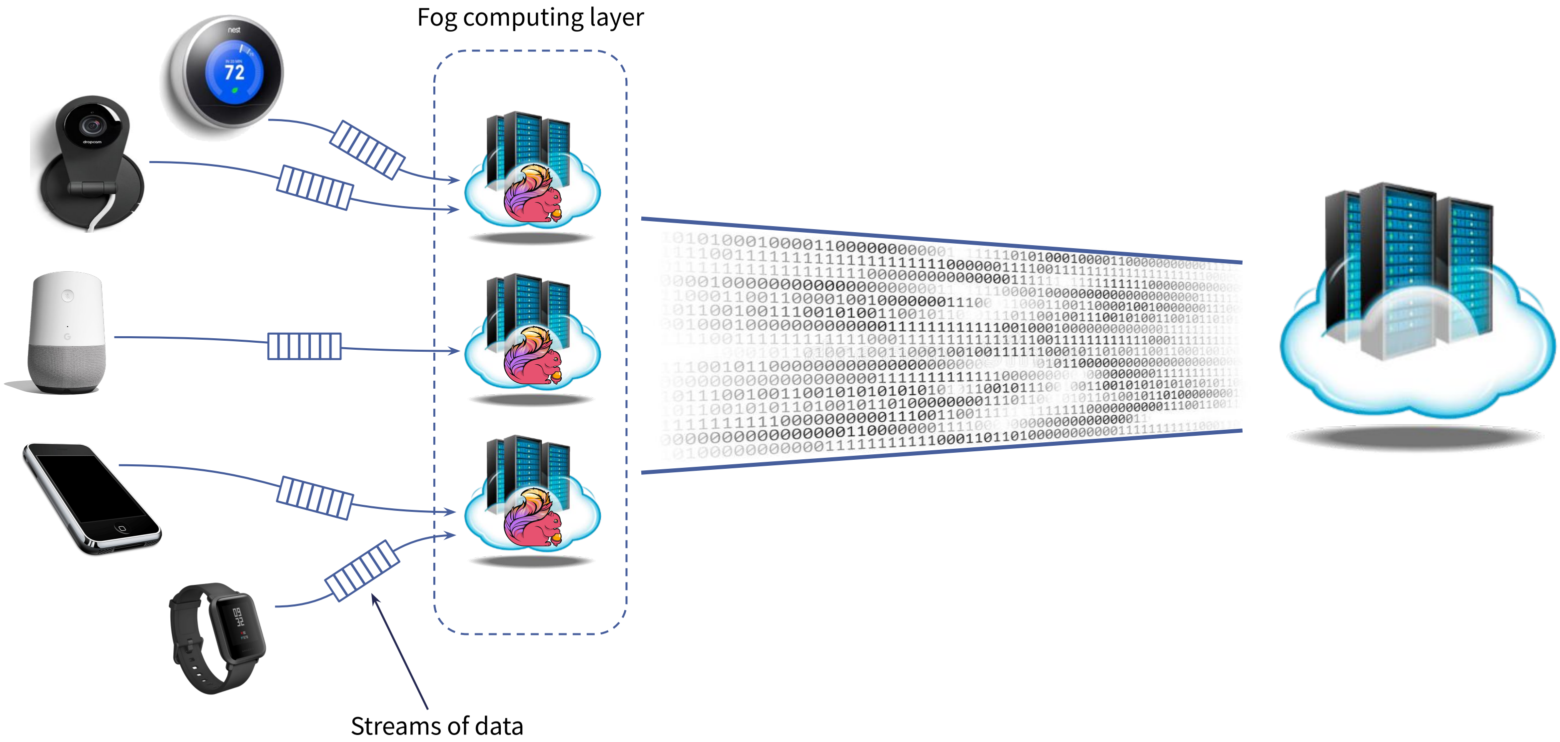


Workflow execution model



Deployment in Fog geo-distributed environment

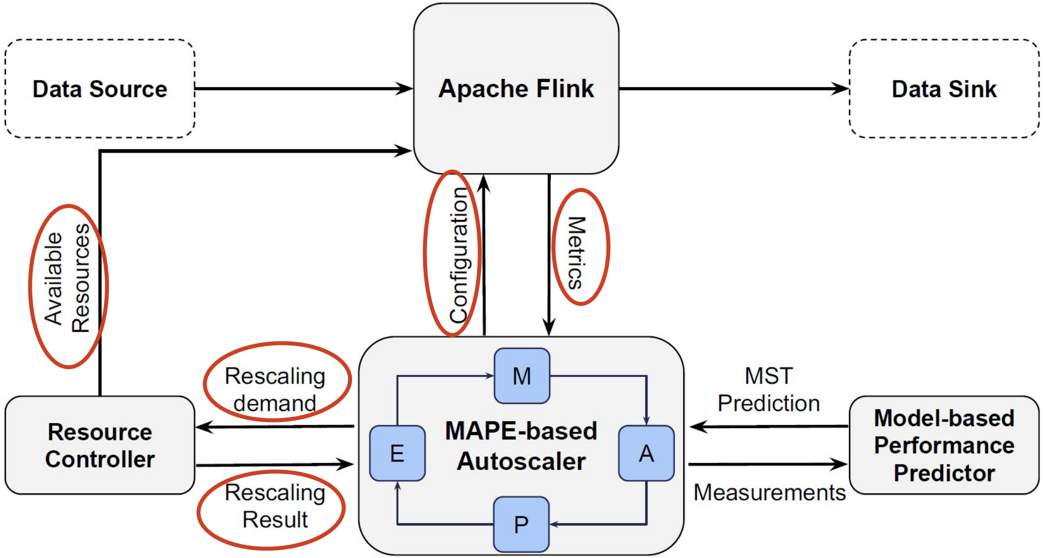
Fog-based architecture



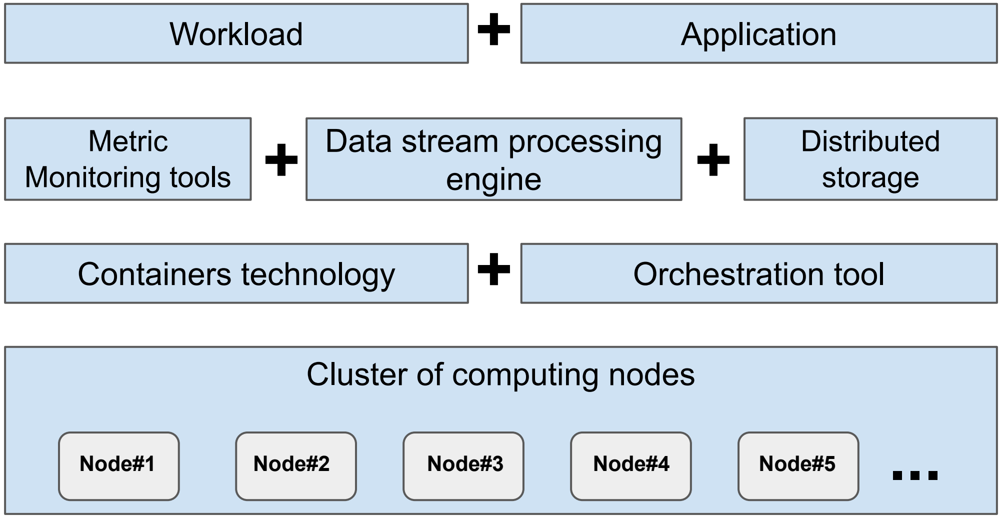
An overview on our work

- *Gesscale* (GEo-distributed Stream autoSCALEr)
  - an auto-scaler for stream processing applications in geo-distributed environments
  - Objective: to maintain a sufficient throughput (considering incoming workload) while using no more or less resources than necessary.

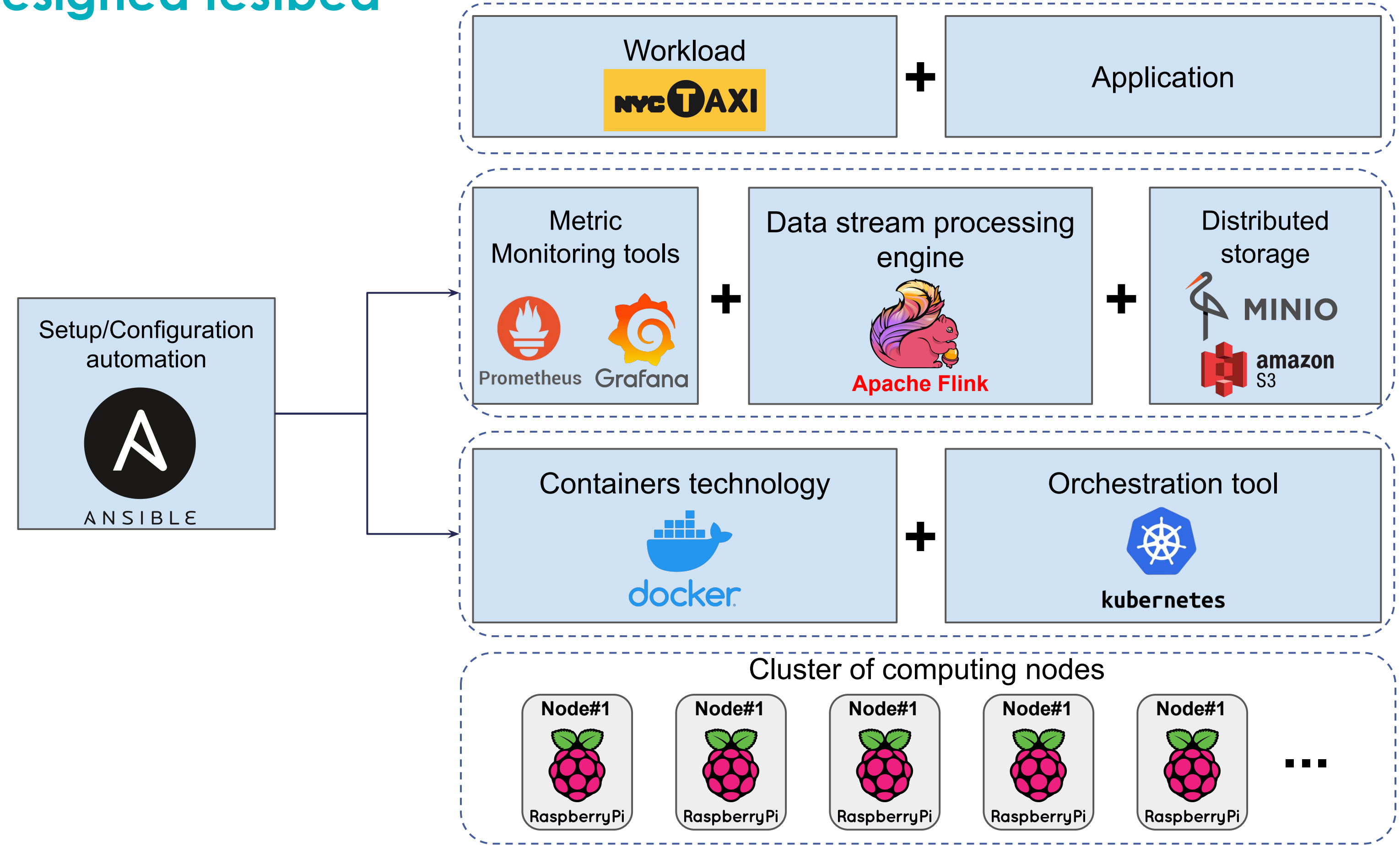
- *Gesscale* continuously monitors the workload and performance of the running system and dynamically adds or removes replicas to/from individual stream processing operators.



Required testbed

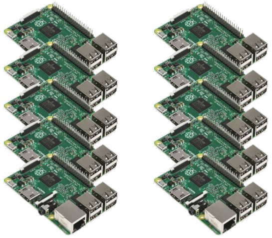


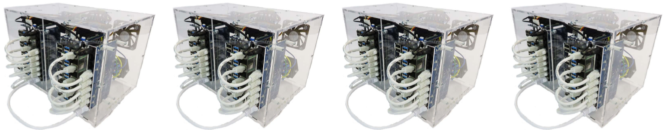
Designed testbed



RassperyPI-based infrastructure

- Cluster of 10 * RPI4
  - Powerful enough as a testbed.
  - Sclable (horizontal & vertical)





- Ansible bootstrap.yml:

```yaml
- hosts: localhost
  gather_facts: yes
  become: yes

  tasks:
    - name: Add our node names to hosts
      blockinfile:
        dest: /etc/hosts
        marker: "# {mark} ANSIBLE MANAGED BLOCK HOSTS"
        block: |
          10.188.180.183     pico1-0
          10.188.181.109     pico1-1
          10.188.180.50      pico1-2
          10.188.180.210     pico1-3
          10.188.180.198     pico1-4
          10.188.174.212     pico1-5
          10.188.175.213     pico1-6
          10.188.145.15      pico1-7
          10.188.176.11      pico1-8
          10.188.175.141     pico1-9

    - name: Add our ansible hosts setup
      blockinfile:
        dest: /etc/ansible/hosts
        marker: "# {mark} ANSIBLE MANAGED BLOCK HOSTS"
        block: |
          # Ungrouped
          pico1-[0:9]

          # cluster node
          [cluster]
          pico1-[0:9]

          # master
          [master]
          pico1-0

          # worker
          [worker]
          pico1-[1:9]
```

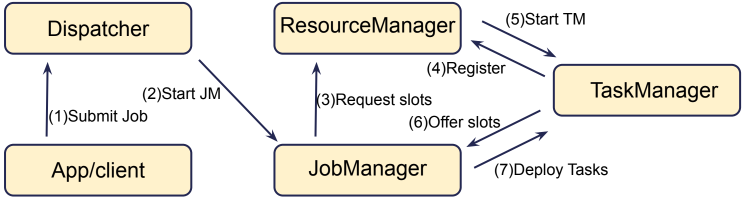
Using Ansible to install Docker & Kubernetes

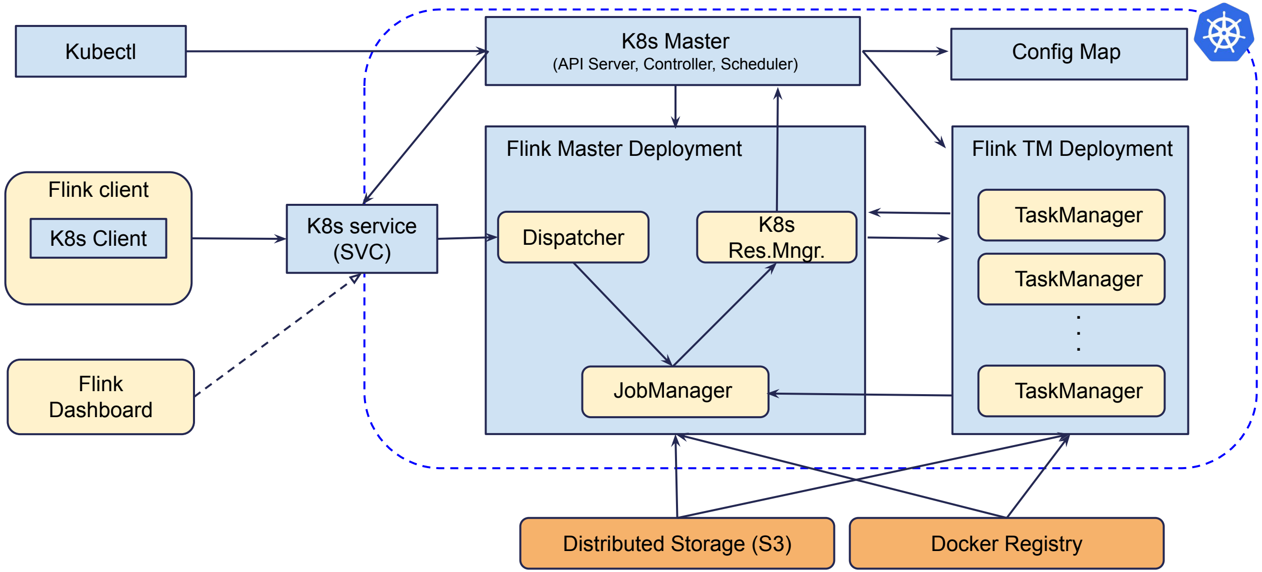
Ansible_Docker_K8s.yml:

```yaml
- name: Check to see if Docker is already installed
  shell: dpkg-query -W 'docker'
  ignore_errors: True
  register: is_docker

# Docker install. Skip if already installed
- block:
    - name: install latest docker.io
      apt:
        name: ['docker.io']
        state: present

    - name: Create docker daemon file
      blockinfile:
        dest: /etc/docker/daemon.json
        block: |
          {
            "exec-opts": ["native.cgroupdriver=systemd"],
            "log-driver": "json-file",
            "log-opts": {
              "max-size": "100m"
            },
            "storage-driver": "overlay2"
          }
        create: yes
        marker: ""
      ignore_errors: True

    - name: Make docker.service.d directory
      shell: "mkdir -p /etc/systemd/system/docker.service.d"

    - name: restart docker
      systemd:
        state: restarted
        daemon_reload: yes
        name: docker

    - name: hold docker.io so it's not upgraded
      shell: "apt-mark hold docker.io"

    - name: Append picocluster to docker Group
      user:
        name: picocluster
        groups: docker
        append: yes
      register: group
  when: is_docker is failed
```

```yaml
- name: Check to see if Kubernetes is already installed
  shell: dpkg-query -W 'kubeadm'
  ignore_errors: True
  register: is_kubernetes

# Kubernetes install. Skip if already installed
- block:
    - name: Install Kubernetes repository key
      shell: "curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -"

    - name: Add Kubernetes source for apt
      lineinfile:
        dest: /etc/apt/sources.list.d/kubernetes.list
        line: "deb http://apt.kubernetes.io/ kubernetes-xenial main"
        create: yes

    - name: Update cache to get kubernetes
      apt:
        update_cache: yes

    - name: Install Kubernetes
      apt:
        name: ['kubeadm=1.15.5-00', 'kubectl=1.15.5-00', 'kubelet=1.15.5-00', 'kubernetes-cni=0.7.5-00']
        state: present

    - name: hold kubelet kubeadm kubectl so they are not upgraded
      shell: "apt-mark hold kubelet kubeadm kubectl"
      register: kubernetes_install


  when: is_kubernetes is failed

- block:
    # Create Kubernetes cluster and save join command to file
    - block:
        - name: Init kubernetes
          command: "kubeadm init --pod-network-cidr 10.244.0.0/16"
          register: kube_init

        - name: Extract join command
          command: "kubeadm token create --print-join-command"
          register: join_command

        - name: Save join command
          local_action: copy content={{ join_command.stdout_lines | last  | trim }} dest="{{ join_command_location }}"

        - name: Copy join command to worker nodes
          synchronize:
            src: "{{ join_command_location }}"
            dest: "{{ join_command_location }}"
      when: "'master' in group_names"
```

Flink on K8s architecture





Deploying Flink on Kubernetes

Dockerfile:

```
FROM openjdk:8-jre-alpine

# Install requirements
RUN apk add --no-cache bash snappy libc6-compat


# Flink environment variables
ENV FLINK_HOME=/opt/flink
ENV PATH=$FLINK_HOME/bin:$PATH


# Install Flink
COPY flink-1.11.2 $FLINK_HOME


#Add flink group/user
RUN  addgroup -S flink && adduser -D -S -H -G flink -h $FLINK_HOME flink && \
  chown -R flink:flink $FLINK_HOME
WORKDIR $FLINK_HOME
```

- There is no manifest (no native support) for ARMv7 architecture in Flink docker-hub

Flink-configuration-configmap.yaml:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: flink-config
  labels:
    app: flink
data:
  flink-conf.yaml: |+
    jobmanager.rpc.address: flink-jobmanager
    taskmanager.numberOfTaskSlots: 1
    blob.server.port: 6124
    jobmanager.rpc.port: 6123
    taskmanager.rpc.port: 6122
    queryable-state.proxy.ports: 6125
    jobmanager.memory.process.size: 1024m
    taskmanager.memory.process.size: 1024m
    parallelism.default: 1
    metrics.reporter.prom.class: org.apache.flink.metrics.prometheus.PrometheusReporter
    taskmanager.network.detailed-metrics: true
    web.backpressure.refresh-interval: 1000
    state.backend: filesystem
    state.checkpoints.dir: s3://state/checkpoints
    state.savepoints.dir: s3://state/savepoints
    s3.path-style-access: true
    s3.endpoint: http://172.17.0.2:30090
    s3.access-key: minio
    s3.secret-key: minio123

  log4j-console.properties: |+
    # This affects logging for both user code and Flink
    rootLogger.level = INFO
    rootLogger.appenderRef.console.ref = ConsoleAppender
    rootLogger.appenderRef.rolling.ref = RollingFileAppender
    # Log all infos to the console
    appender.console.name = ConsoleAppender
    appender.console.type = CONSOLE
    appender.console.layout.type = PatternLayout
    appender.console.layout.pattern = %d{yyyy-MM-dd HH:mm:ss,SSS} %-5p %-60c %x - %m%n
```

Deploying Flink on Kubernetes

Jobmanager_deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flink-jobmanager
spec:
  replicas: 1
  selector:
    matchLabels:
      app: flink
      component: jobmanager
  template:
    metadata:
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/port:   '9249'
      labels:
        app: flink
        component: jobmanager
    spec:
      containers:
      - name: jobmanager
        image: flink:1.11.2-scala_2.11
        command: ["/bin/bash", "-c", "/opt/flink/bin/jobmanager.sh start-foreground jobmanager"]
        ports:
        - containerPort: 6123
          name: rpc
        - containerPort: 6124
          name: blob
        - containerPort: 8081
          name: ui
        livenessProbe:
          tcpSocket:
            port: 6123
          initialDelaySeconds: 30
          periodSeconds: 60
        volumeMounts:
        - name: flink-config-volume
          mountPath: /opt/flink/conf
        securityContext:
          runAsUser: 9999
      volumes:
      - name: flink-config-volume
        configMap:
          name: flink-config
          items:
          - key: flink-conf.yaml
            path: flink-conf.yaml
          - key: log4j.properties
            path: log4j.properties
          - key: log4j-console.properties
            path: log4j-console.properties
```

Taskmanager_deployment.yaml:

```
        app: flink
        component: taskmanager
    spec:
      containers:
      - name: taskmanager
        image: flink:1.11.2-scala_2.11
        command: ["/bin/bash", "-c", "/opt/flink/bin/taskmanager.sh start-foreground -Djobmanager.rpc.address=jobmanager"]
        ports:
        - containerPort: 6122
          name: rpc
        livenessProbe:
          tcpSocket:
            port: 6122
```

Jobmanager-service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: jobmanager
spec:
  type: ClusterIP
  ports:
  - name: rpc
    port: 6123
  - name: blob
    port: 6124
  - name: ui
    port: 8081
  selector:
    app: flink
    component: jobmanager
```

Jobmanager-rest-service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: flink-jobmanager-rest
spec:
  type: NodePort
  ports:
  - name: rest
    port: 8081
    targetPort: 8081
    nodePort: 30081
  selector:
    app: flink
    component: jobmanager
```

Deploying Prometheus and Grafana on Kubernetes

Prometheus_cluster_role.yaml:

```yaml
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: prometheus
rules:
- apiGroups: [""]
  resources:
  - nodes
  - nodes/proxy
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups:
  - extensions
  resources:
  - ingresses
  verbs: ["get", "list", "watch"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects:
- kind: ServiceAccount
  name: default
  namespace: default
```

Grafana_datasource_config.yaml:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: grafana-ini
  namespace: default
data:
  grafana.ini: |
    auth:
      disable_login_form: true
    auth.anonymous:
      enabled: true
      org_role: Editor
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: grafana-datasources
  namespace: default
data:
  prometheus.yaml: |-
    {
        "apiVersion": 1,
        "datasources": [
            {
               "access":"proxy",
                "editable": true,
                "name": "prometheus",
                "orgId": 1,
                "type": "prometheus",
                "isDefault": "true",
                "url": "http://prometheus-service.default.svc:9090",
                "version": 1
            }
        ]
    }
```

Deploying Minio on Kubernetes

Minio_PV.yaml:

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: minio-pv
spec:
  storageClassName: manual
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /data
```

Minio_PVC.yaml:

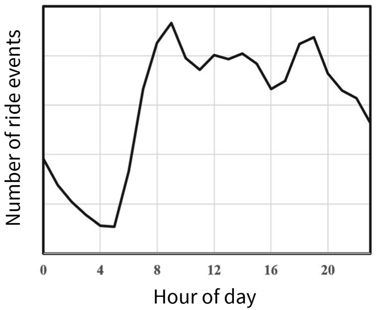
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: minio-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

Minio_deplyment.yaml:

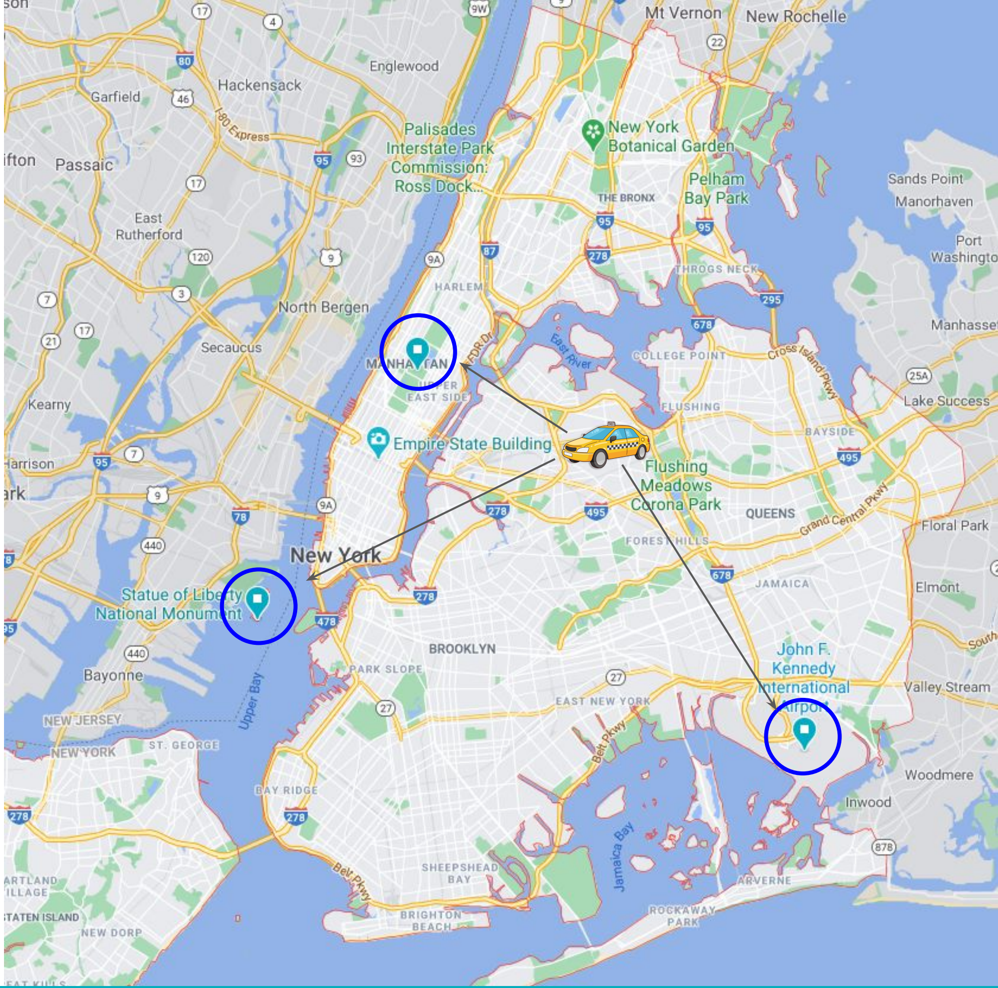
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  # This name uniquely identifies the Deployment
  name: minio
spec:
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: minio
  template:
    metadata:
      labels:
        # Label is used as selector in the service.
        app: minio
    spec:
      # Refer to the PVC created earlier
      volumes:
      - name: data
        persistentVolumeClaim:
          # Name of the PVC created earlier
          claimName: minio-pv-claim
      containers:
      - name: minio
        # Pulls the default MinIO image from Docker Hub
        image: minio/minio
        args:
        - server
        - /data
        env:
        # MinIO access key and secret key
        - name: MINIO_ACCESS_KEY
          value: "minio"
        - name: MINIO_SECRET_KEY
          value: "minio123"
        ports:
        - containerPort: 9000
        # Mount the volume into the pod
        volumeMounts:
        - name: data # must match the volume name, above
          mountPath: "/data"
```

Workload & application

- Dataset: New York taxi rides
- Workload: Stream of rides’ start events
- Application: Finding the closest famous place to the starting point of each ride.

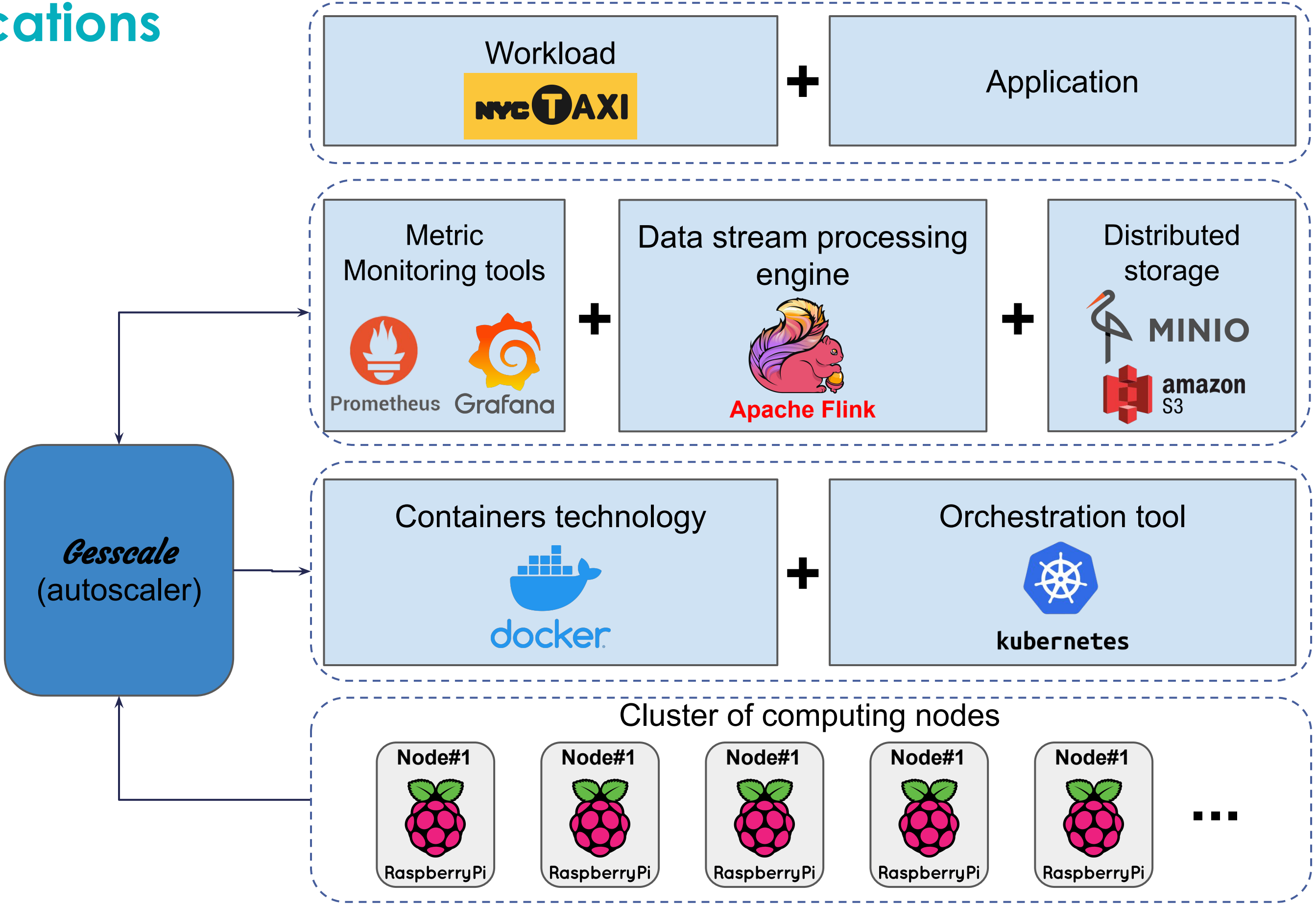


- Algorithm:
    - Get the selected fields:
        - <RideID, StartTime, Lat., Lon.>
    - Calculate the distances.
    - Compare the distances.
    - Create the output Tuple:
        - (RideID, StartTime, ClosestPlace, Distance)



Gesscale communications with the testbed

- Monitoring input rate, throughput, level of back pressure of operators
- Providing the updated list of resources to Flink
- Dynamically adding or removing replicas of operators
- Triggering Flink to make a change in its execution model



Remaining issues and challenges

- Completing and then integrating Ansible files to automate all installations and configurations
- Using MinIO for savepointing of Flink reconfiguration
- Changing network latencies based on experiments’ scenarios
- Remaining *Gesscale* communications with the testbed

Hamidreza Arkian

hamidreza.arkian@irisa.fr

Training the next generation of European Fog computing experts

http://www.fogguru.eu/

Acknowledgment

This work is part of a project that has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 765452.