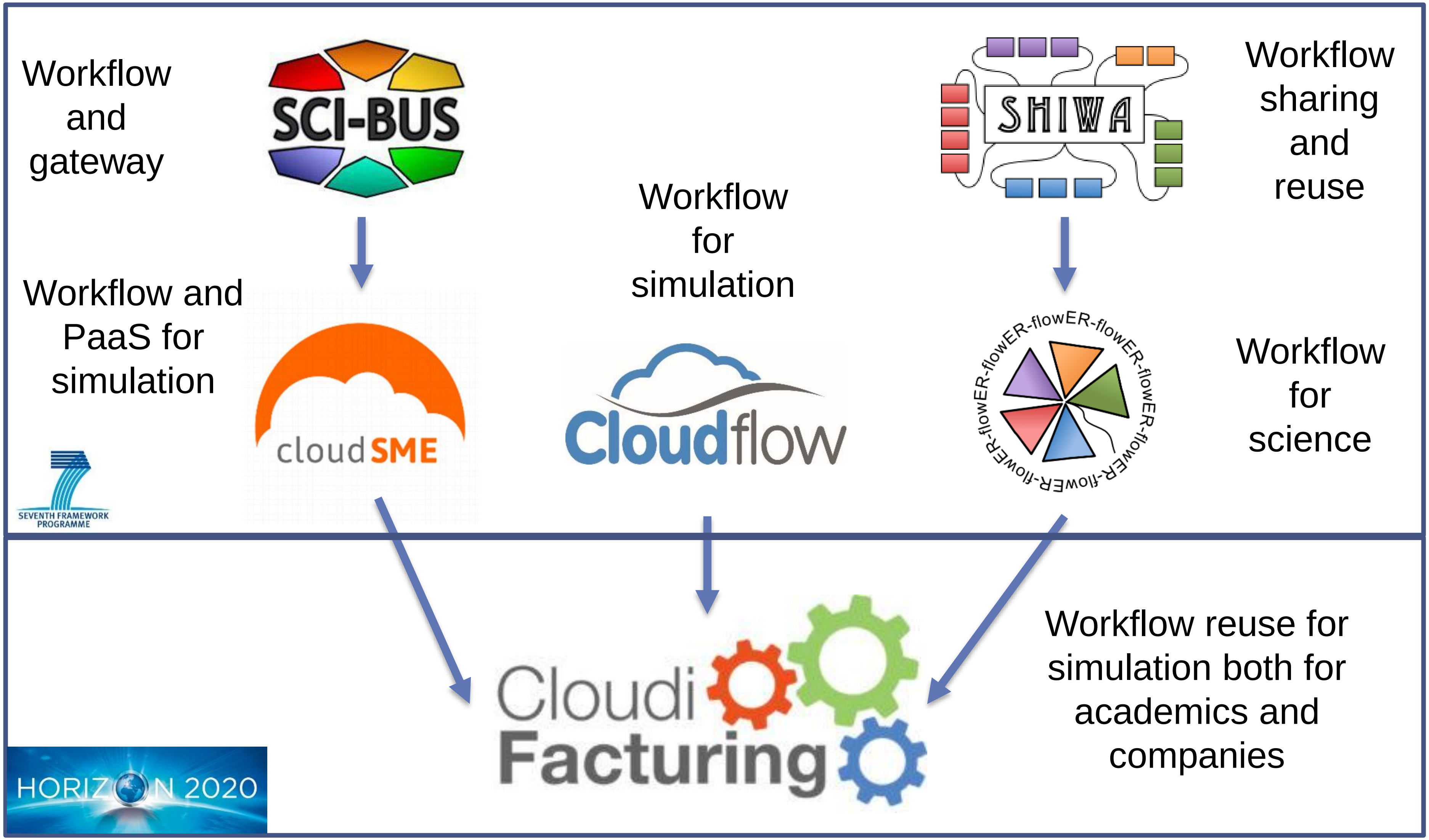
Scientific Workflows in the Era of Clouds

Péter Kacsuk
MTA SZTAKI

Agenda

- Lessons learnt from FP7 European projects using workflows
- Requirements for simulation purpose workflows in the H2020 CloudiFacturing project
- Solving the workflow sharing and reuse problems in clouds
- Infrastructure-aware workflows for clouds
- Flowbster stream-oriented workflow system for clouds
- Summary

Lessons Learnt from FP7 projects



SCI-BUS
SHIWA
cloudSME
Cloudflow
flowER-flowER-flowER-flowER-flowER-flowER-
CloudiFacturing
SEVENTH FRAMEWORK PROGRAMME
HORIZON 2020

Workflow and gateway

Workflow sharing and reuse

Workflow for simulation

Workflow and PaaS for simulation

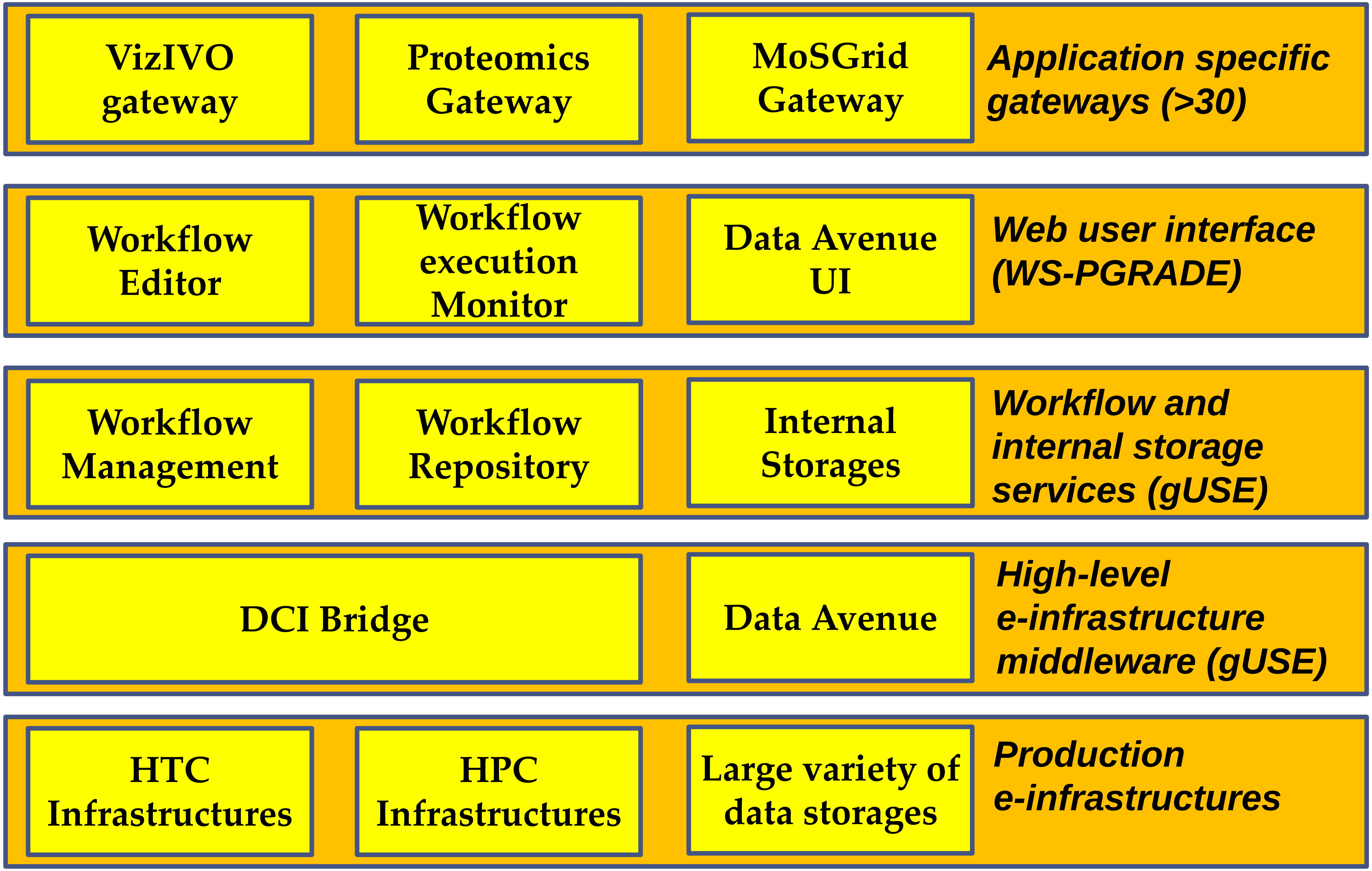
Workflow for science

Workflow reuse for simulation both for academics and companies

How Do These Projects Contribute to CloudiFacturing?

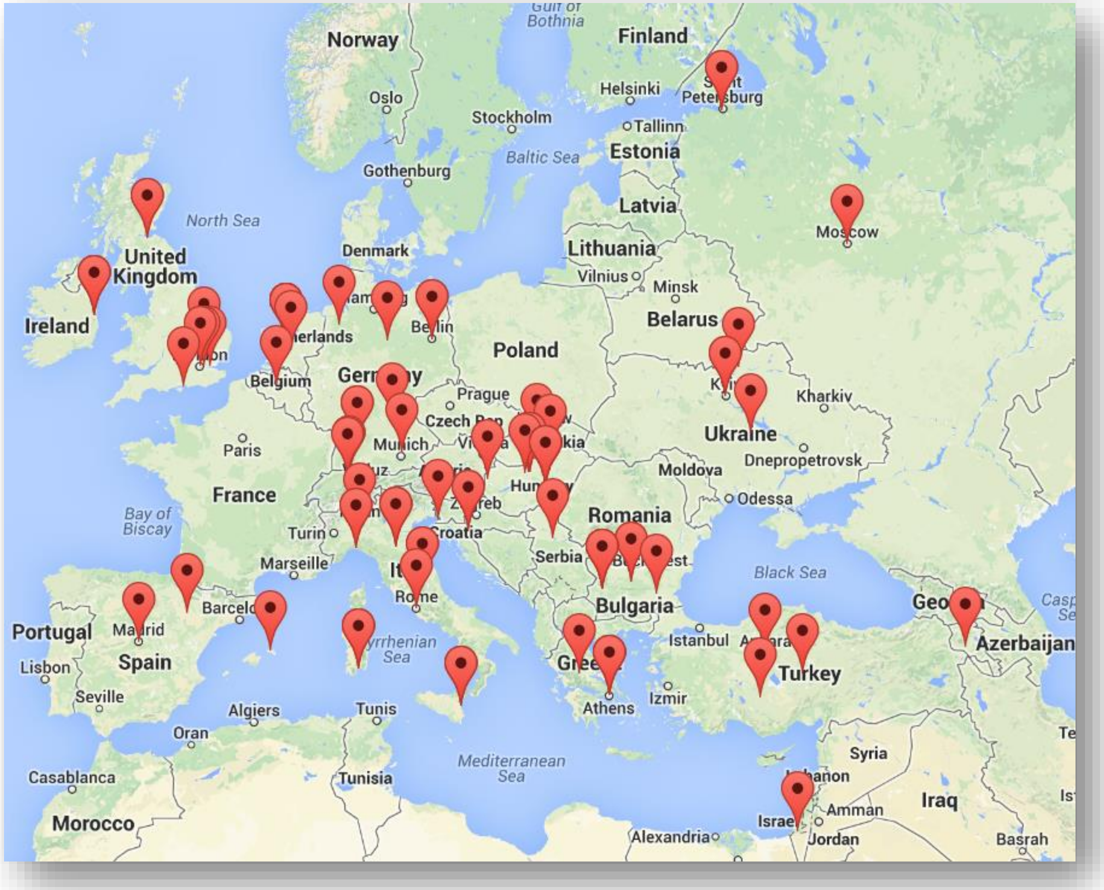
- **SCI-BUS** has developed **WS-PGRADE/gUSE workflow system** and gateway framework for scientific communities and companies
- In **CloudSME**, WS-PGRADE/gUSE was applied for simulation applications of SMEs to run in even hybrid, heterogeneous clouds by **integrating its stack with CloudBroker Platform**
- **CloudFlow** has developed the CloudFlow workflow infrastructure for companies to enable the **integration of different companies' products** into a single workflow application
- **SHIWA** has developed the **coarse-grain interoperability** solution to share and combine workflows written in different workflow systems
- **ER-Flow** enabled the shared and integrated usage of existing scientific workflows developed in different workflow systems (**put into practice the results of SHIWA**)
- **CloudiFacturing** will integrate all these results to enable the shared and integrated usage of the workflows developed in CloudSME and CloudFlow

SCI-BUS in a NutShell: WS-PGRADE/gUSE

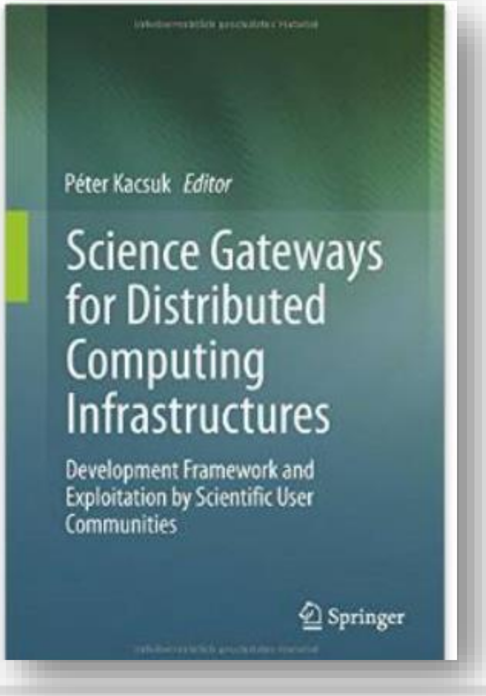


VizIVO gateway	Proteomics Gateway	MoSGrid Gateway	***Application specific gateways (>30)***
Workflow Editor	Workflow execution Monitor	Data Avenue UI	***Web user interface (WS-PGRADE)***
Workflow Management	Workflow Repository	Internal Storages	***Workflow and internal storage services (gUSE)***
DCI Bridge		Data Avenue	***High-level e-infrastructure middleware (gUSE)***
HTC Infrastructures	HPC Infrastructures	Large variety of data storages	***Production e-infrastructures***

gUSE Based Gateways



- More than 100 deployments world-wide
- More than 20.000 downloads from 75 countries on sourceforge



Péter Kacsuk Editor

Science Gateways for Distributed Computing Infrastructures

Development Framework and Exploitation by Scientific User Communities

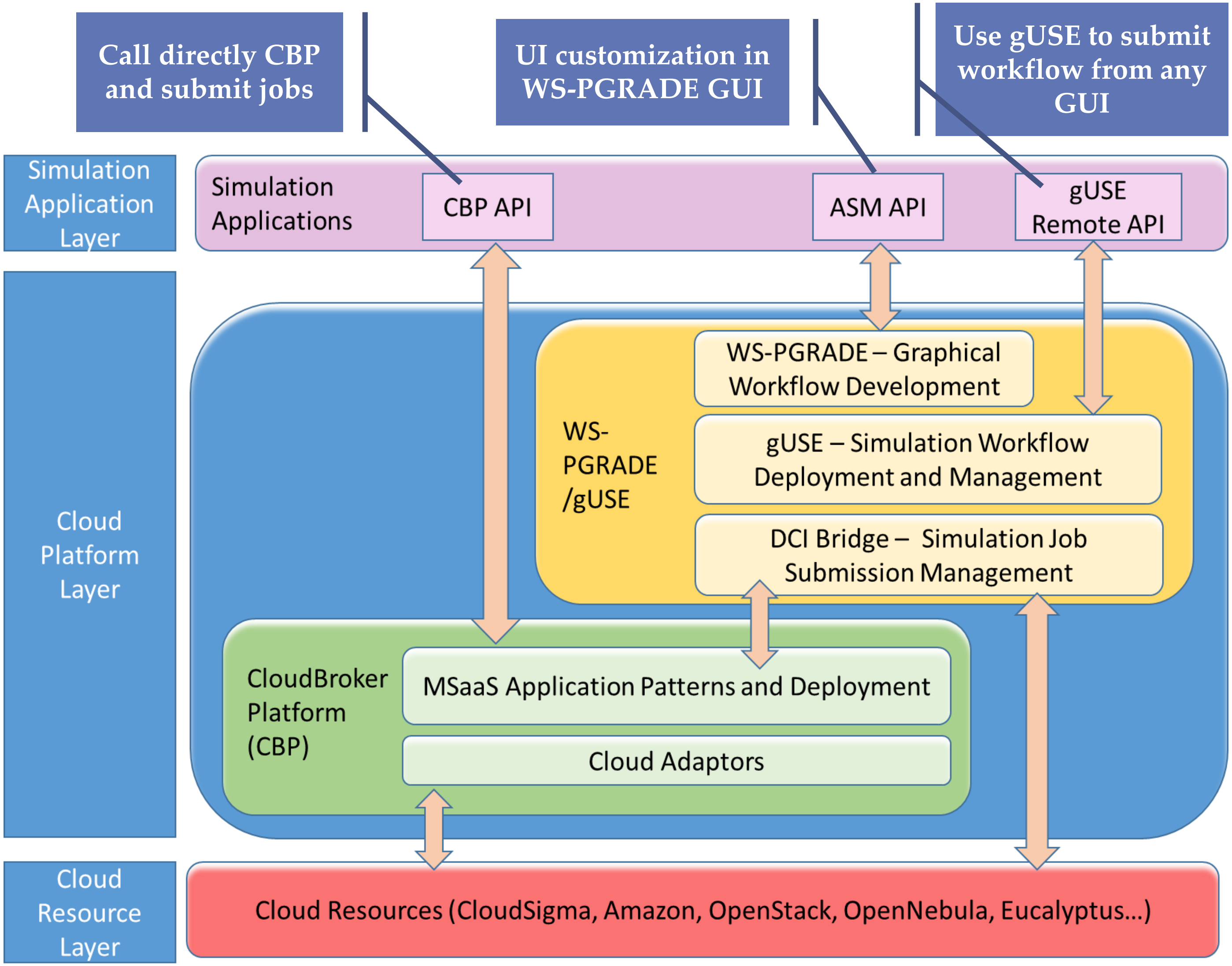
Springer

CloudSME in a NutShell

Deploy, develop and run simulations on multiple clouds

Fast application development via workflows

Application deployment


Call directly CBP and submit jobs

UI customization in WS-PGRADE GUI

Use gUSE to submit workflow from any GUI

Simulation Application Layer
Cloud Platform Layer
Cloud Resource Layer
Simulation Applications
CBP API
ASM API
gUSE Remote API
WS-PGRADE /gUSE
WS-PGRADE – Graphical Workflow Development
gUSE – Simulation Workflow Deployment and Management
DCI Bridge – Simulation Job Submission Management
CloudBroker Platform (CBP)
MSaaS Application Patterns and Deployment
Cloud Adaptors
Cloud Resources (CloudSigma, Amazon, OpenStack, OpenNebula, Eucalyptus...)

CloudFlow in a NutShell

What the user sees

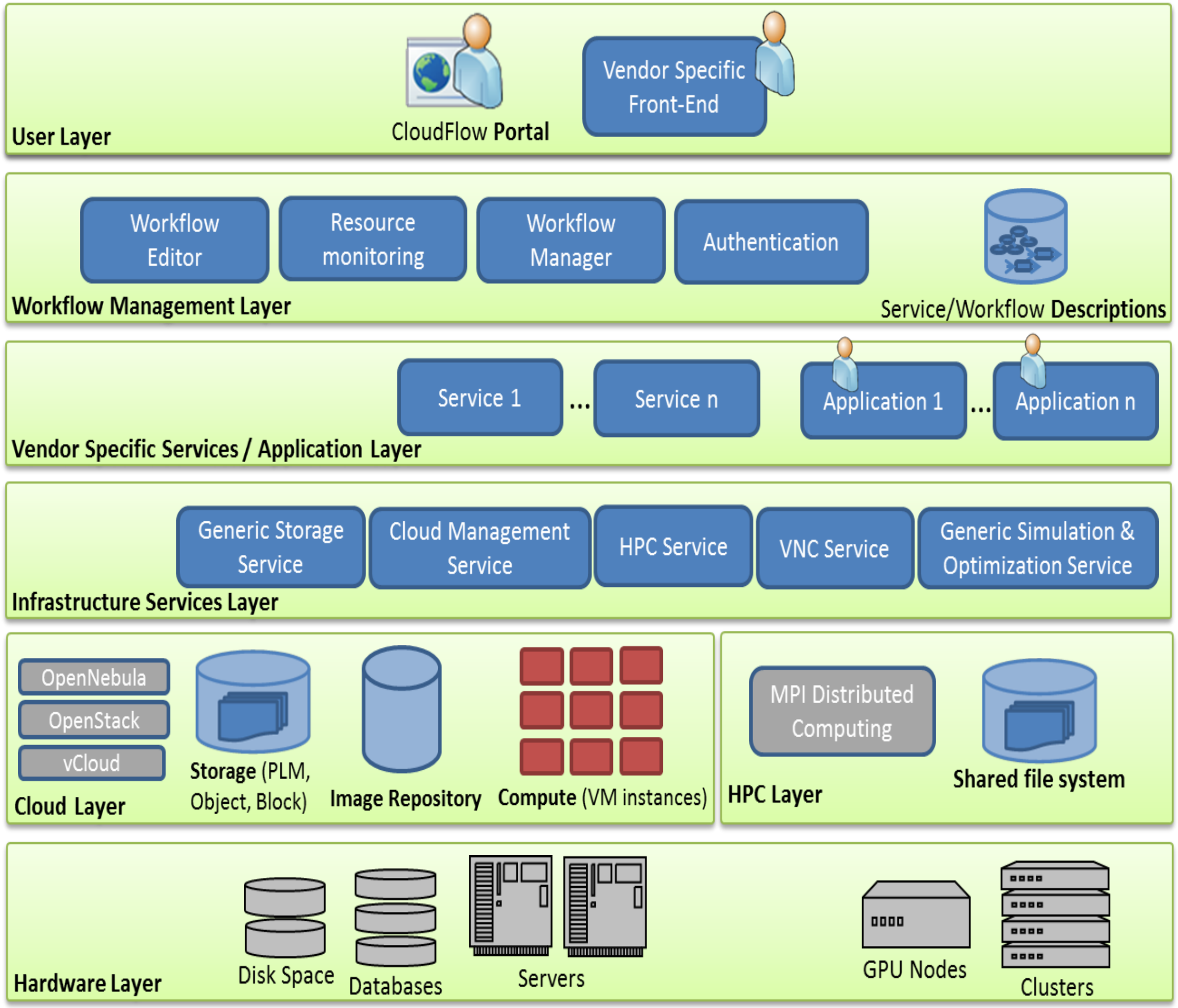
Required components

ISV contribution

Optional components

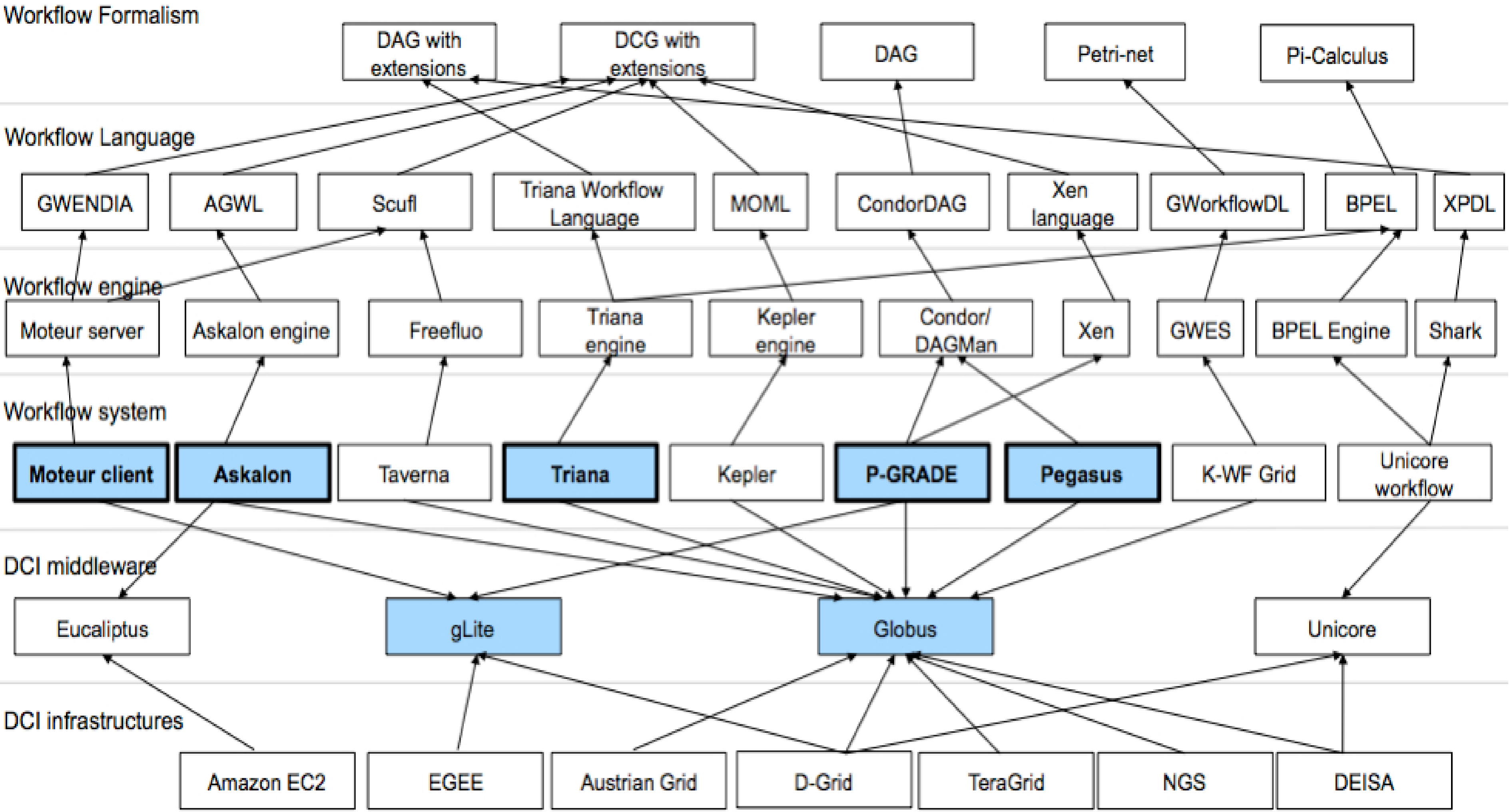
Off the shelf software

Physical machines



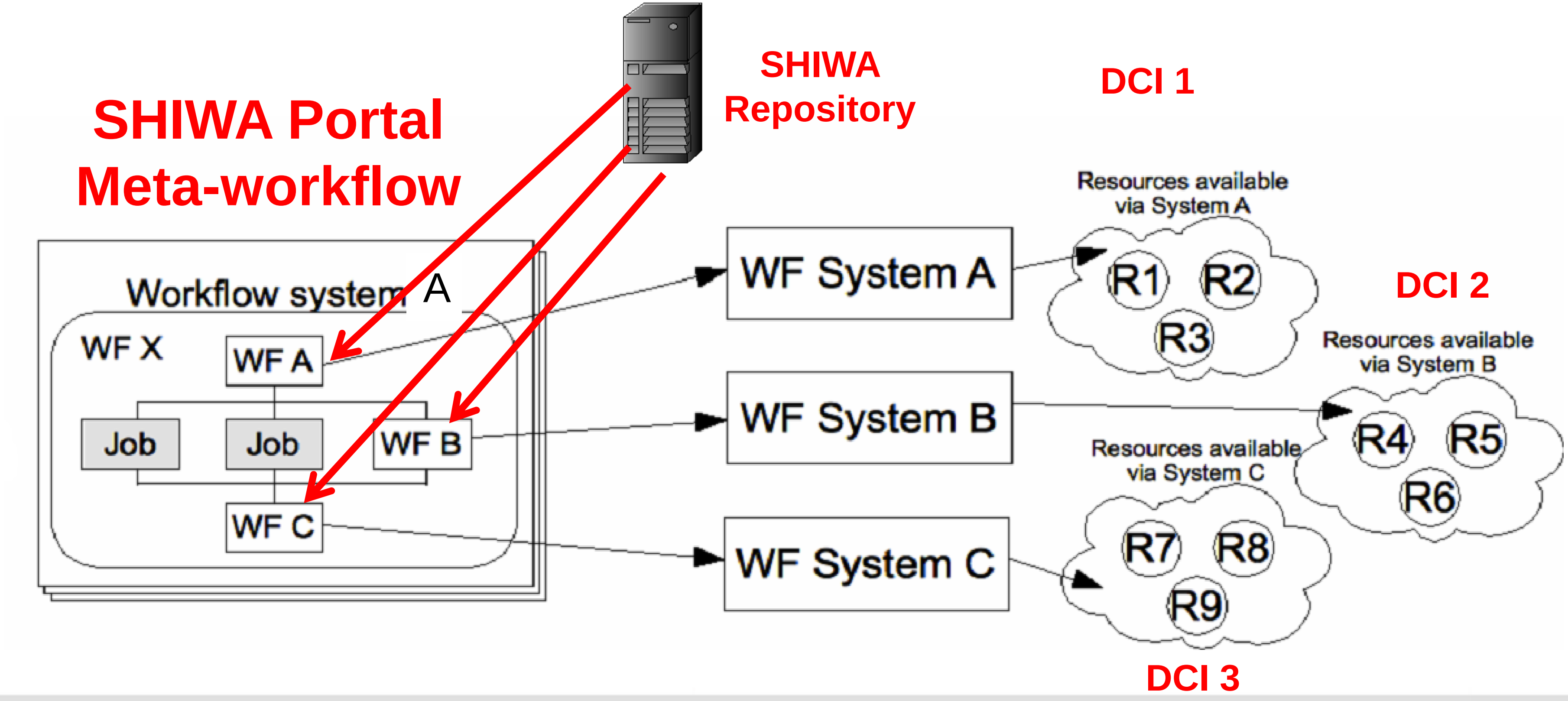
SHIWA in a NutShell

Make the WFs of the WF Ecosystem shareable and interoperable



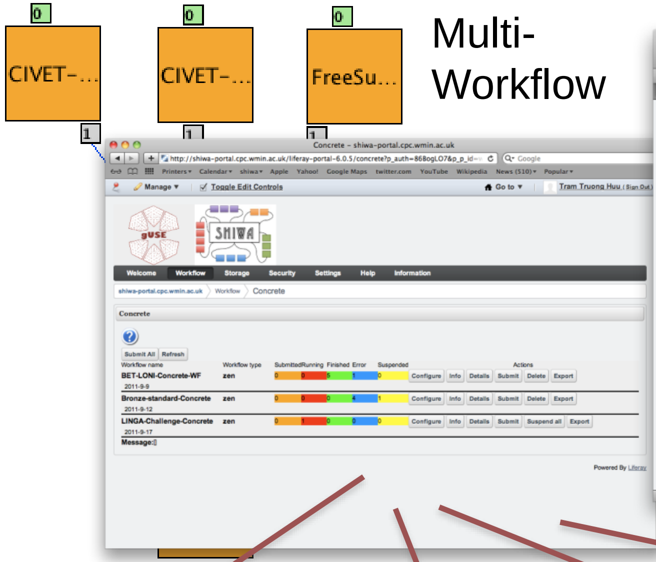
SHIWA Solution: Coarse-grained Interoperability (CGI)

- **Coarse-grained interoperability (CGI)** = Embedding of different workflows to achieve interoperability of WF execution frameworks
- If WF X running by WF system A contains a WF C that is to be executed by WF system C in DCI3 then the CGI execution mechanism takes care of executing WF C in DCI3 by WF system C



SHIWA Success Story for LINGA Brain Research

Multi-Workflow

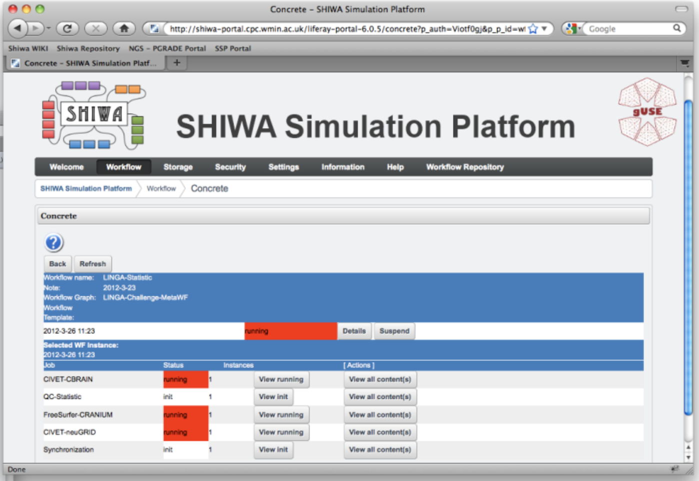


Concrete

Workflow name	Workflow type	Submitted	Running	Finished	Error	Suspended	Actions
BET-LONI-Concrete-WF 2011-9-9	zen	0	0	5	1	0	Configure Info Details Submit Delete Export
Bronze-standard-Concrete 2011-9-12	zen	0	0	0	4	1	Configure Info Details Submit Delete Export
LINGA-Challenge-Concrete 2011-9-17	zen	0	1	0	0	0	Configure Info Details Submit Suspend all Export

Message:[]

Powered By Liferay


Concrete - SHIWA Simulation Platform
http://shiwa-portal.cpc.wmin.ac.uk/liferay-portal-6.0.5/concrete?p_auth=Viotf0gj&p_p_id=wi
Shiwa WIKI
Shiwa Repository
NGS - PGRADE Portal
SSP Portal
SHIWA
gUSE
SHIWA Simulation Platform
Welcome
Workflow
Storage
Security
Settings
Information
Help
Workflow Repository
SHIWA Simulation Platform
Workflow
Concrete
Back
Refresh
Done

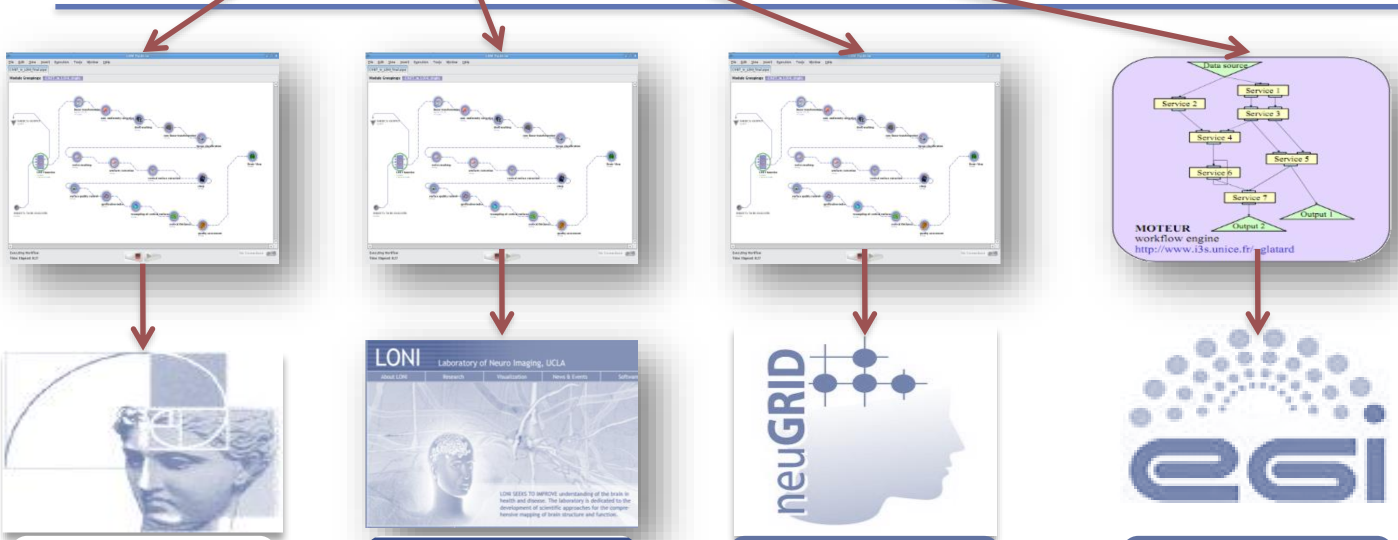
Concrete

Workflow name: LINGA-Statistic
Note: 2012-3-23
Workflow Graph: LINGA-Challenge-MetaWF
Workflow Template:

2012-3-26 11:23 running Details Suspend

Selected WF Instance:
2012-3-26 11:23

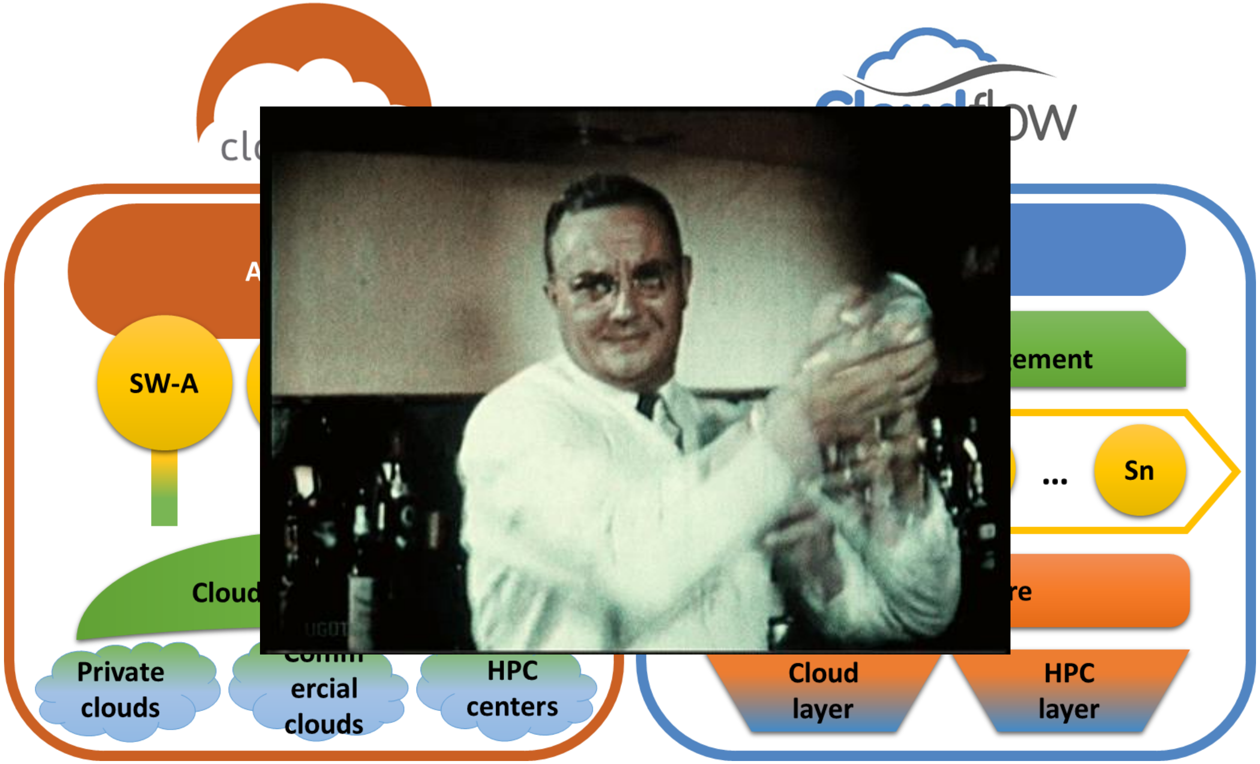
Job	Status	Instances		[Actions]
CIVET-CBRAIN	running	1	View running	View all content(s)
QC-Statistic	init	1	View init	View all content(s)
FreeSurfer-CRANIUM	running	1	View running	View all content(s)
CIVET-neuGRID	running	1	View running	View all content(s)
Synchronization	init	1	View init	View all content(s)



ER-Flow in a NutShell

- Integrating various types of workflows for various types of user communities
  - Astrophysics
    - **Taverna + gUSE** workflows running on Italian NGI resources (collaboration between Canadian, French, Italian and Spannish teams)
  - Computational Chemistry
    - **Galaxy + gUSE + UNICORE** workflows running on GERMAN NGI resources (collaboration between several German and US teams)
  - Heliophysics
    - **Taverna + gUSE** workflows running on SHIWA EGI resources (collaboration between French, English and Irish teams)
  - Life Science
    - **MOTEUR + gUSE** workflows running on EGI NGI resources (Collaboration between Dutch and German teams)

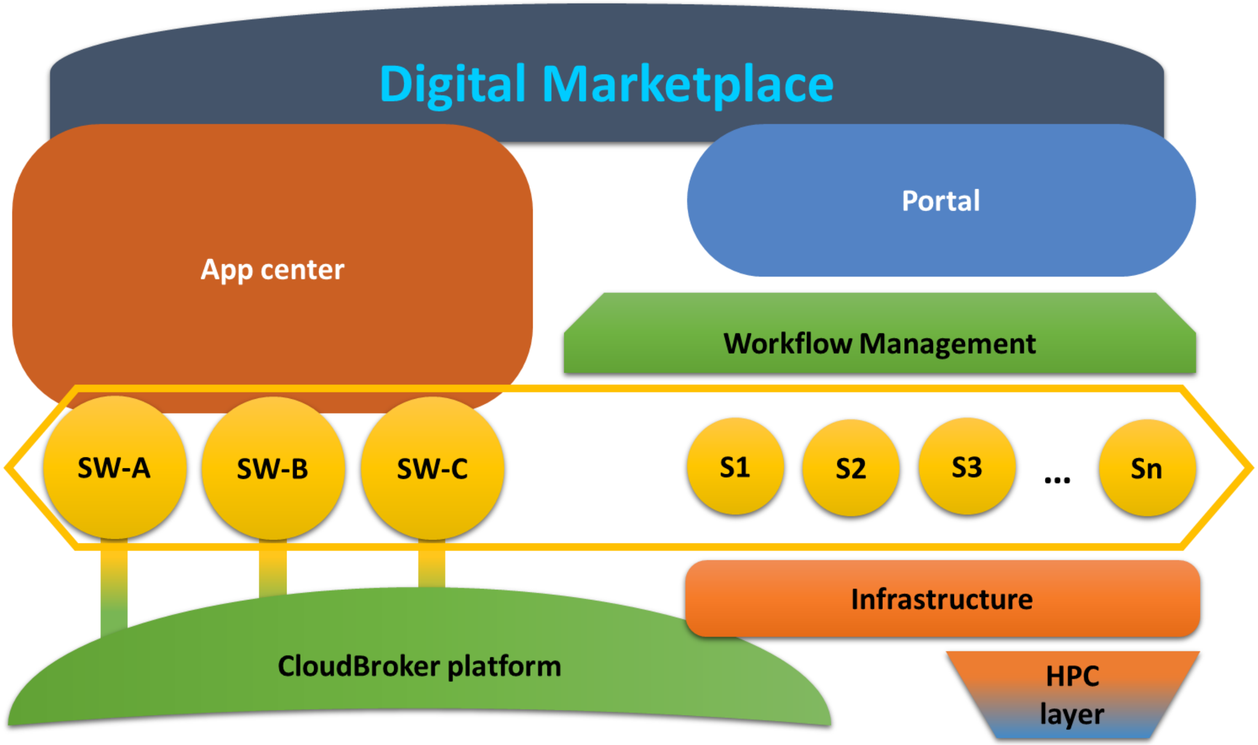
Objectives of CloudiFacturing



Target of CloudiFacturing

Companies should be able

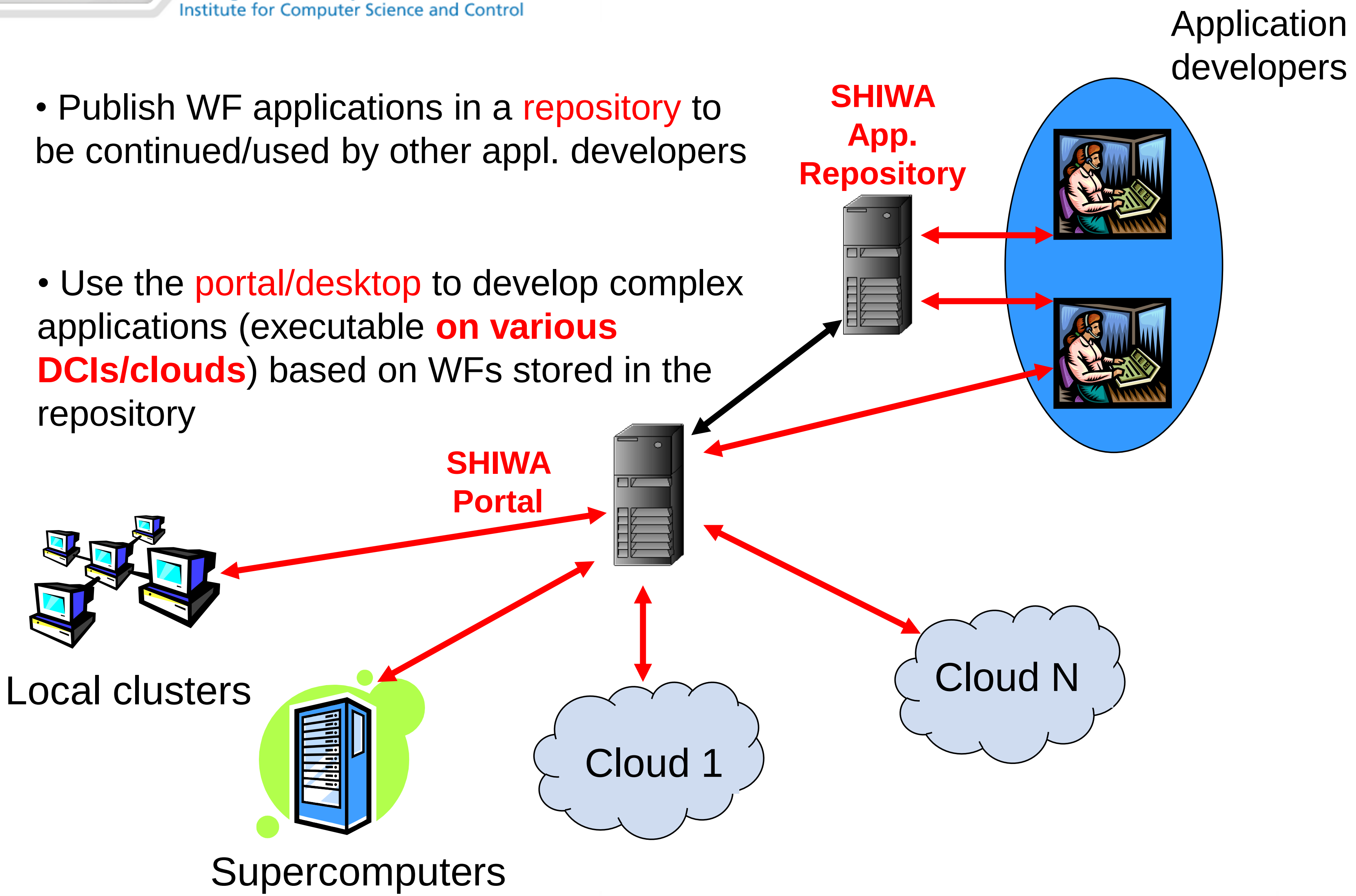
- To publish ready-to-use workflow applications
- Execute the published workflows in various clouds



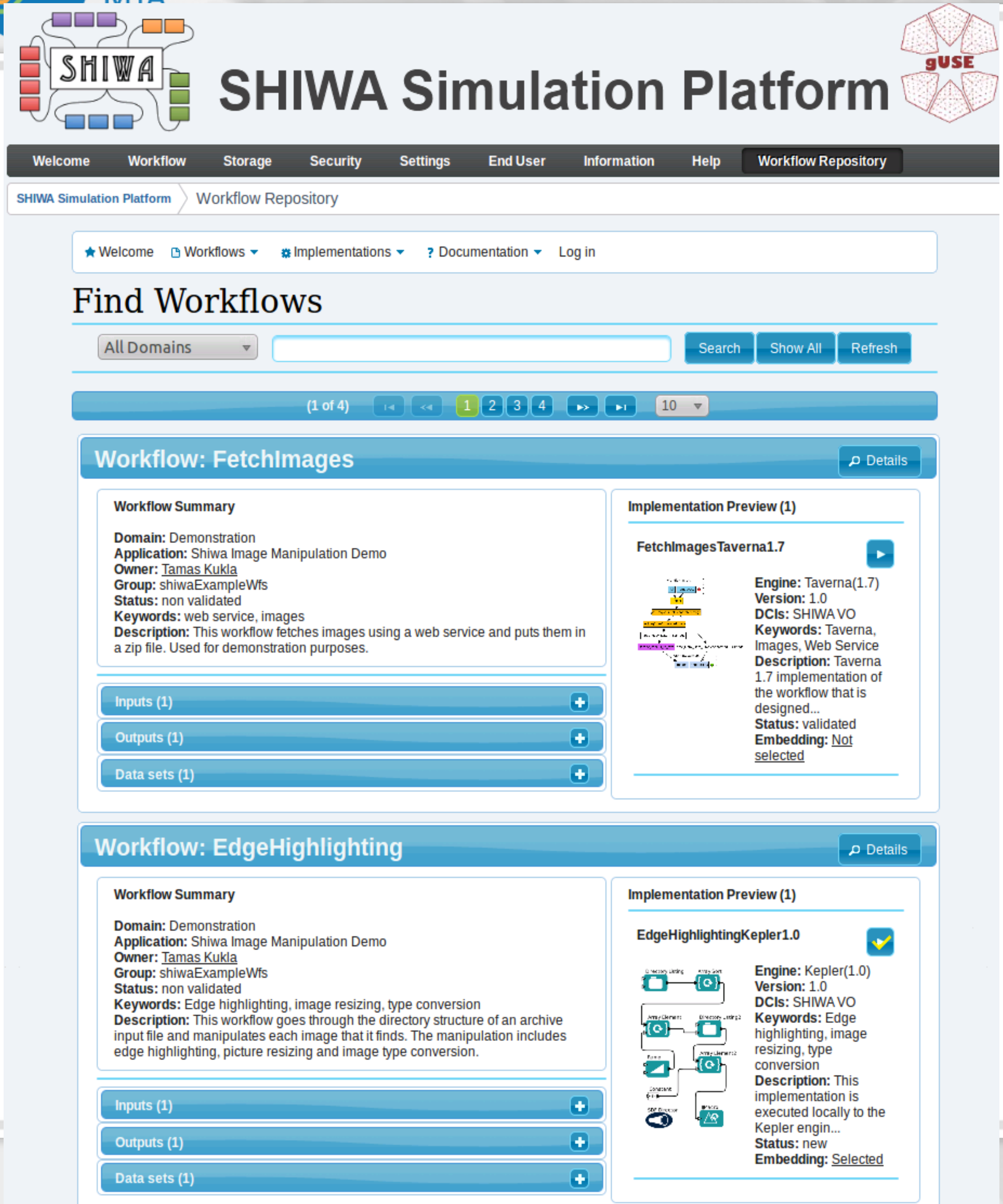
Reference Production Infrastructure of SHIWA

- Publish WF applications in a repository to be continued/used by other appl. developers

- Use the portal/desktop to develop complex applications (executable **on various DCIs/clouds**) based on WFs stored in the repository



SHIWA Repository



SHIWA Simulation Platform

Welcome | Workflow | Storage | Security | Settings | End User | Information | Help | Workflow Repository

SHIWA Simulation Platform > Workflow Repository

★ Welcome | Workflows ▾ | Implementations ▾ | ? Documentation ▾ | Log in

Find Workflows

All Domains ▾ | Search | Show All | Refresh

(1 of 4) 1 2 3 4 | 10 ▾

Workflow: FetchImages | Details

Workflow Summary

Domain: Demonstration
Application: Shiwa Image Manipulation Demo
Owner: Tamas Kukla
Group: shiwaExampleWfs
Status: non validated
Keywords: web service, images
Description: This workflow fetches images using a web service and puts them in a zip file. Used for demonstration purposes.

Inputs (1) +
Outputs (1) +
Data sets (1) +

Implementation Preview (1)

FetchImagesTaverna1.7

Engine: Taverna(1.7)
Version: 1.0
DCIs: SHIWA VO
Keywords: Taverna, Images, Web Service
Description: Taverna 1.7 implementation of the workflow that is designed...
Status: validated
Embedding: Not selected

Workflow: EdgeHighlighting | Details

Workflow Summary

Domain: Demonstration
Application: Shiwa Image Manipulation Demo
Owner: Tamas Kukla
Group: shiwaExampleWfs
Status: non validated
Keywords: Edge highlighting, image resizing, type conversion
Description: This workflow goes through the directory structure of an archive input file and manipulates each image that it finds. The manipulation includes edge highlighting, picture resizing and image type conversion.

Inputs (1) +
Outputs (1) +
Data sets (1) +

Implementation Preview (1)

EdgeHighlightingKepler1.0

Engine: Kepler(1.0)
Version: 1.0
DCIs: SHIWA VO
Keywords: Edge highlighting, image resizing, type conversion
Description: This implementation is executed locally to the Kepler engin...
Status: new
Embedding: Selected

Facilitates **publishing** and **sharing** workflows

Supports:

- Abstract workflows with multiple implementations of 10 workflow systems (ASCALON, gUSE, Moteur, Taverna, etc.)
- Storing execution specific data

Lessons Learnt in ER-Flow

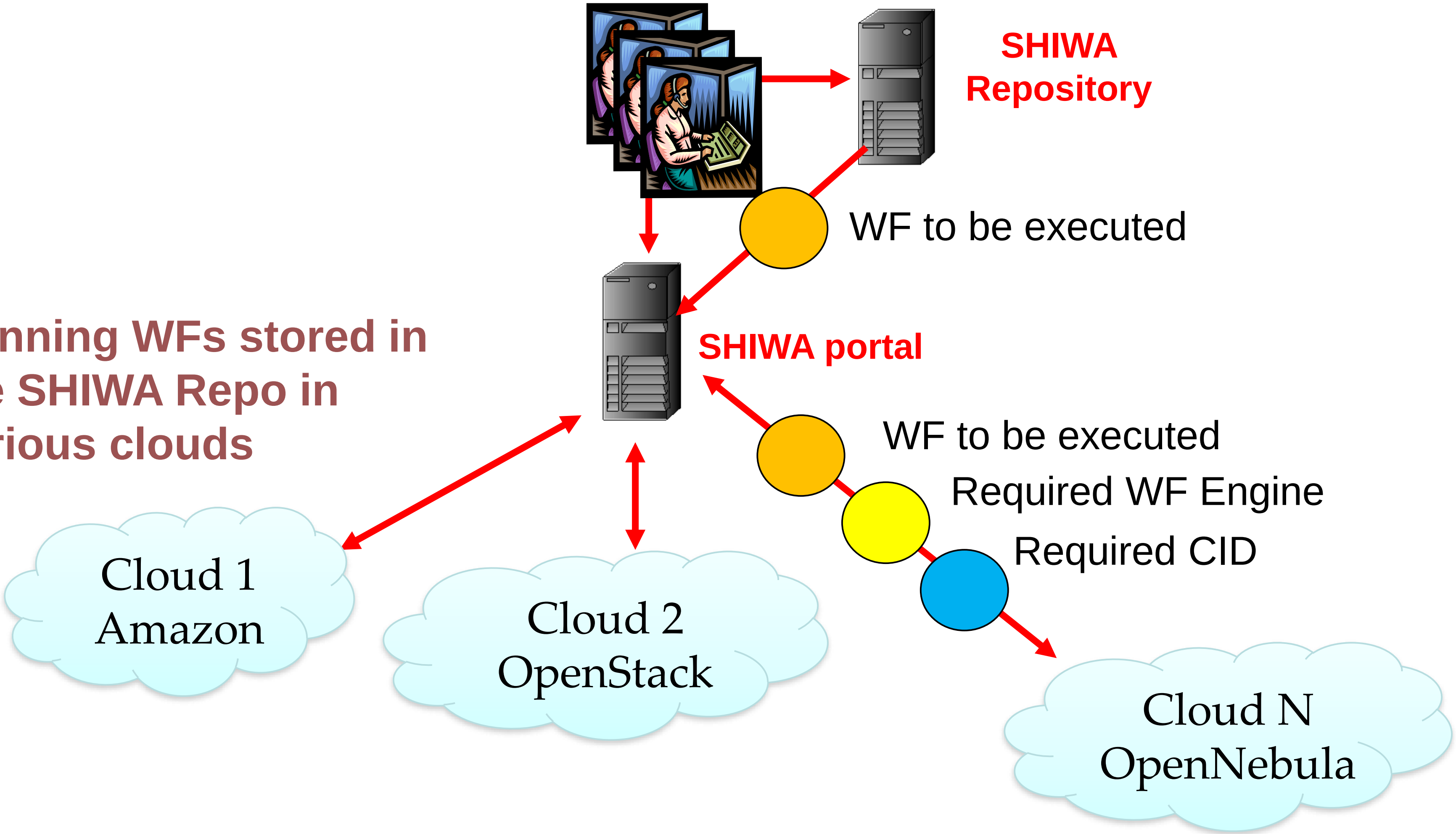
- The SHIWA CGI technology was very useful to integrate various types of workflow
- However, we have discovered a major problem: The infrastructure where the embedded WF is supposed to run can be
  - Inaccessible to the current user
  - faulty
  - removed

Solution in the Cloud

- Extend SHIWA Repo: three kinds of entities should be stored
  - WFs as before
  - WF engines as before
  - Cloud Infrastructure Descriptors (CID)
- Before executing the WF the SHIWA portal should call a cloud orchestrator to deploy the required infrastructure in the cloud
- There are many cloud orchestrators that help to deploy infrastructures in the cloud based on descriptors
- One of them is Occopus (developed in SZTAKI) and this is used to extend the SHIWA portal for deploying the required infrastructure

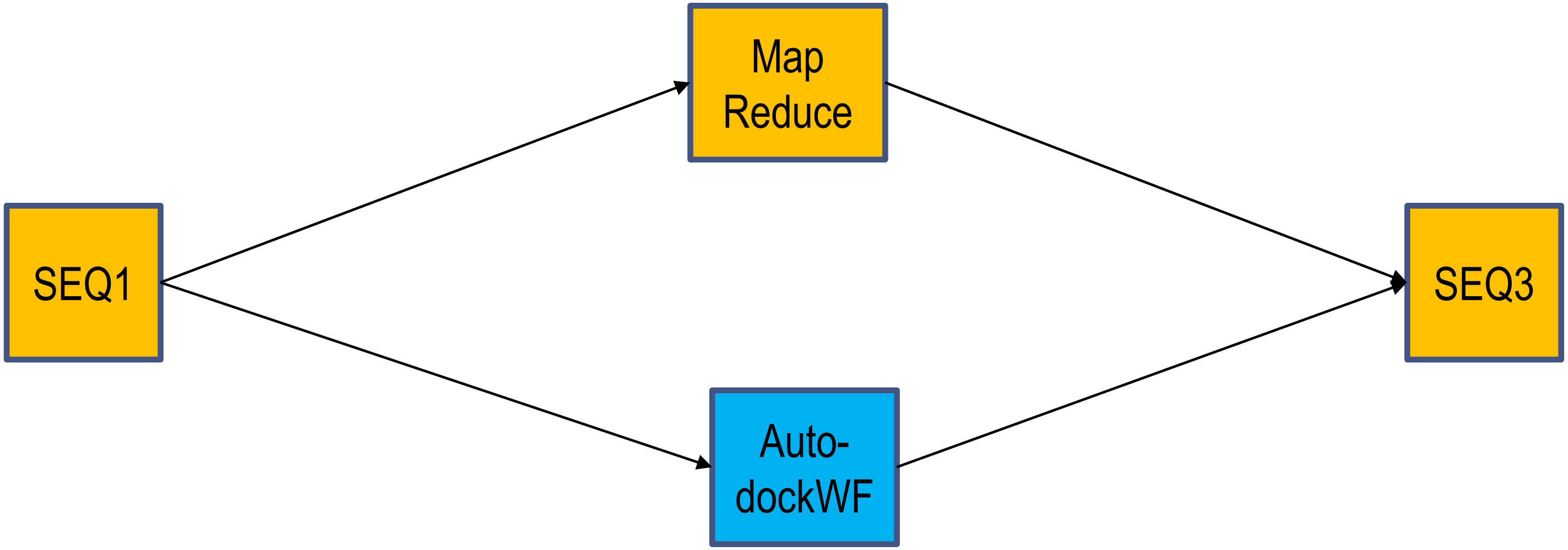
SHIWA CGI Solution with Clouds

Running WFs stored in the SHIWA Repo in various clouds

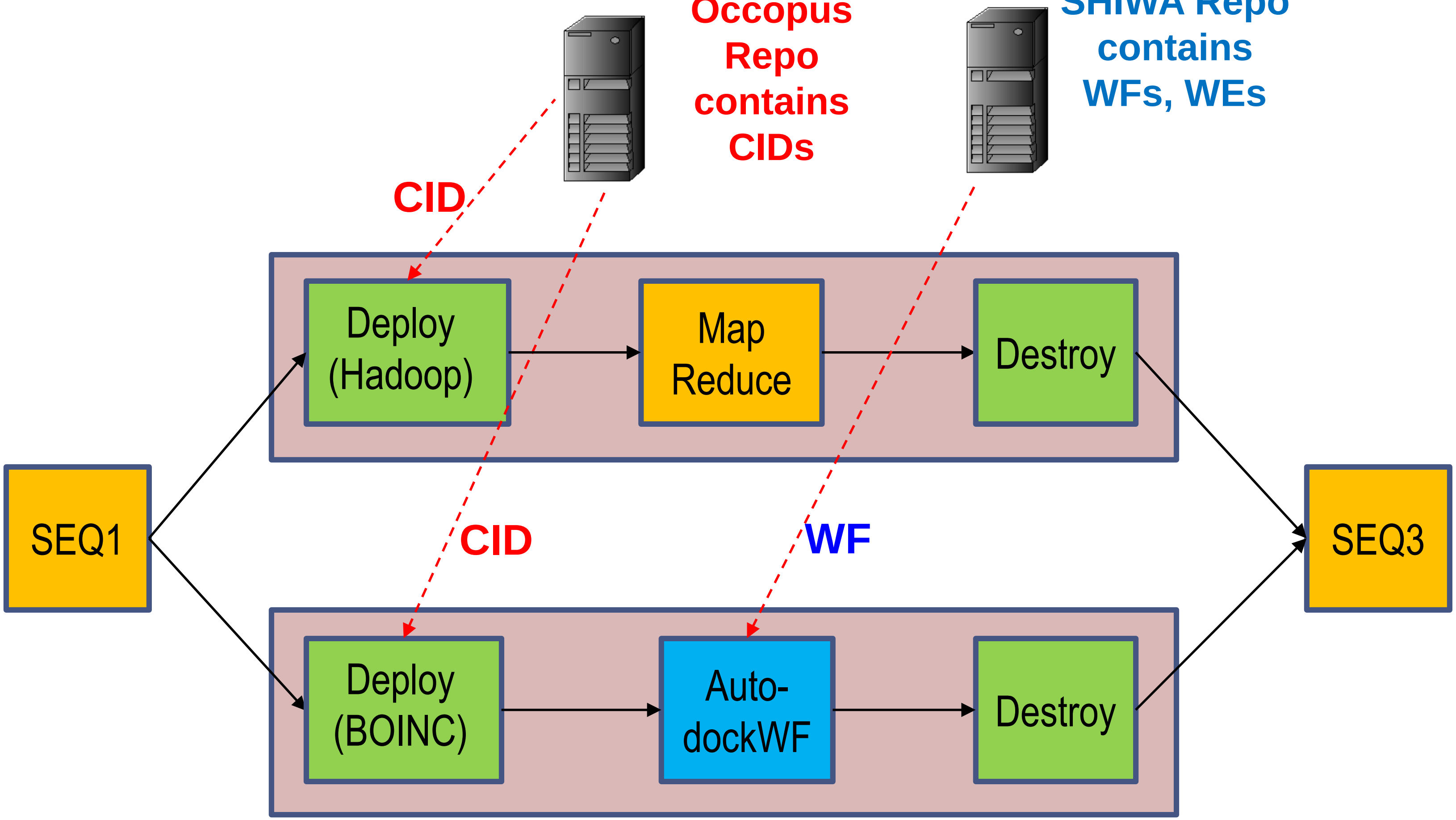


Infrastructure-aware Workflows

The Problem

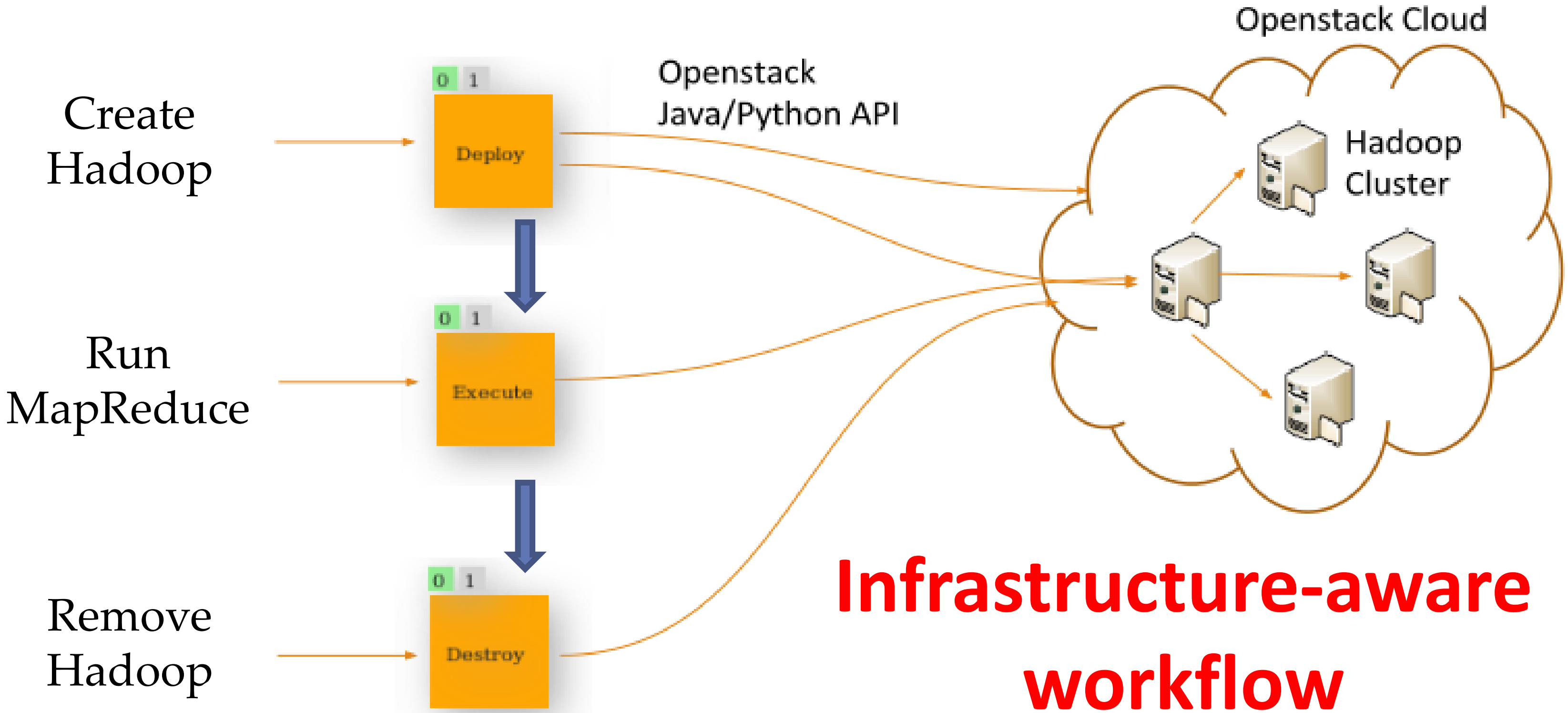


Infrastructure-aware Workflow



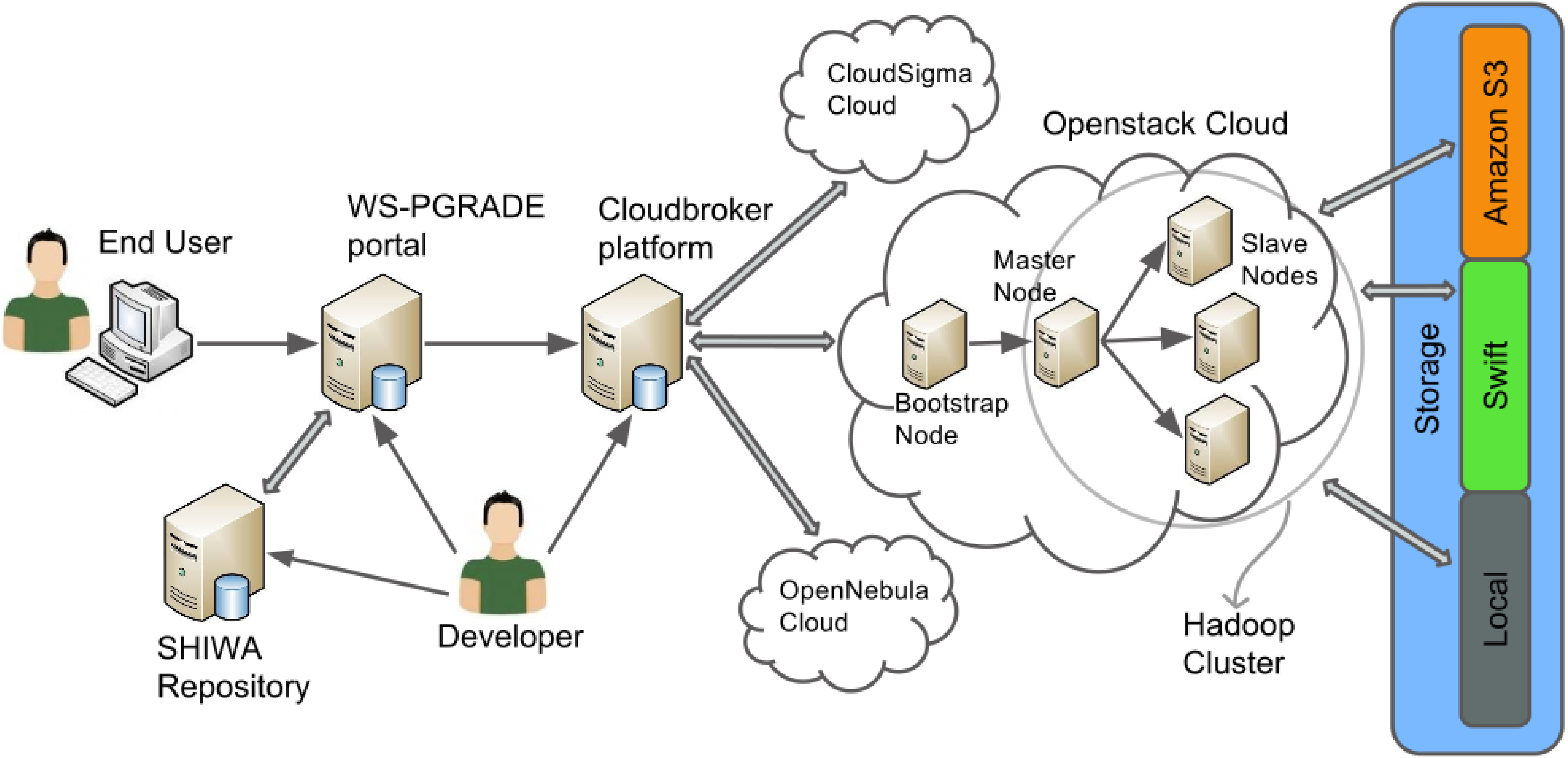
Hadoop/MapReduce Workflow Pattern

- Stage 1 or Deploy Hadoop Node: Launch servers in a cloud, connect to master node and setup Hadoop cluster
- Stage 2 or Execute Node: Upload input files and job executable to master node, execute job and get result back
- Stage 3 or Destroy Hadoop Node: Destroy cluster to free up resources

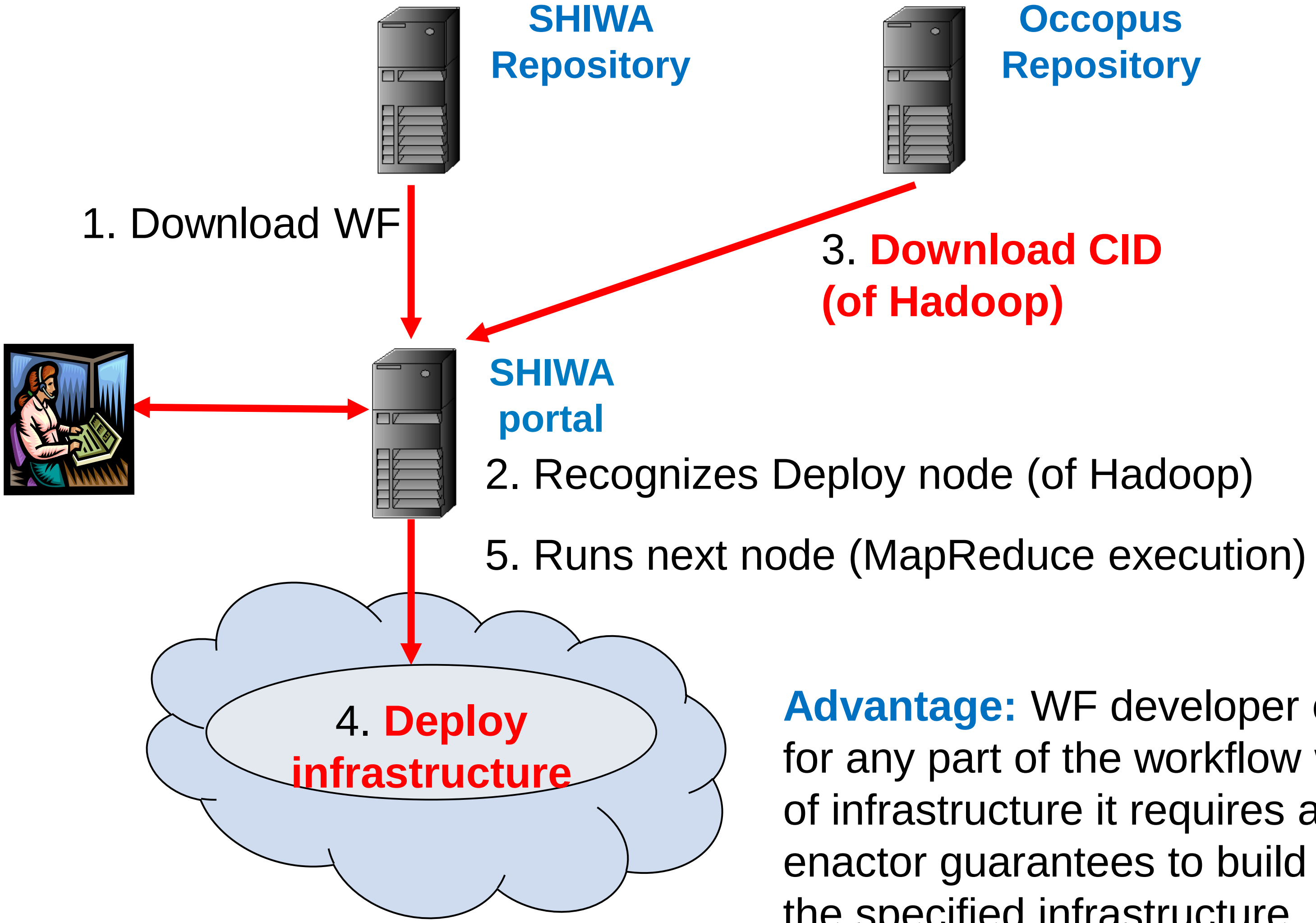


Infrastructure-aware workflow

Implementation in CloudSME



Generic Infrastructure-aware Workflow Execution

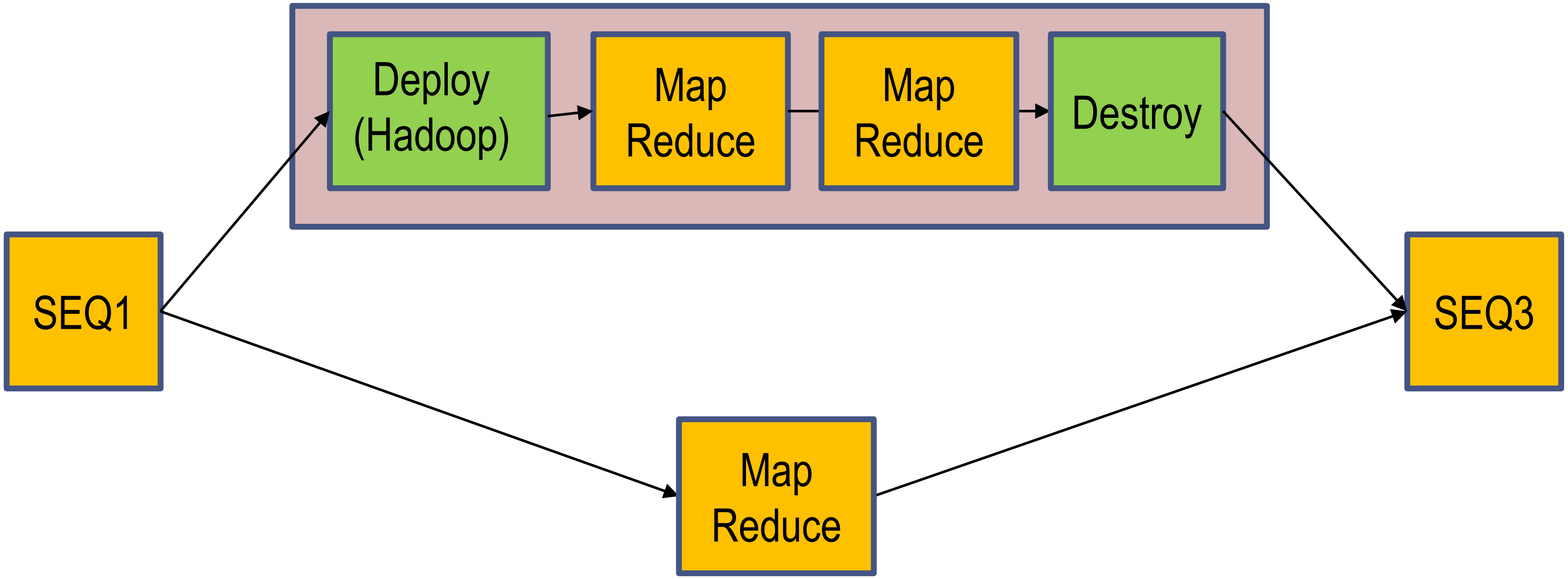

SHIWA Repository
Occopus Repository
1. Download WF
3. Download CID (of Hadoop)
SHIWA portal
2. Recognizes Deploy node (of Hadoop)
5. Runs next node (MapReduce execution)
4. Deploy infrastructure

Advantage: WF developer can specify for any part of the workflow what kind of infrastructure it requires and the WF enactor guarantees to build and use the specified infrastructure

Infrastructure-aware Workflow Issues

- How many nodes can use the infrastructure of the same deploy node?
- Can parallel branch node use the infrastructure of the same deploy node?

Structured versus unstructured concept



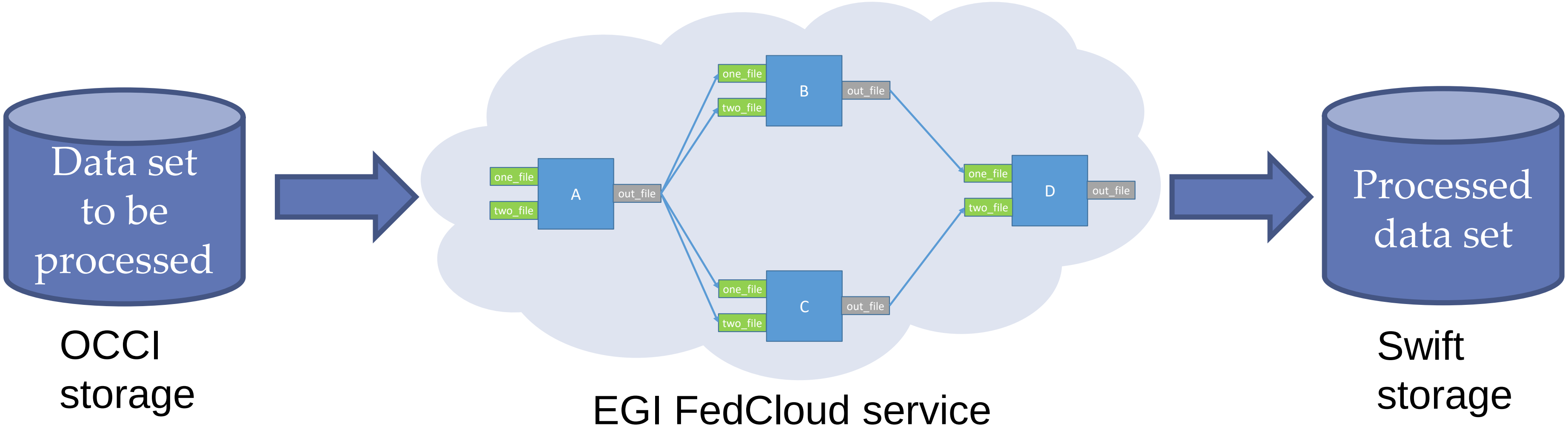
Flowbster

Lessons Learnt in CloudSME and ER-Flow

- There are two options to process large data sets:
  - 1. **WS-PGRDADE/gUSE and CloudFlow**: A whole data set (e.g. file) is given as input for the workflow. The next data set can be sent as input for the workflow only when the processing of the previous data set is completely finished by the workflow (job-oriented workflow management)
  - 2. Divide the data set into many small items and these items as stream should flow through the workflow. Nodes of the workflow work in parallel on different data element (stream-oriented or pipeline workflow management) -> **Flowbster**

Concept of Flowbster

- The goal of Flowbster is to enable
  - The quick deployment of the workflow as a pipeline infrastructure in the cloud
  - Once the pipeline infrastructure is created in the cloud it is activated and data elements of the data set to be processed flow through the pipeline
  - As the data set flows through the pipeline its data elements are processed as defined by the Flowbster workflow



Data set to be processed

OCCI storage

EGI FedCloud service

Processed data set

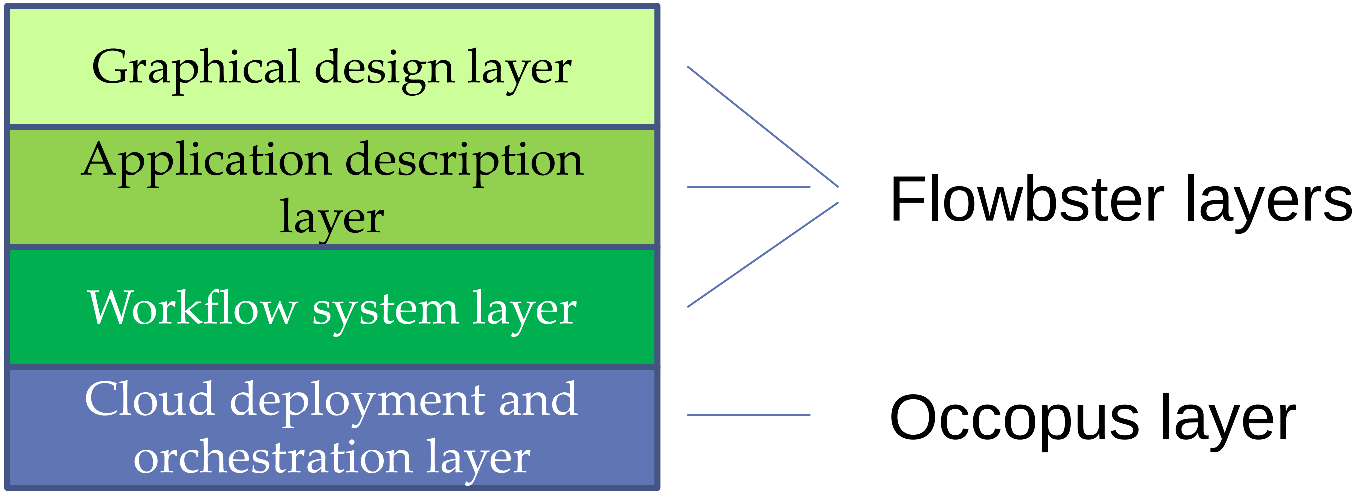
Swift storage

Flowbster Contra Job-oriented Workflow Systems

- Nodes of the Flowbster workflow directly communicate the data among them
- Data is passed through the workflow as a **data stream**
- A node is activated and executes the assigned task when all the input data arrived
- Nodes of Flowbster workflows are deployed in the cloud as VMs (or docker containers) and they exist until all the input data sets are processed
- As a result a Flowbster workflow works as a temporary **virtual infrastructure deployed in the cloud**
- Input data sets flow through this virtual infrastructure and meanwhile they flow through they are processed by the nodes of the workflow

Structure of the Flowbster Workflow System

- Goal:
  - To create the Flowbster workflow in the cloud without any cloud knowledge
- Solution:
  - To provide a layered concept where users with different expertise can enter to the use of Flowbster
- 4 layers:

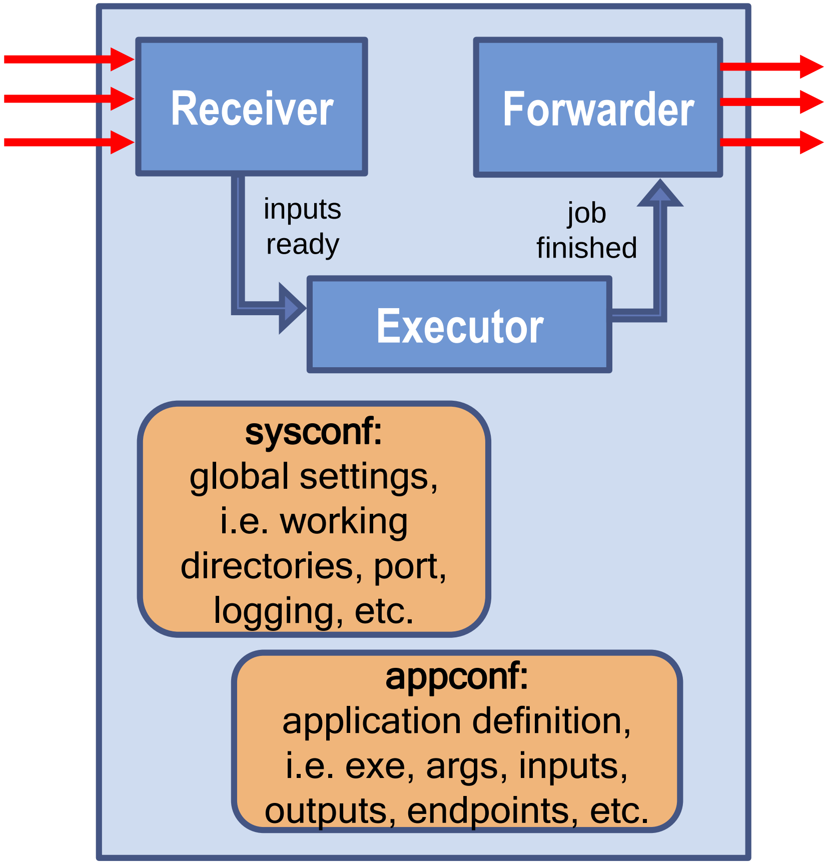


Occopus Layer

- Occopus is a cloud orchestrator and manager tool
- It automatically deploys virtual infrastructures (like Flowbster workflows) in the cloud based on an Occopus descriptor that consists of:
  - **Virtual infrastructure description**:
    - Specifies the **nodes** (services) to be deployed and all **cloud-independent** attributes e.g. input values for a service.
    - Specifies the **dependencies** among the nodes, to decide the order of deployment
    - Specifies **scaling** related attributes like min, max number of instances
  - **Node definition:**
    - Defines **how to construct the node** on a target cloud. This contains all **cloud dependent** settings, e.g. image id, flavour, contextualization
- See detailed tutorials at the Occopus web page:
  - http://occopus.lpds.sztaki.hu/tutorials

Flowbster Workflow System Layer

- Contains uniform Flowbster workflow nodes which have the internal structure shown in the figure
- Every node provides the following actions:
  - Receives and keeps track of the input items
  - Executes the (pre-) configured application when inputs are ready
  - Identifies and forwards results of execution towards a (pre-) configured endpoint
- Contains 3 components:
  - Receiver: service to receive inputs
  - Executor: service to execute predefined app
  - Forwarder: service to send results of the finished app to a predefined remote location

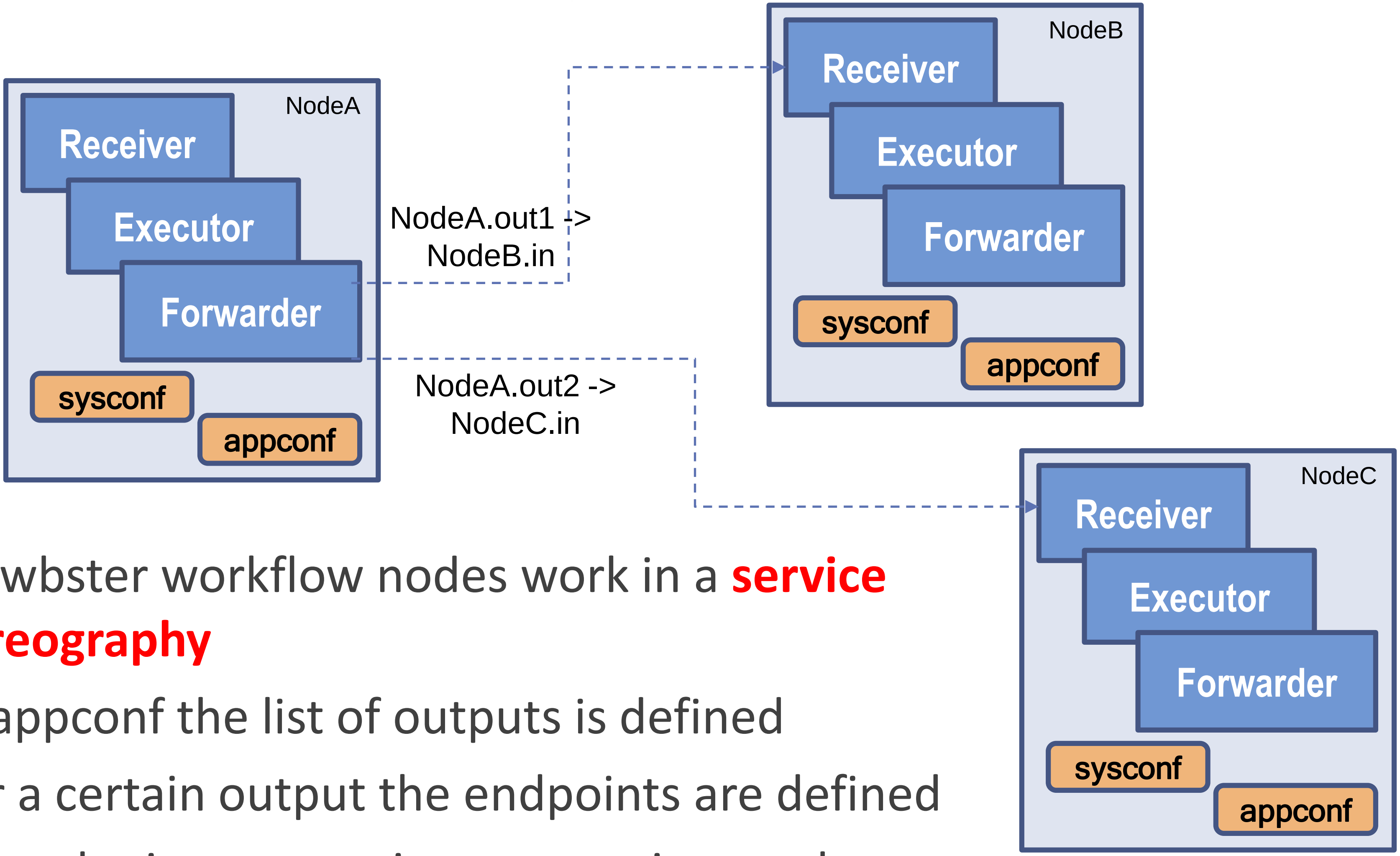


sysconf:
global settings, i.e. working directories, port, logging, etc.

appconf:
application definition, i.e. exe, args, inputs, outputs, endpoints, etc.

Also requires 2 config files in order to costumize the node according to the workflow definition

Connecting Flowbster Nodes into a Workflow



- Flowbster workflow nodes work in a **service coreography**
- In appconf the list of outputs is defined
- For a certain output the endpoints are defined
- An endpoint must point to a receiver node

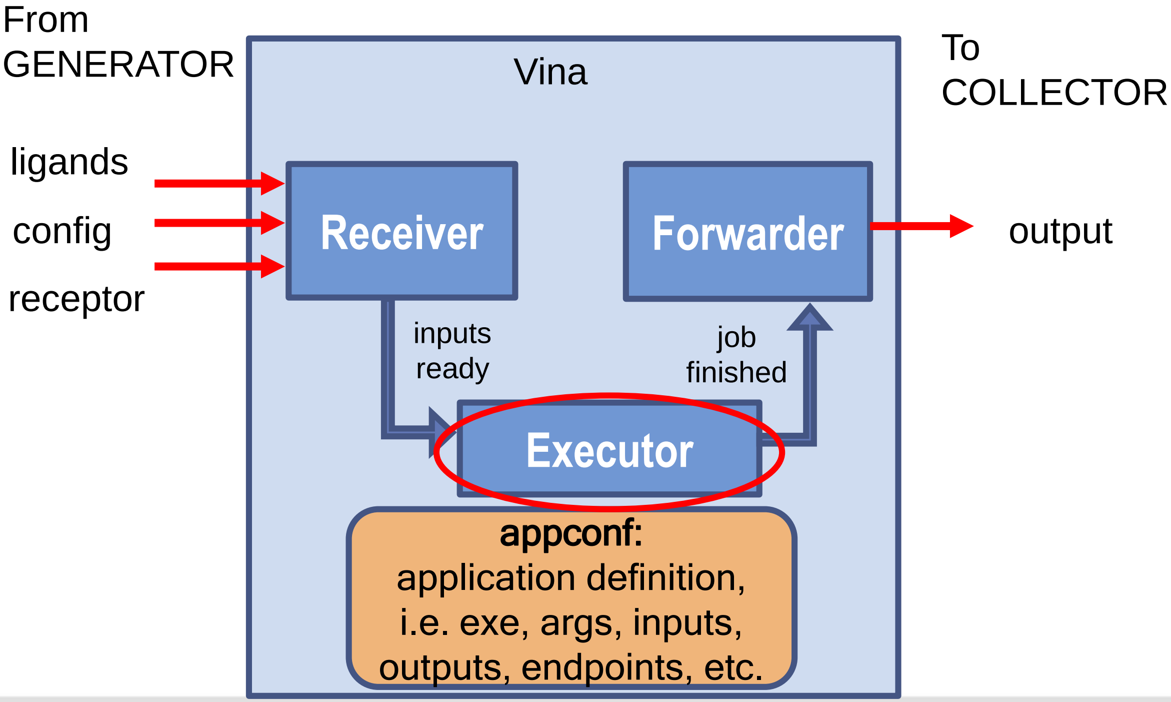
Flowbster Application Description Layer

- It contains the Occopus descriptor of the Flowbster workflow
  - Virtual infrastructure descriptor representing the workflow graph
  - Customized node definitions for each node of the workflow. E.g. Vina node:

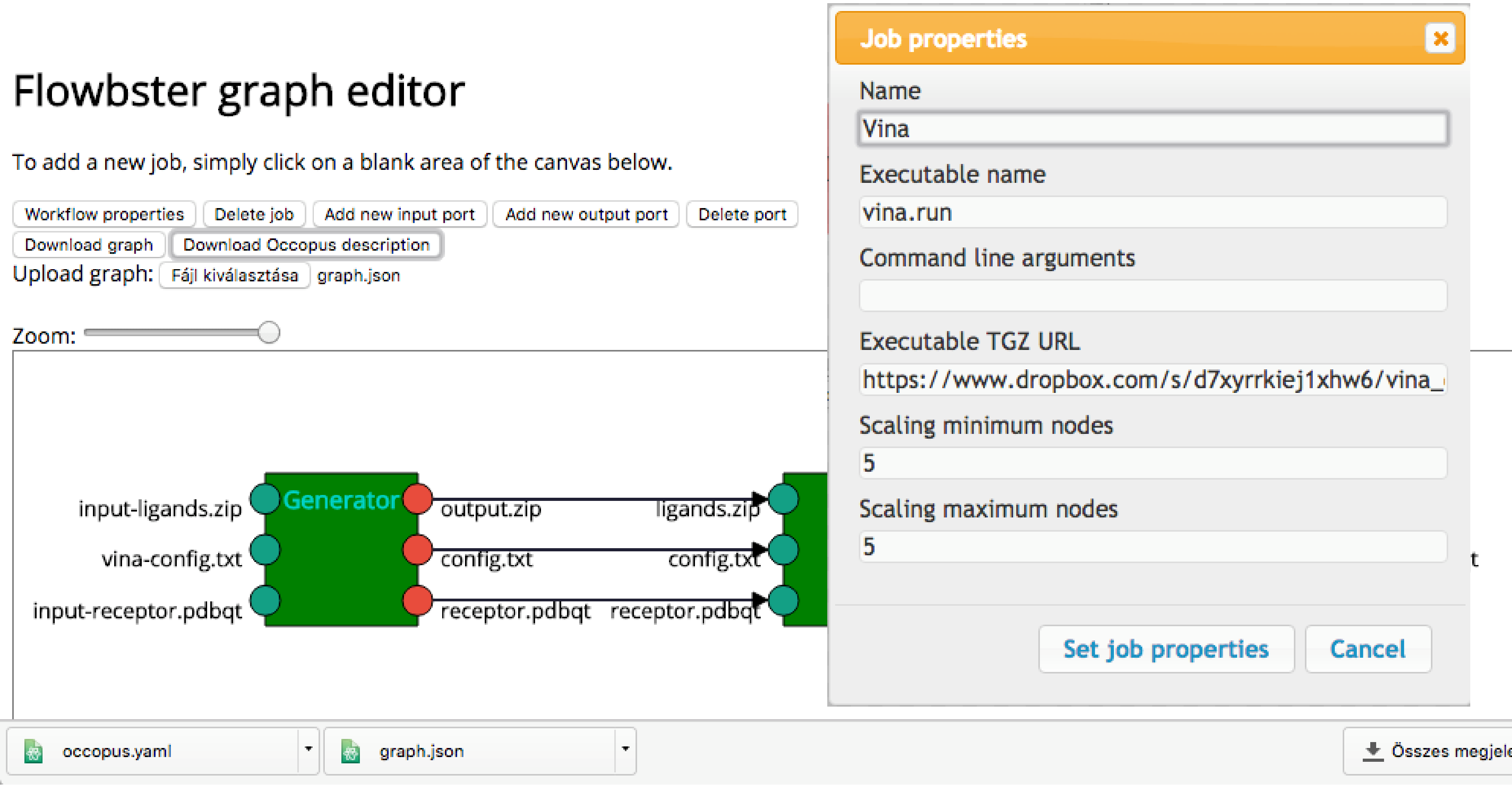
Automatically generated from the graphical view

```
- &Vina
  name: Vina
  type: flowbster_node
  scaling:
    min: 5
    max: 5
  variables:
   jobflow:
    app:
     exe:
      filename: vina.run
      tgzurl: http://foo.bar/vina.tgz
     args: ''
     in:
      -
       name: ligands.zip
      -
       name: config.txt
      -
       name: receptor.pdbqt
     out:
      -
       name: output.tar
       targetname: output.tar
       targetnode: COLLECTOR
```

Define parallelism



Flowbster Graphical Design Layer



Flowbster graph editor

To add a new job, simply click on a blank area of the canvas below.

Job properties

Name: Vina

Executable name: vina.run

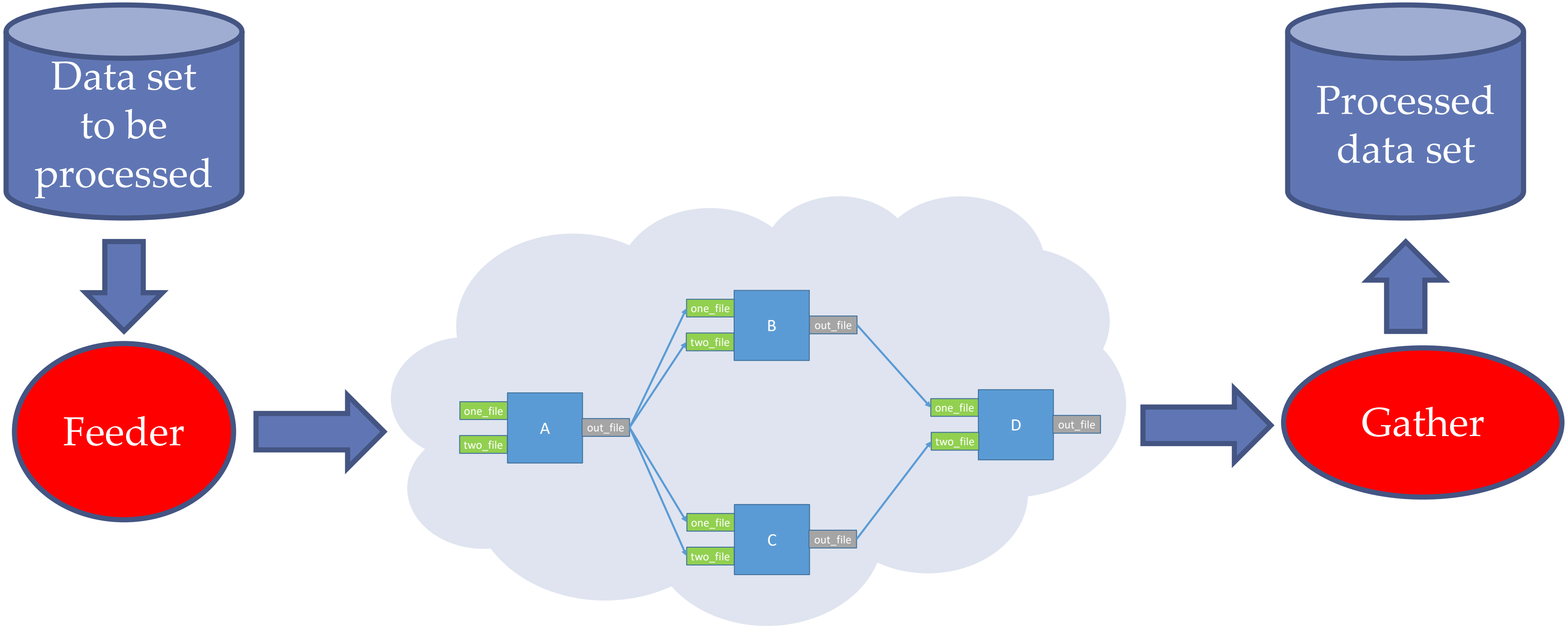
Command line arguments:

Executable TGZ URL: https://www.dropbox.com/s/d7xyrrkiej1xhw6/vina_

Scaling minimum nodes: 5

Scaling maximum nodes: 5

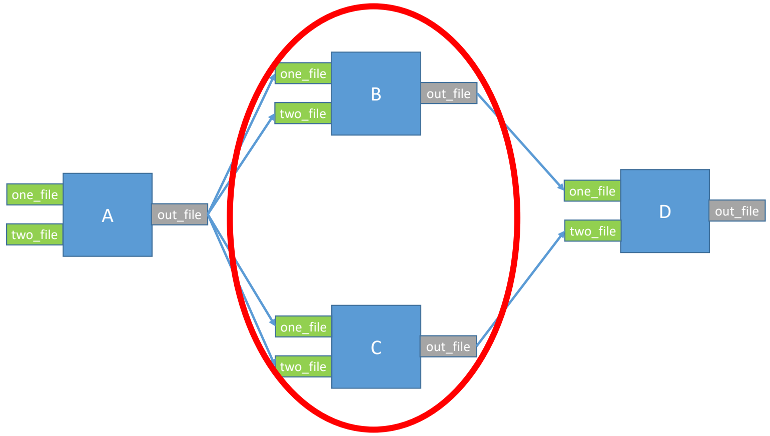
Feeding and Gathering Data Set Elements



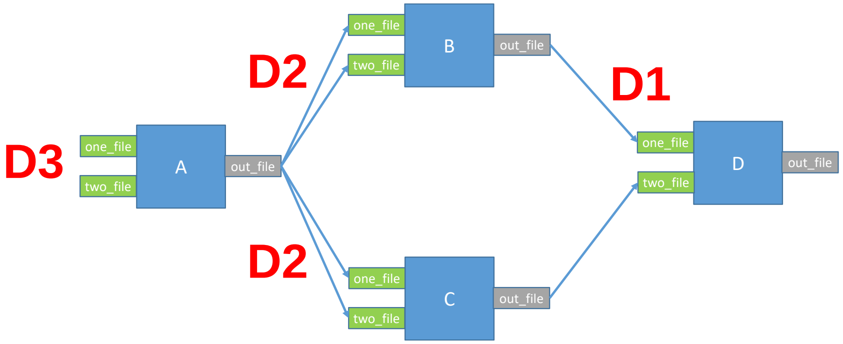
- Feeder: **not part of Flowbster, should be written by the user**
  - Command line tool
  - Feeds a given node/port of Flowbster workflow with input data items
- Gather: **not part of Flowbster, should be written by the user**
  - Web service acting as a receiver
  - Transfers the incoming data items into the target storage

Exploitable Parallelisms in Flowbster

- **Parallel branch parallelism**



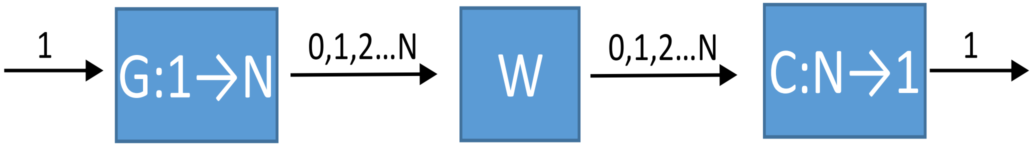
- **Pipeline parallelism**



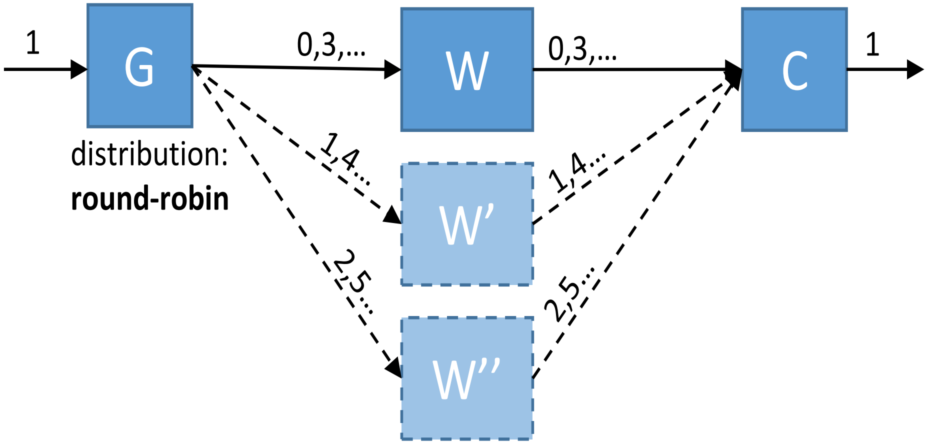
- **Node scalability parallelism**

Node Scalability Parallelism in Flowbster

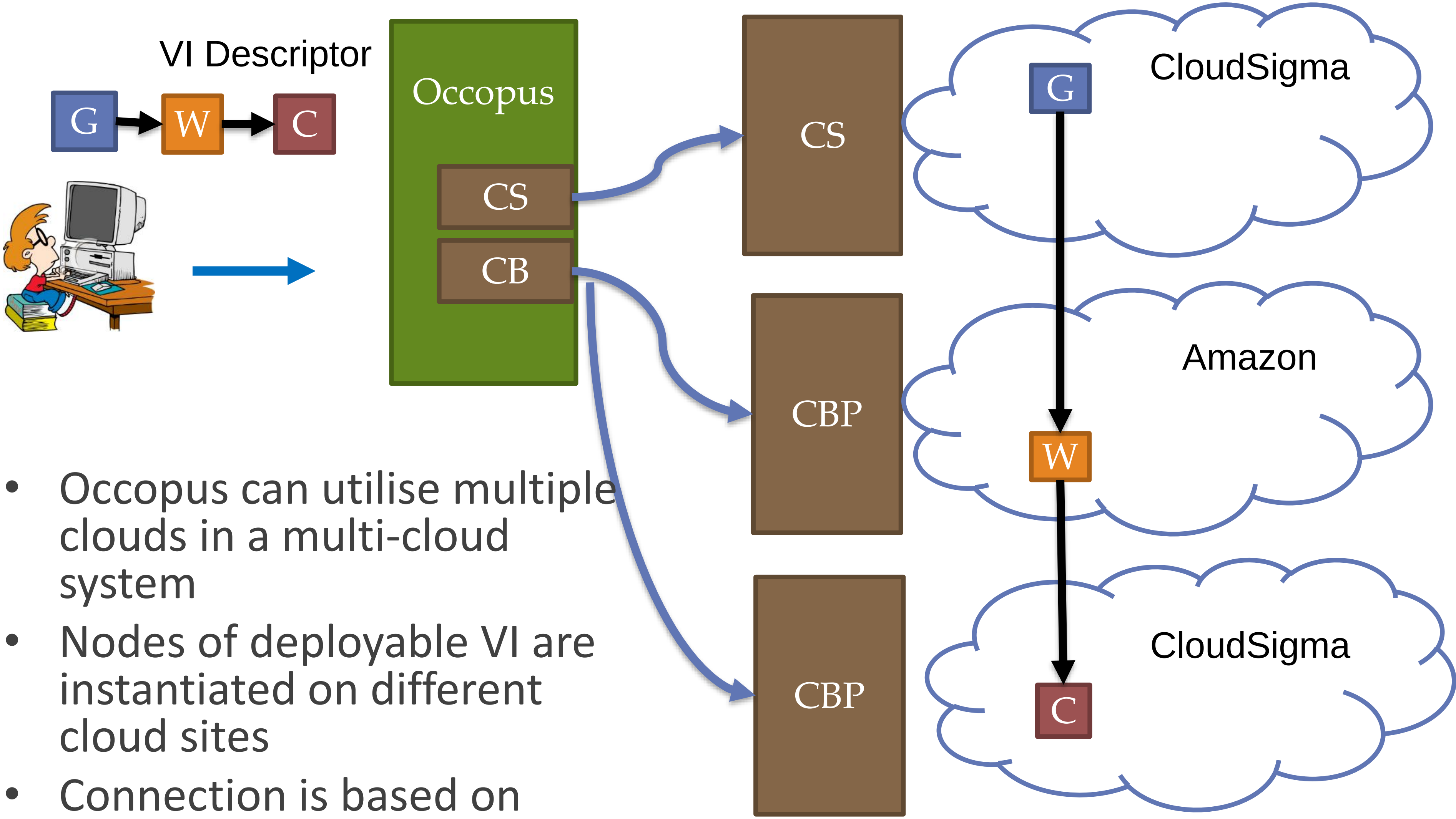
Generator-Worker-Collector parameter sweep processing pattern:



- The Generator generates N output data from 1 input data
- The Worker should be executed for every input data -> **N Worker instances can run in parallel** for processing the N data
- The Collector collects the N results coming from the N Worker instances and after processing them creates 1 output data

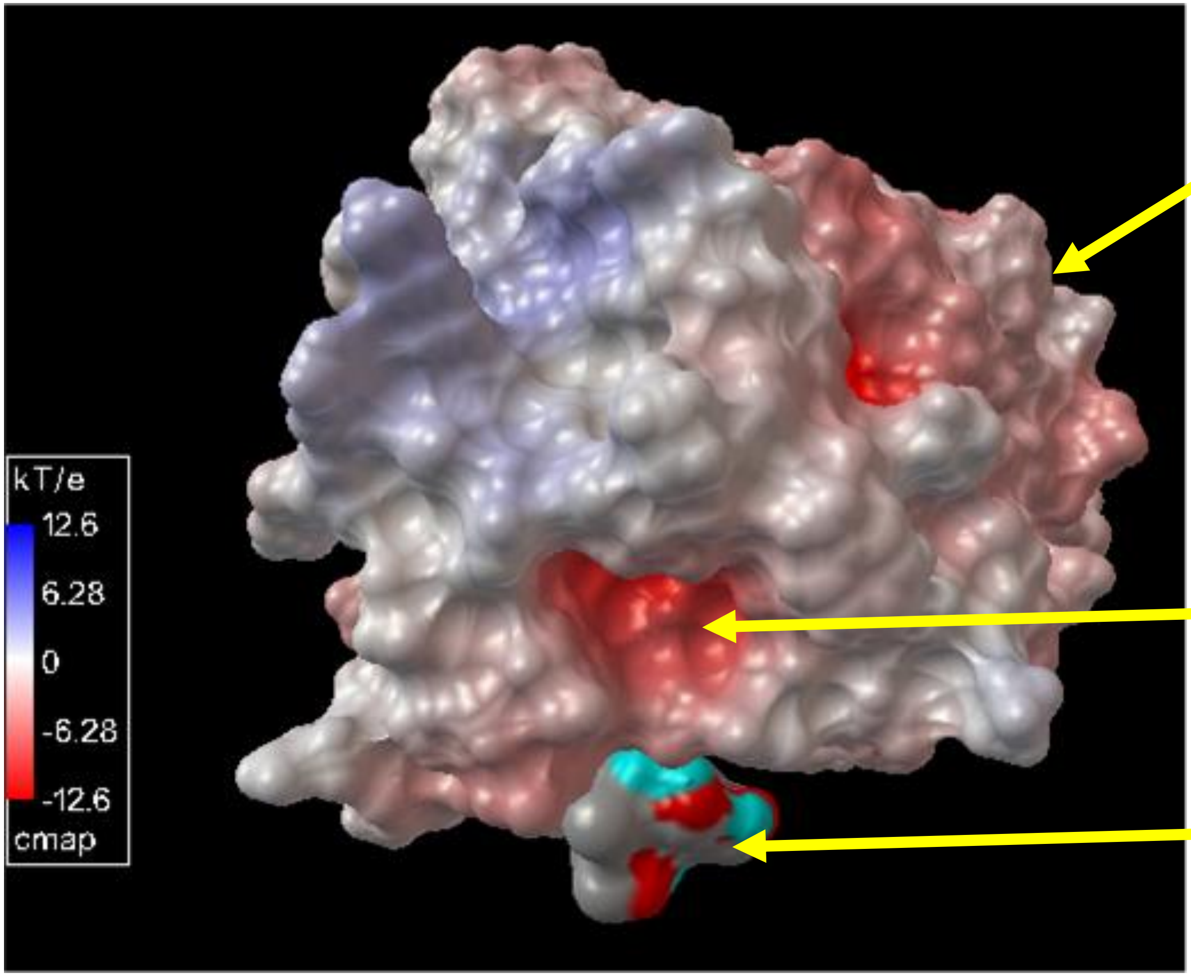


Heterogeneous Multi-Cloud Setup of Flowbster



- Occopus can utilise multiple clouds in a multi-cloud system
- Nodes of deployable VI are instantiated on different cloud sites
- Connection is based on public ips

Autodock Simulation to Measure the Performance

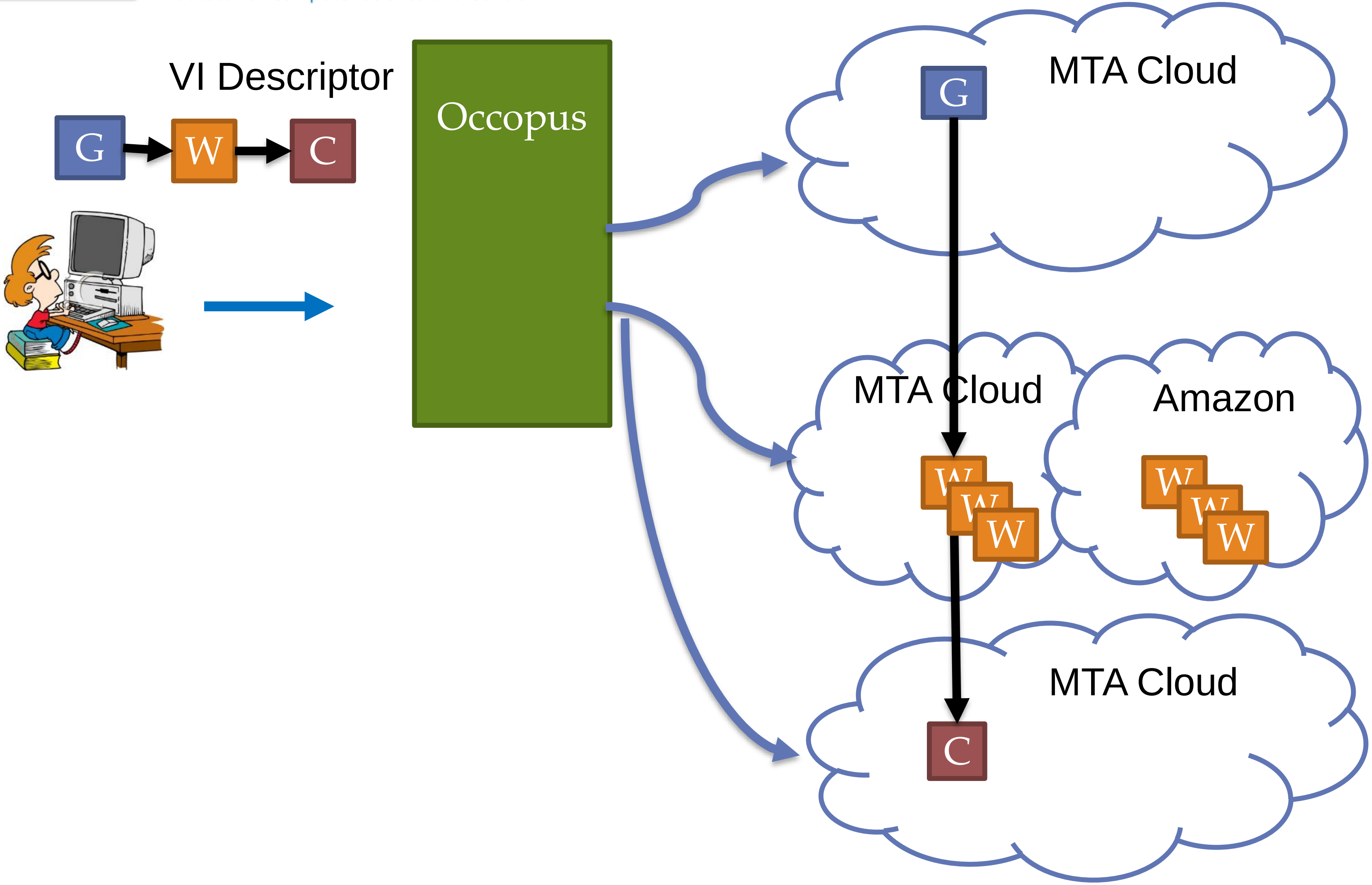


Protein (receptor)

Binding pocket

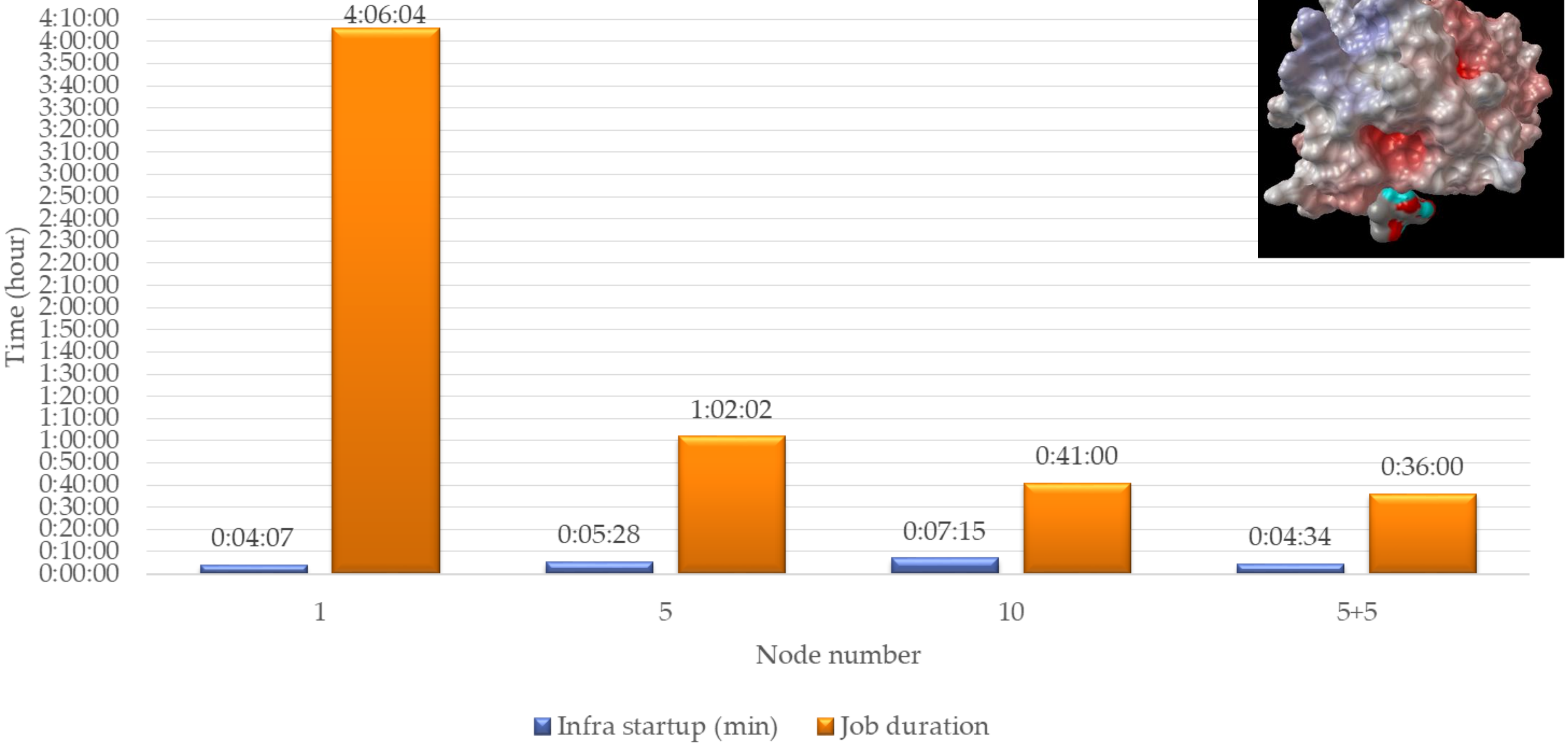
Sugar (ligand)

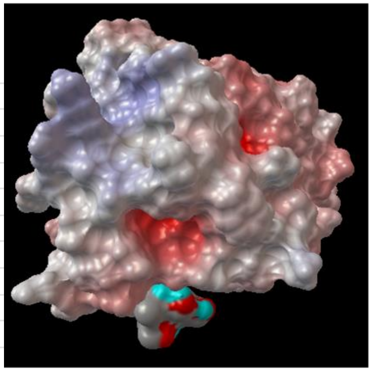
Single and Multi-cloud Setup of Measurement



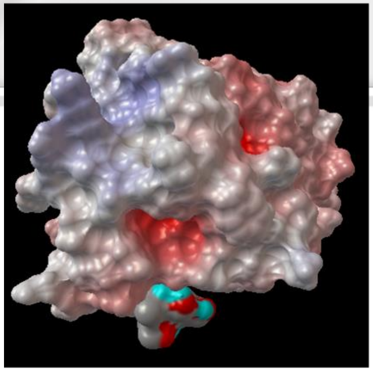
Performance Results

Autodock simulation execution time on MTA Cloud and Amazon (3840 molecules, 240 data item each containing 16 molecules)

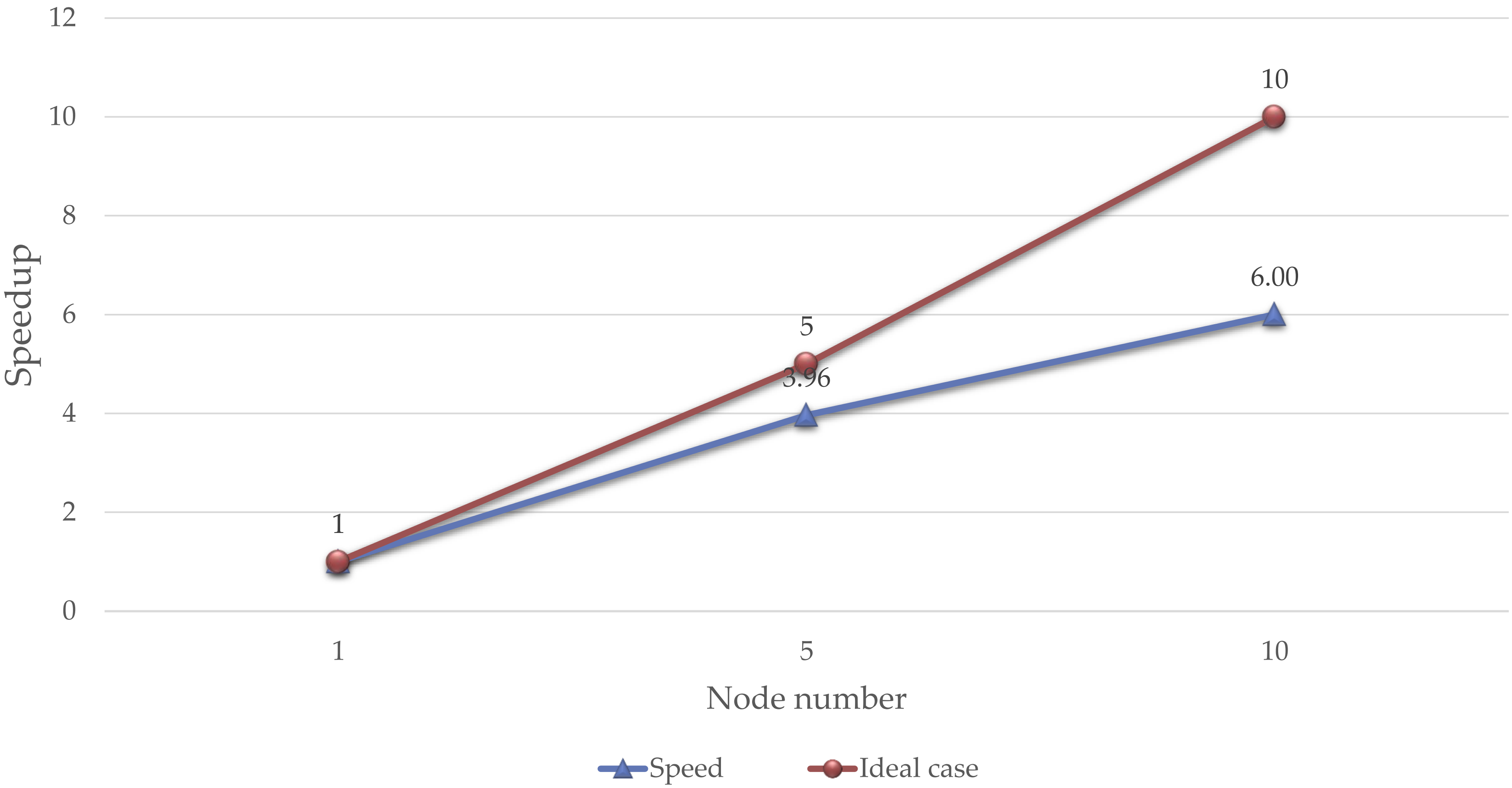




Performance Results



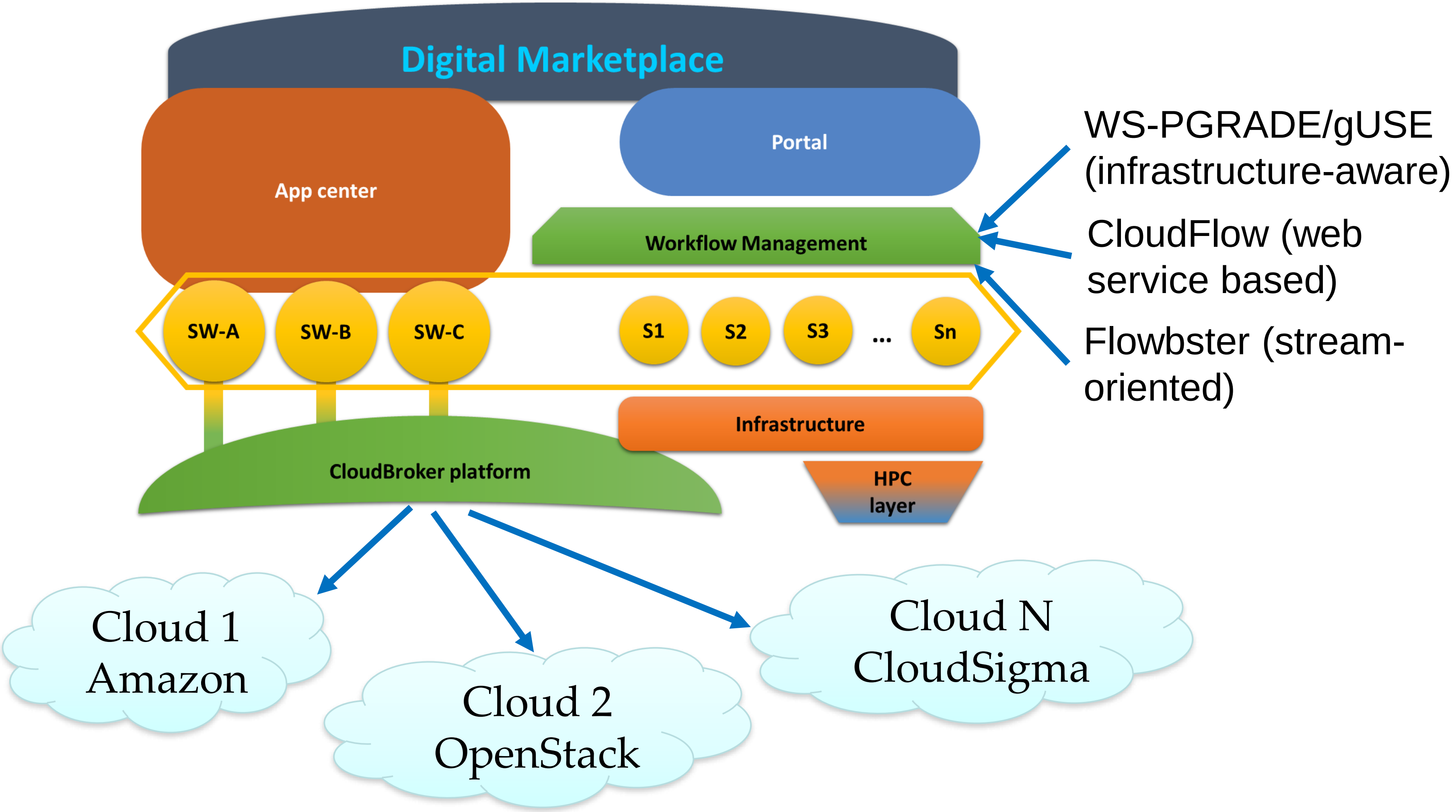
Autodock simulation speedup on MTA Cloud



Current State of Flowbster

- Open-source (License: Apache v2)
- Running prototype
- Available at github: https://github.com/occopus
- Documentation under development:
  - Users' Guide
  - Developers' Guide
  - Tutorials
- Further development plans
  - Dynamic scalability for node scalability parallelism
  - Built-in error diagnostic and fault-recovery mechanism

Result: Flexible Marketplace for CloudiFacturing



Digital Marketplace

App center

Portal

Workflow Management

SW-A

SW-B

SW-C

S1

S2

S3

...

Sn

Infrastructure

CloudBroker platform

HPC layer

WS-PGRADE/gUSE (infrastructure-aware)

CloudFlow (web service based)

Flowbster (stream-oriented)

Cloud 1 Amazon

Cloud 2 OpenStack

Cloud N CloudSigma

Conclusions

- The workflow ecosystem is very rich (rather too rich) that prevents the sharing and reusing of existing workflows
- The talk showed how clouds can facilitate the solution of this problem
- The introduction of infrastructure-aware workflows combined and implemented with cloud orchestrators can significantly increase the flexibility of executing workflows on various virtual infrastructures
- The usage of stream-oriented, service choreography based workflows in clouds can accelerate the processing of large scientific date sets

Thank You!

Any Questions?