h e p i a

Haute école du paysage, d'ingénierie
et d'architecture de Genève

Hes·SO GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Combining Grid & Cloud for e-sciences applications

Nabil Abdennadher, HES-SO, hepia
Switzerland

Hes·SO GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

L'avenir est à créer

h e p i a

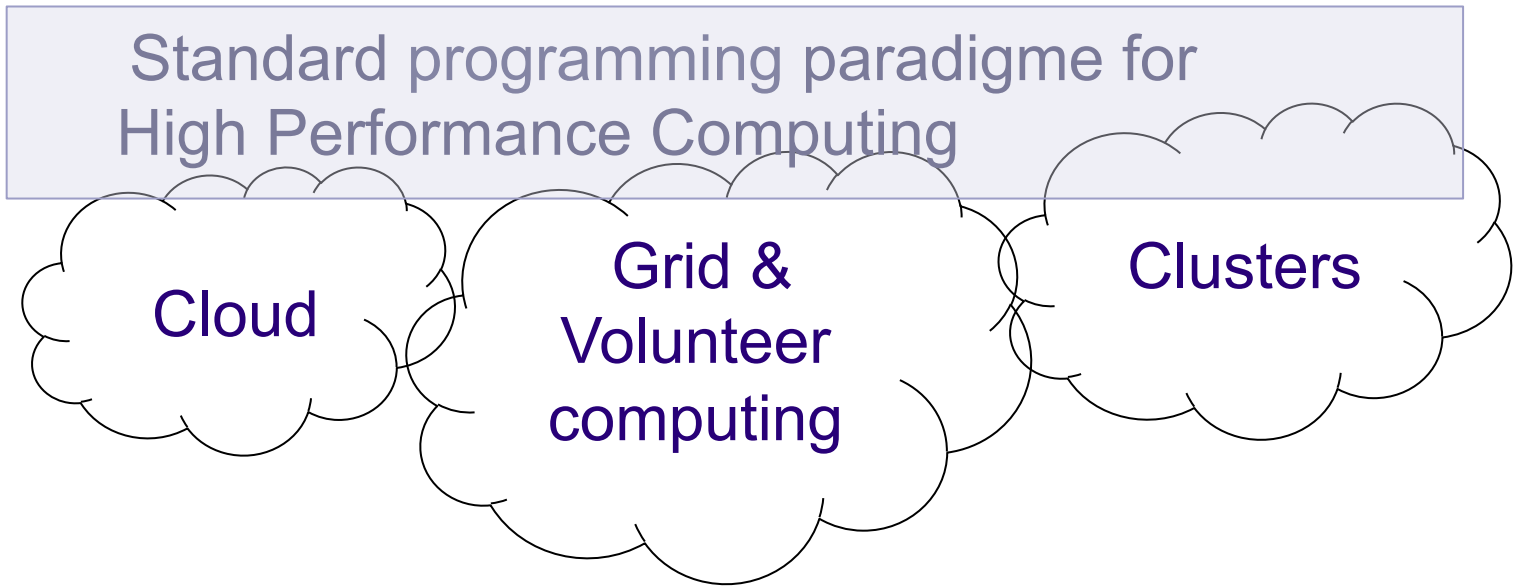
Haute école du paysage, d'ingénierie
et d'architecture de Genève

The context

- High performance computing …
  - Massively parallel machines
  - Clusters
  - Grid computing
  - Volunteer computing
  - Cloud computing
- … for high performance applications (e-sciences)
  - Medecine
  - Life science & biotechnology
  - Chemesrtry
  - Physics

I have two **dreams …**

- Write once, run anywhere (cloud, volunteer computing, grid, cluster) : portability
- Use several infrastructures for the same deployment : Interoperability



Let's start ….

Outline

- Cloud vs. Grid & Volunteer Computing
- XtremWeb-CH (XWCH)
- Combining XtremWeb-CH with Cloud
  - Amazon
  - OpenStack
  - Azure
- Conclusion

- 11:00 – 12:15 : Grid vs. Cloud + XtremWeb-CH (Architecture)
- 12:15 – 13:15 : Lunch
- 13:15 – 14:15 : Programming with XWCH
- 14:15 – 16:00 : Combining XWCH with Cloud

Grid vs. Cloud*

Criteria	Grid	Cloud
Virtualization	In its beginning	Essential
Type of application	Batch	Interactive
Development of applications	Local	In the cloud
Access	Grid middleware	Standard Web protocols
Organizations	Virtual	Physical
Business models	Sharing	Pricing (utility model)

* ***Cloud Computing – A classification, Business Models, and Research Directions.*** **Business and information systems Engineering, 2009**

Grid vs. Cloud*

Criteria	Grid	Cloud
SLA	Absent	Essential
Control	Decentralized	Centralized (data center)
Openness	High	Low
Ease of use	Hard	Easy
Switching cost	Low (due to "standardization")	High (due to incompatibilities)
Availability	Low	High

** Cloud Computing – A classification, Business Models, and Research Directions.* **Business and information systems Engineering, 2009**

Grid and cloud are complementary and not competitors

Outline

- Cloud vs. Grid & Volunteer Computing
- **XtremWeb-CH (XWCH)**
  - **Architecture**
  - Programming with XWCH
- Combining XtremWeb-CH with Cloud
  - Amazon
  - OpenStack
  - Azure
- Conclusion

h e p i a
Haute école du paysage, d'ingénierie et d'architecture de Genève

XtremWeb-CH @ a glance (www.xtremwebch.net)

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

- A volunteer computing middleware
  - developped at univ. of applied sciences, Western Switzerland (HES-SO) **since 2003**
  - Can combine resources coming either from institutions or individuals
  - Can bypass firewalls and NAT
  - Manages the volatility of resources
  - Easy to install and monitor
  - Easy to program : XWCHClient API

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

Applications

MetaPIGA : a robust implementation of several stochastic heuristics for large phylogeny inference.

Domain : life science

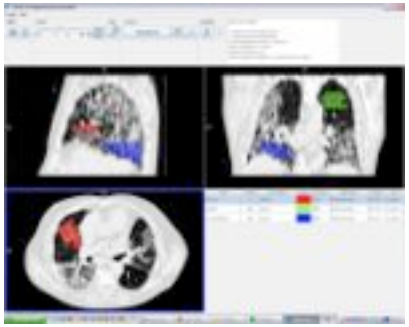
Selector : Reconstruction of modern humans' prehistoric migrations in East Asia

Domain : genetics

Applications

The Neurad software provides a fast and accurate evaluation of radiation doses (treatment of cancerous tumours).

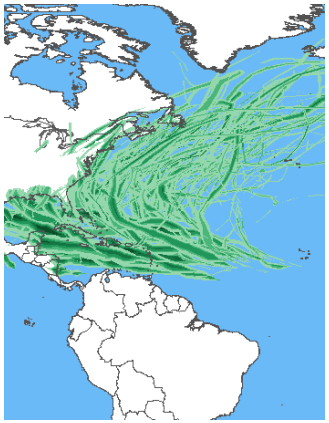
Domain : medicine



Gift, a content-based image indexing and retrieval package.

Domains: medicine and art.

Applications



Cyclone : to analyse the risk of cyclones

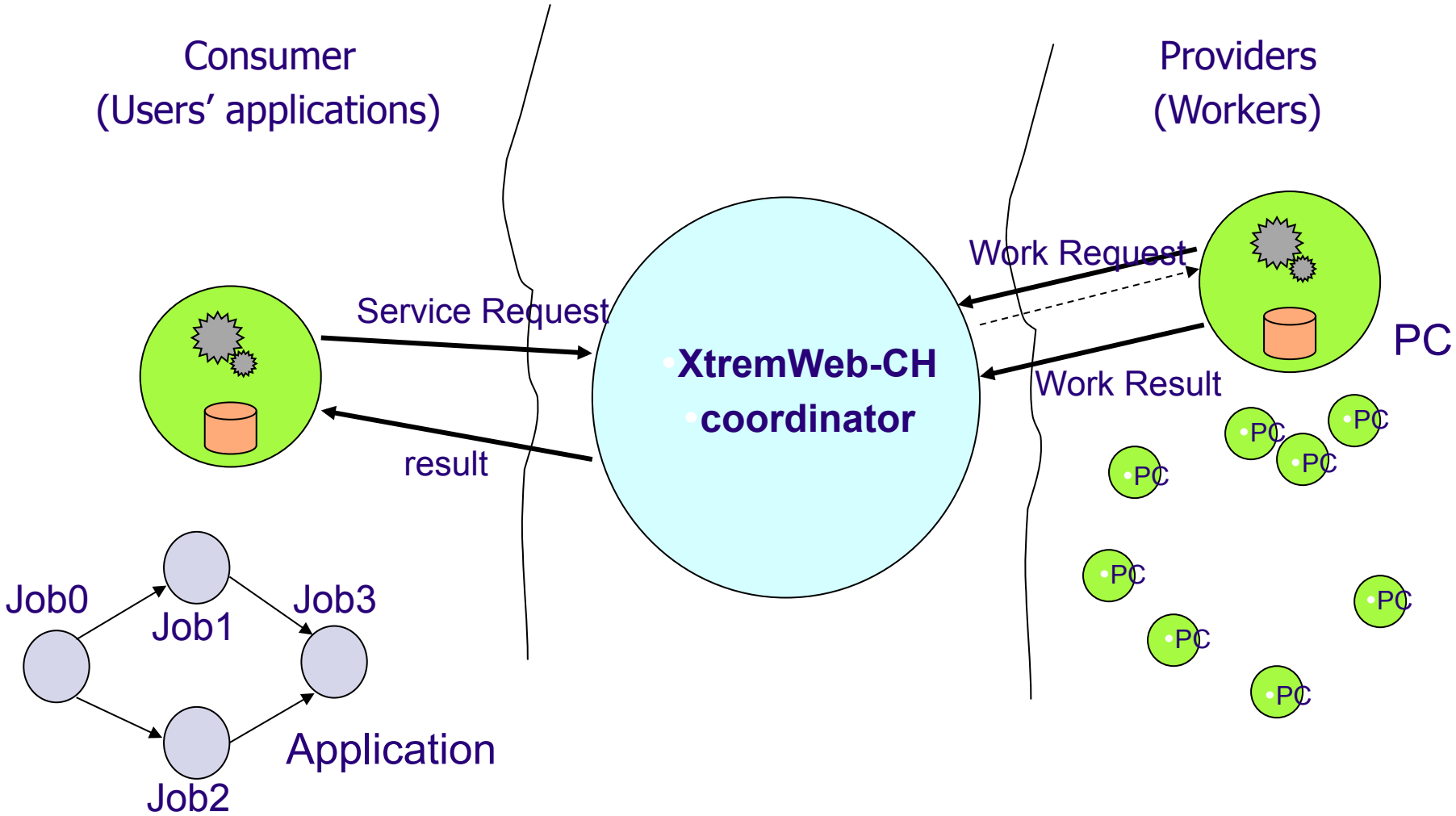
Domain : Environement

CleanCity : to measure and analyse data related to air pollution in the cities

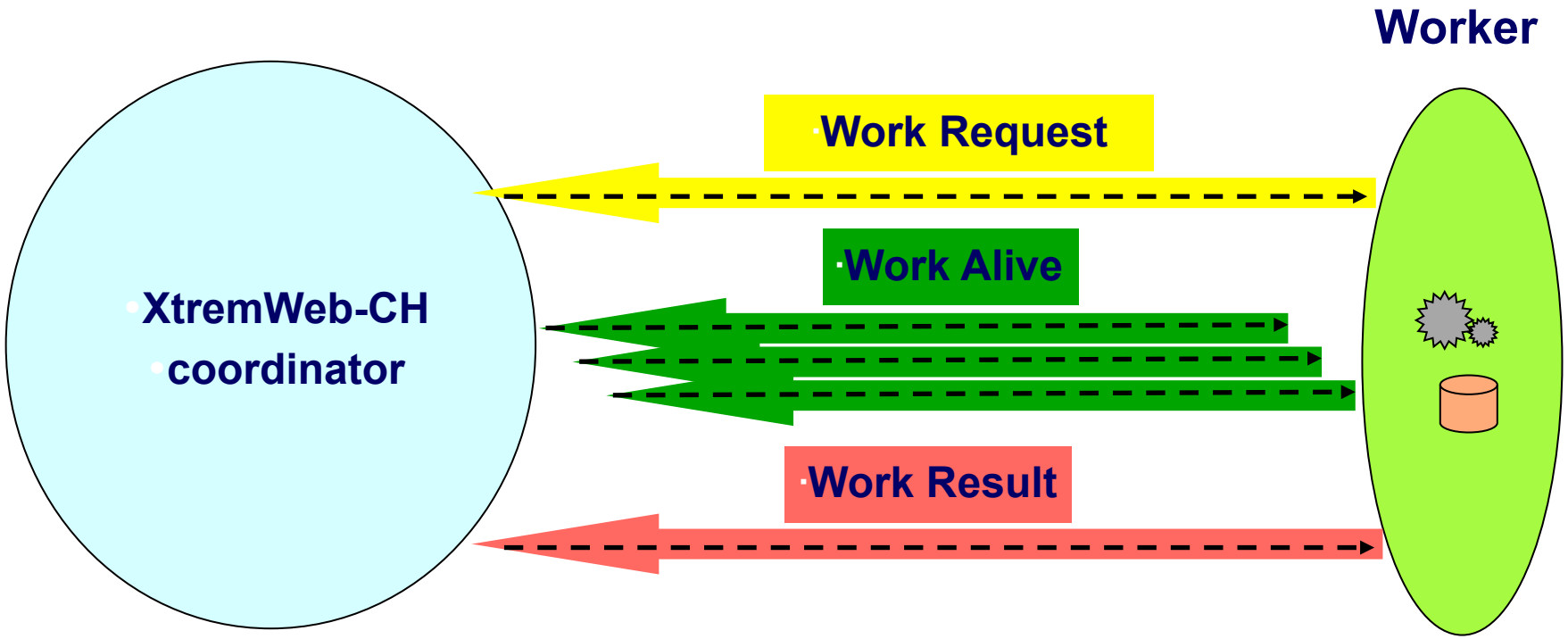
Domain : Environement (urban climate)

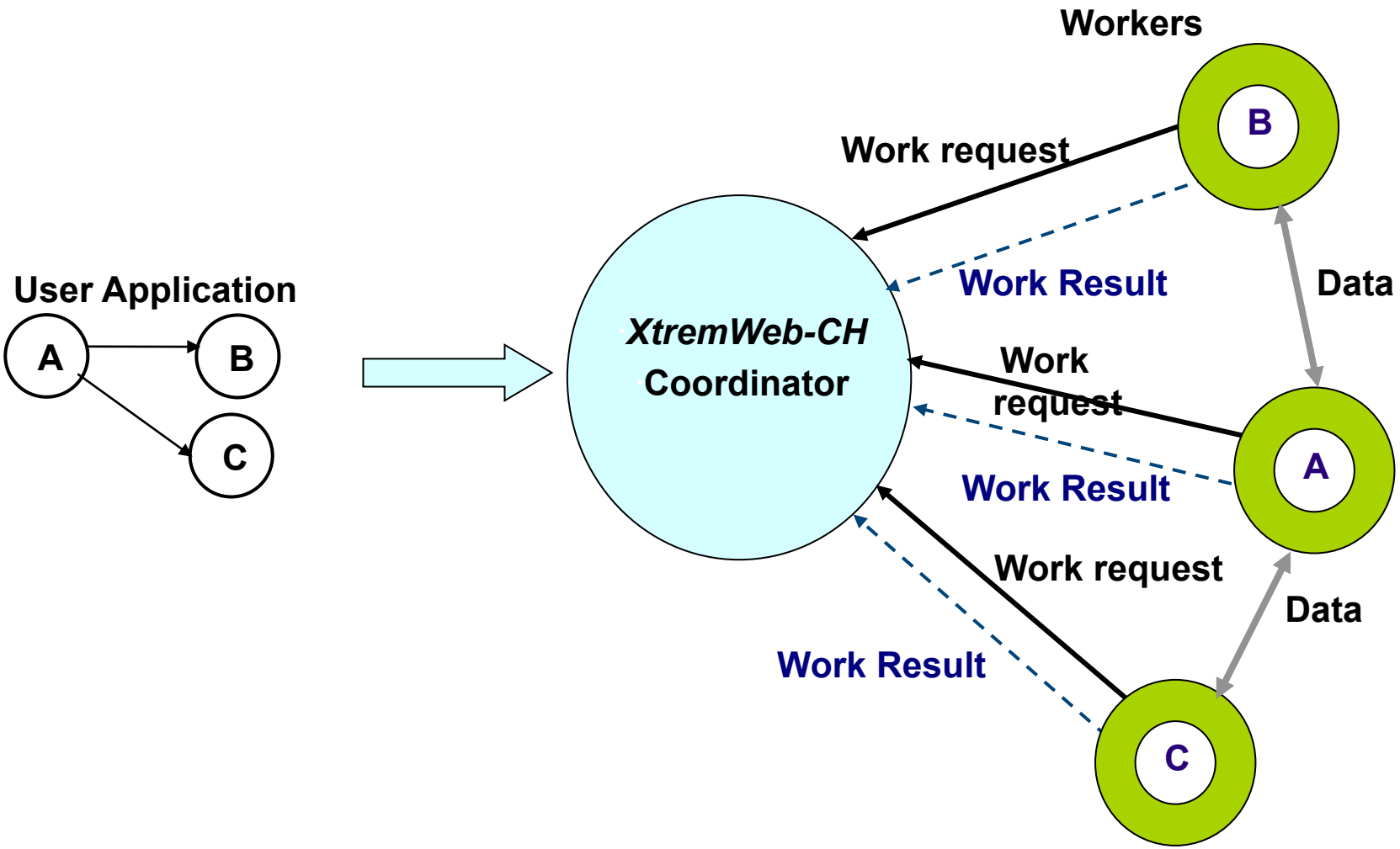
Learned lessons

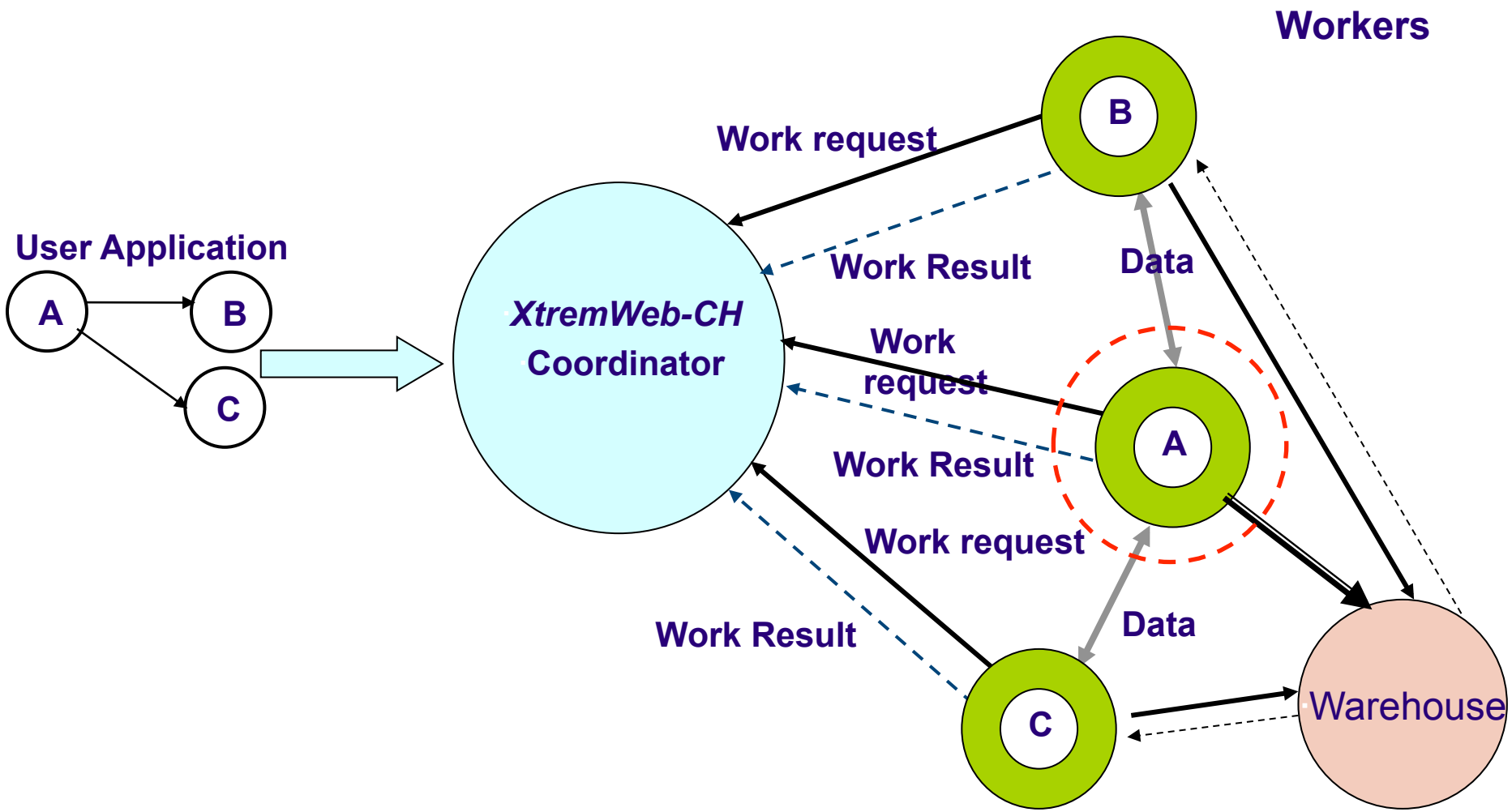
- Nodes volatility
- Nodes are not dedicated to the volunteer platform
- Nodes are administrated by IT support of different institutions with different security politics,
- We have only 250 connected nodes (out of 1'000 computers)
- Resources required by jobs are often not supported by XtremWeb-CH (XWCH) nodes

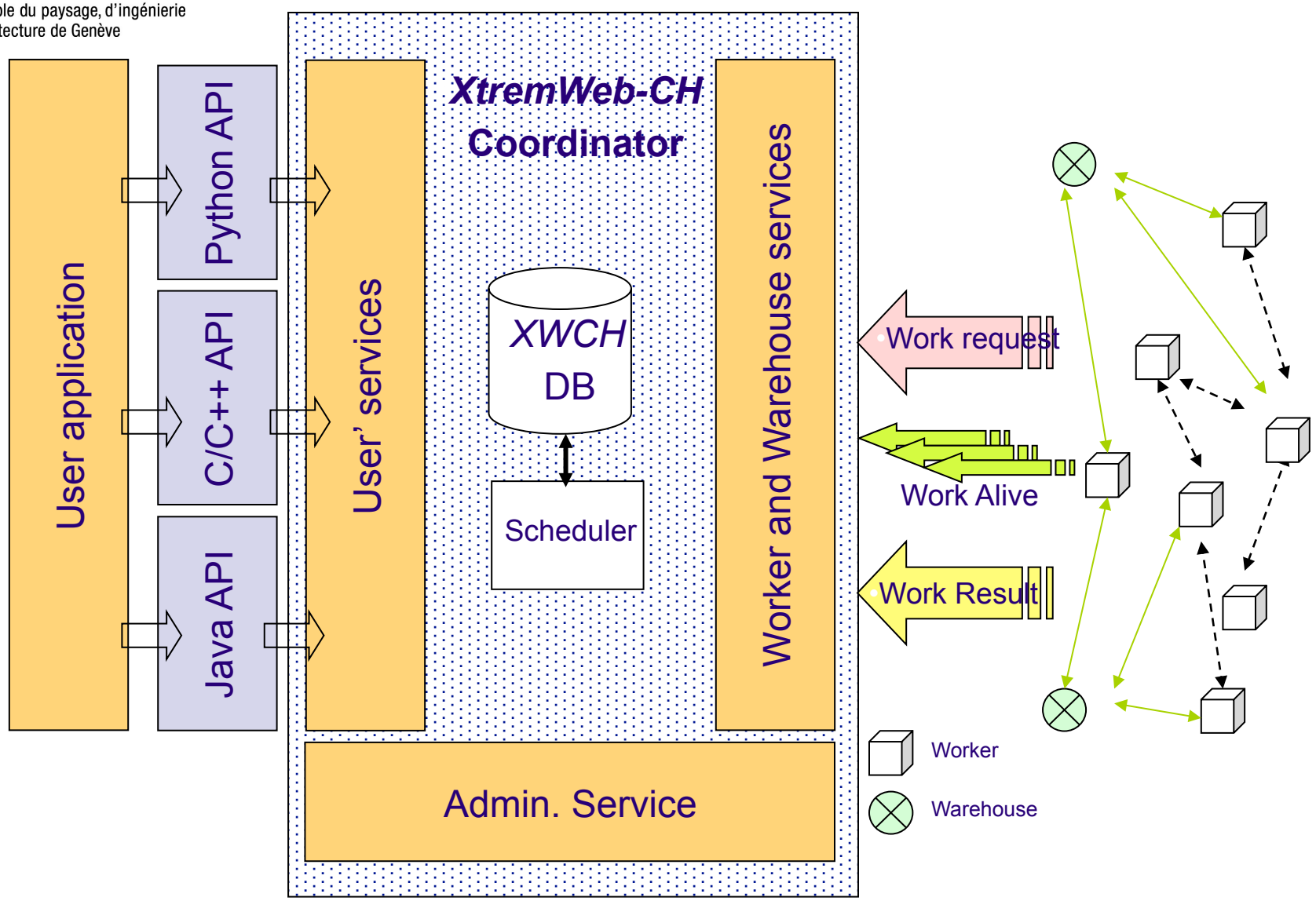

Consumer
(Users' applications)
Providers
(Workers)
Service Request
result
XtremWeb-CH
coordinator
Work Request
Work Result
PC
PC
PC
PC
PC
PC
PC
PC
Job0
Job1
Job3
Job2
Application

Communication protocol




Workers
B
Work request
Work Result
Data
User Application
A
B
C
XtremWeb-CH
Coordinator
Work
request
A
Work Result
Work request
Data
Work Result
C


Workers
User Application
A
B
C
XtremWeb-CH
Coordinator
Work request
Work Result
Work
request
Work Result
Work request
Work Result
Data
Data
B
A
C
Warehouse


User application
Python API
C/C++ API
Java API
User' services
XtremWeb-CH
Coordinator
XWCH
DB
Scheduler
Worker and Warehouse services
Admin. Service
Work request
Work Alive
Work Result
Worker
Warehouse

XWCH components

Coordinator

accepts execution requests coming from clients,

assigns tasks to workers (pre-assignment)

transfers binary codes to workers (if necessary),

detects worker crash/disconnection

Workers

initiate connection with the coordinator,

ask for work (pull model),

execute jobs

save results locally and on one or several warehouses

inform the coordinator about their "status"

XWCH components

Warehouses

used by workers to download input data needed to execute tasks and/or upload output data produced by tasks.

a warehouse acts as a repository or file server.

Client

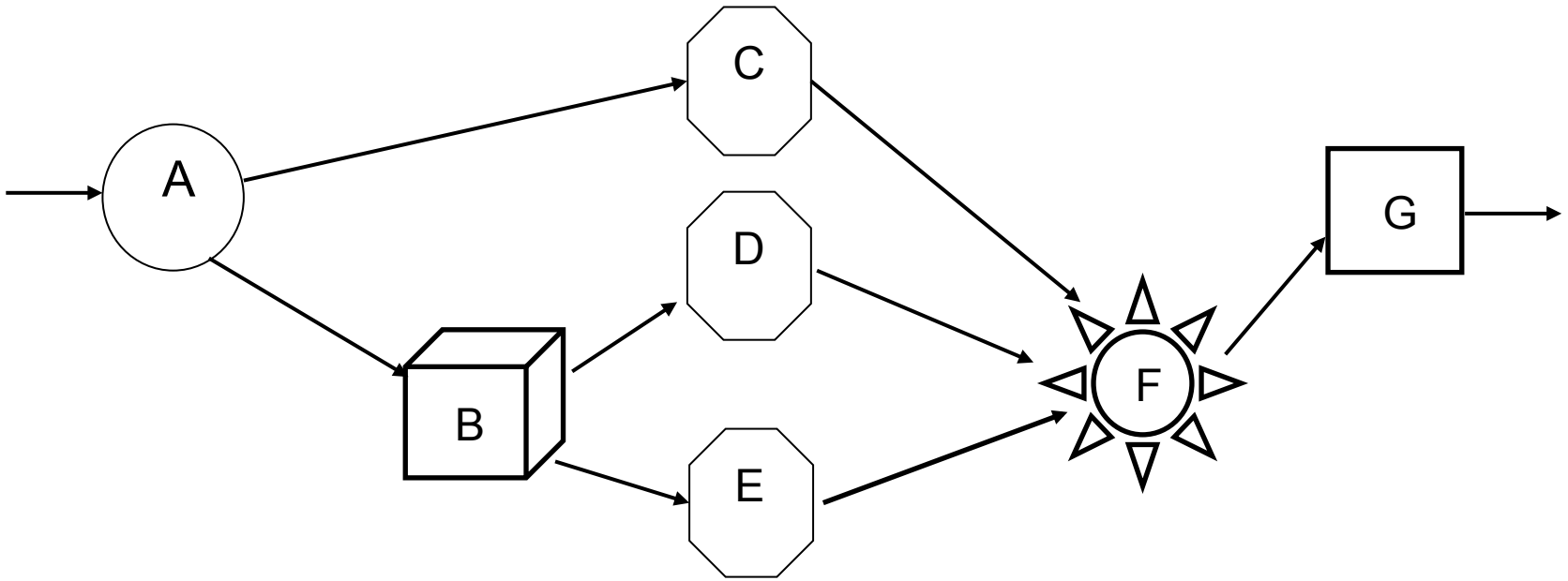
an ordinary program (written in Java, C/C++ or Python) which uses the XWCH API

Connecting your worker to our platform

- How to install a worker on your platform: http://www.xtremwebch.net/distrib/XWCH-Installer.jar
- file config.properties

XWCH application

- Application : set of jobs
- Modules : set of binaries having the same "behaviour" (they often have the same source code)
- Jobs : Execution of one module (one binary) on one worker



XWCH users

- Users are identified by a client ID
- Users can be:
  - Platforms' administrators
    - Can create empty modules
    - Give access rights to users (modules administrators, simple users)
  - Modules administrators
    - Can fill modules with binaries
    - Can execute applications
  - Simple users
    - Can only execute applications (assuming they know the modules Id to use)

hepia
Haute école du paysage, d'ingénierie et d'architecture de Genève

How to submit applications to *XWCH*?

Hes·so GENÈVE
Haute Ecole Spécialisée de Suisse occidentale

1. Initialize connection with an *XWCH* coordinator (parameters :URL of the coordinator, *XWCH* account)
2. Create the application
3. Create and initialize modules belonging to the application (assuming you are modules administrators)
4. Insert input data
5. Submit/monitor jobs
6. Download results
7. End the application

Hello World application


```
// Initialize connection (Point 1)
c = new XWCHClient (ServerAddress, ".", IdClient);
c.init();
//Create the application (Point 2)
String appid = c.addApplication(“Hello World application");
```

Hello World application


```
//Create “empty module” (Point 3)
String ModuleId = c.addModule(”MyFirstModule");
//Initialize modules by binary codes (Point 3)
String ref_Win = c.addBinary (ModuleId, windowsBinary,
PlateformEnumType.WINDOWS);

String ref_Linux = c.AddBinary (ModuleId, linuxBinary,
PlateformEnumType.LINUX);
```

Hello World application


```
// Insert input data (Point 4)
fileref freftask1 = c.addData (InputFile, appid);
String outputFile = c.getUniqueID ();
…
// Create & monitor jobs (Point 5)
String jobId = c.addJob ("MyFirst_task",          //Job description
                         appid,                   //Application identifier
                         ModuleId,                // Module identifier
                         broadcastCmdLine,        //Command line to run
                         freftask1.toJobReference(),//file1:md5; file2:md5; …
                         listOfFiles,             // list of files to compress
                         outputFile,              // name of the compressed file
                         "");
```

Extra Fields: Jobs & data management

Hello World application


```
// Retrieve results (Point 5)
while (getJobStatus (JobId)) ≠ COMPLETE);
//Download output file (Point 6)
c.getJobResult(JobId, outputFile)
// End the application (Point 7)
c.endApplication (appid)
```

Example 1


```
String j0 =        c.addJob        (”myJob”,
                                    appid,
                                    ModuleId,
                                    “generate.sh”
                                    input_ref.toJobReference(),
                                    name_application + "_0",
                                    OutPutFile,
                                    "cputotalcore;2");
```

Example 2


```
String j0 = c.AddJob    (  "myJob",
                           appid, //Application identifier
                           ModuleId,
                           “generate.sh”
                           input_ref.toJobReference(),
                           name_application + "_0",
                           OutPutFile,
                           "replication;10");
```

Example 3


```
String j0 =        c.AddJob("myJob”,
                            appid, //Application identifier
                            ModuleId,
                            "generate.bat”
                            input_ref.toJobReference(),
                            name_application + "_0",
                            OutPutFile,
                            "samejob;1234qwer7689");
```

Example 4


```
String j0 =        c.addJob        ("myJob”,
                                    appid, //Application identifier
                                    ModuleId,
                                    "generate.bat”,
                                    input_ref.toJobReference(),
                                    name_application + "_0",
                                    OutPutFile,
                                    "host;workerName");
```

Extrafields : Jobs & data management

- Syntax: *criteria*$_1$;*value*$_1$;*criteria*$_2$;*value*$_2$;...;...;
- *criteria*$_i$ should match *value*$_i$.
- *criteria*$_i$ can have several key words:
  - samejob: the job must run on the same node as job *Jobid*. In this case, *value*$_i$ is equal to *Jobid*.
  - replication: result files are replicated "nb" times. In this case, *value*$_i$ is equal to *nb*.
  - host: indicates which worker will execute the job. In this case, *value*$_i$ is equal to the name of the worker.

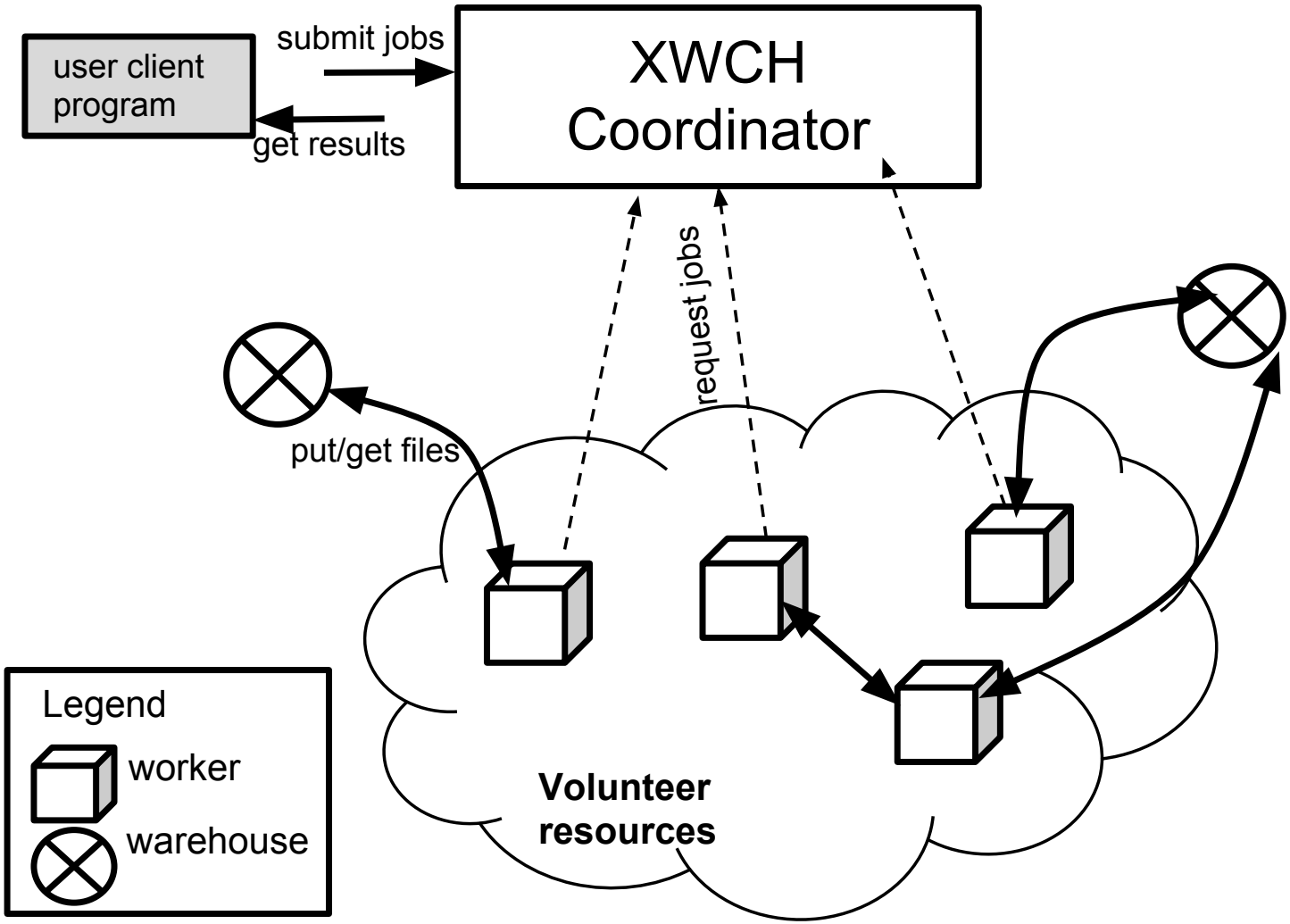
PovRay (Ray Tracing) application

- Create your XWCH account
- Check if you are allowed to execute the two modules PovRay and combineppm
- Import the modulesID and clientID to your client program
- Check Extrafields (not needed for this first exercise)
- Choose image to synthesis
- Compile : *sh compile.sh*
- Execute: *sh runPovRayExample.sh*
- Monitor your application

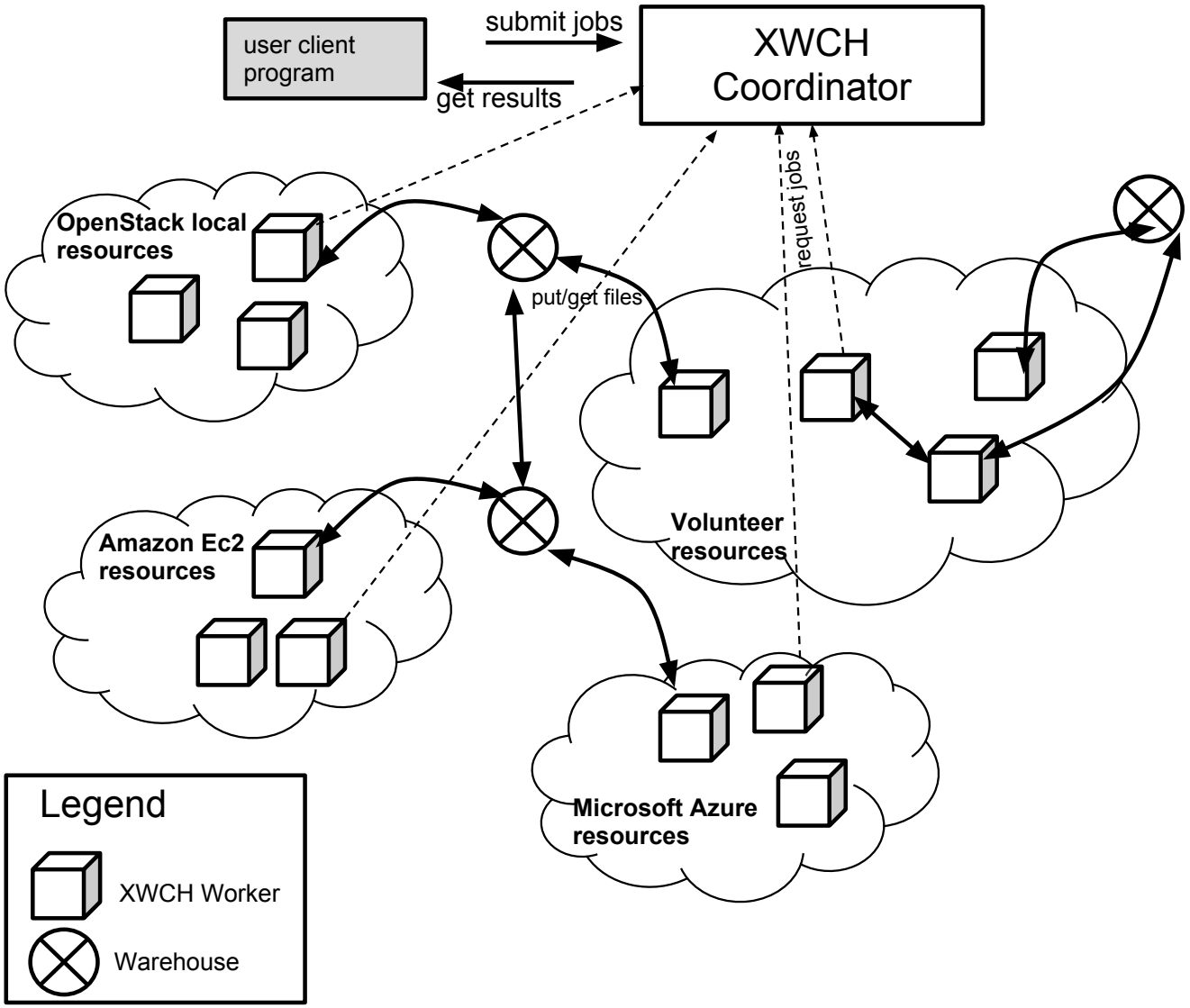
Outline

- Cloud vs. Grid & Volunteer Computing
- XtremWeb-CH (XWCH)
- **Combining XtremWeb-CH with Cloud**
  - Amazon
  - OpenStack
  - Azure
- Conclusion

XWCH as a Volunteer computing platform …



Using XWCH API on Cloud & VC infrastructure



skeleton …

```
//Create connection with the cloud
CloudProvider amazon=getAmazonCloudProviderImpl (….);
CloudProvider openStack=getOpenStackCloudProviderImpl (….);

//Install and run XWH worker modules on the VMs
XWCHCloudLauncher amazonWorker= new XWCHCloudLauncher (amazon);
XWCHCloudLauncher openStackWorker= new XWCHCloudLauncher (openStack);

String prefix="MyWorker";
String coordURL="http://URL-coordinator:port";
amazonWorker.runOnVirtualMachines (20, prefix+"Amazon", coordURL);
openStackWorker.runOnVirtualMachines (30, prefix+"OpenStack", coordURL);
// run the XWCH application
…
//Delete the VMs
amazonWorker.deleteFromVirtualMachines();
openStackWorker.deleteFromVirtualMachines();
```

XWCH/Amazon connector


```
//Create Amazon VMs (connection with Amazon)
CloudProviderImpl_I amazonHandler=
this.getAmazonCloudHandler(AWSAccessKeyId, AWSSecretKey,
ami, OSUserName, instanceType, amazonEndPoint);
//Install XWCH worker module on Amazon VMs
xwchWorker= new XWCHCloudLauncher (amazonHandler);
```

- *CloudProviderImpl_I* class provides low level routines:
  - start virtual machines,
  - get virtual machine status,
  - execute a script on virtual machines,
  - stop virtual machines,
  - etc.

XWCH/Amazon connector


```
//Install XWCH worker module on Amazon VM

xwchWorker= new XWCHCloudLauncher (amazonHandler);
```

- *XWCHCloudLauncher* class provides high level routines:
  - start workers,
  - stop workers,
  - etc.

XWCH/Amazon connector


```
//Prepare and configure the workers to create
Map<String,String> workerParams= new
HashMap<String,String>();
workerParams.put("xwserver", xwServerURL);
workerParams.put("workername", workerNamePrefix);
workerParams.put("port", ""+workerPort);

//Create workers
String
status=xwchWorker.runOnVirtualMachines(numberOfWorkers,
workerParams);

//Stop workers
xwchWorker.deleteFromVirtualMachines();
```

XWCH/OpenStack connector


```
//Create Amazon VMs (connection with OPenStack)
CloudProviderImpl_I openStackHandler= new
CloudProviderImplOpenStack(openStackEndPoint, username,
password, project, imageId, flavorId, keyPairName);
//Install XWCH worker module on Amazon VMs
xwchWorker= new XWCHCloudLauncher (OpenStackHandler);
```

- *CloudProviderImpl_I* class provides low level routines:
  - start virtual machines,
  - get virtual machine status,
  - execute a script on virtual machines,
  - stop virtual machines,
  - etc.

XWCH/OpenStack connector


```
//Install XWCH worker module on OpenStack VM

xwchWorker= new XWCHCloudLauncher (openstackHandler);
```

- *XWCHCloudLauncher* class provides high level routines:
  - start workers,
  - stop workers,
  - etc.

XWCH/OpenStack connector


```
//Prepare and configure the workers to create
Map<String,String> workerParams= new
HashMap<String,String>();
workerParams.put("xwserver", xwServerURL);
workerParams.put("workername", workerNamePrefix);
workerParams.put("port", ""+workerPort);

//Create workers
String
status=xwchWorker.runOnVirtualMachines(numberOfWorkers,
workerParams);

//Stop workers
xwchWorker.deleteFromVirtualMachines();
```

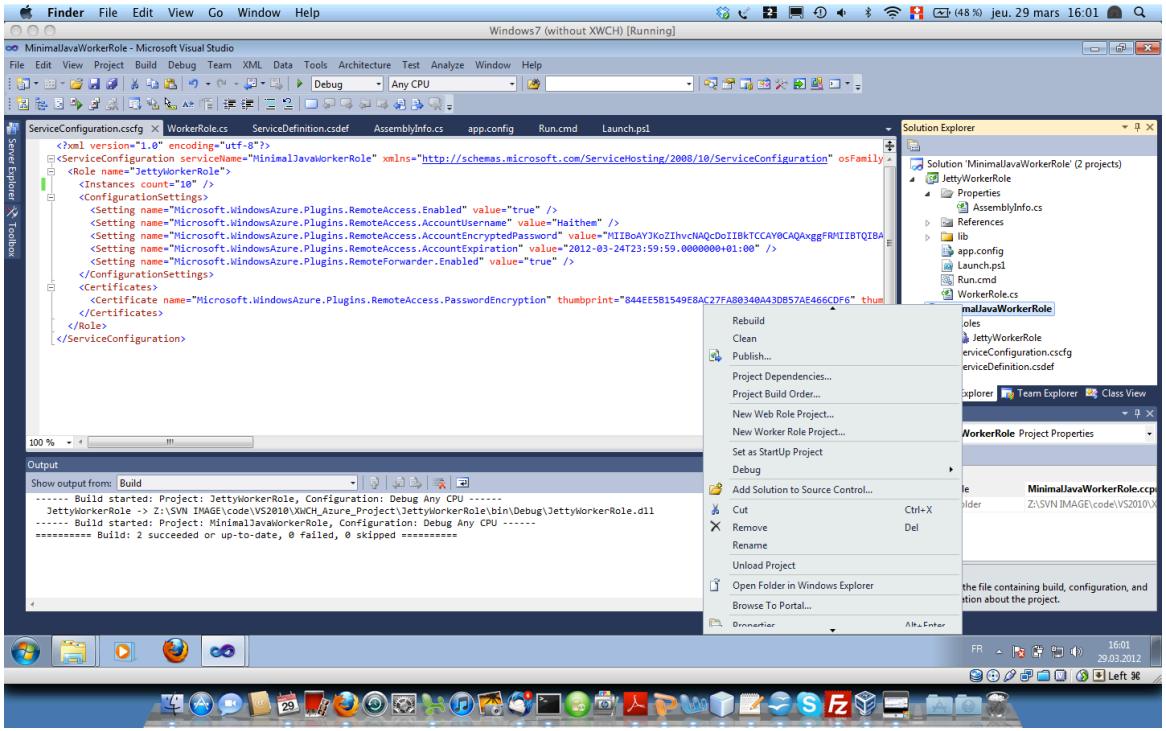
XWCH/Azure connector

IDE : Visual Studio

Azure application is represented by two files:

.cspkg (binary)

.cscfg (configuration) : XML file containing URL, certificates, etc.

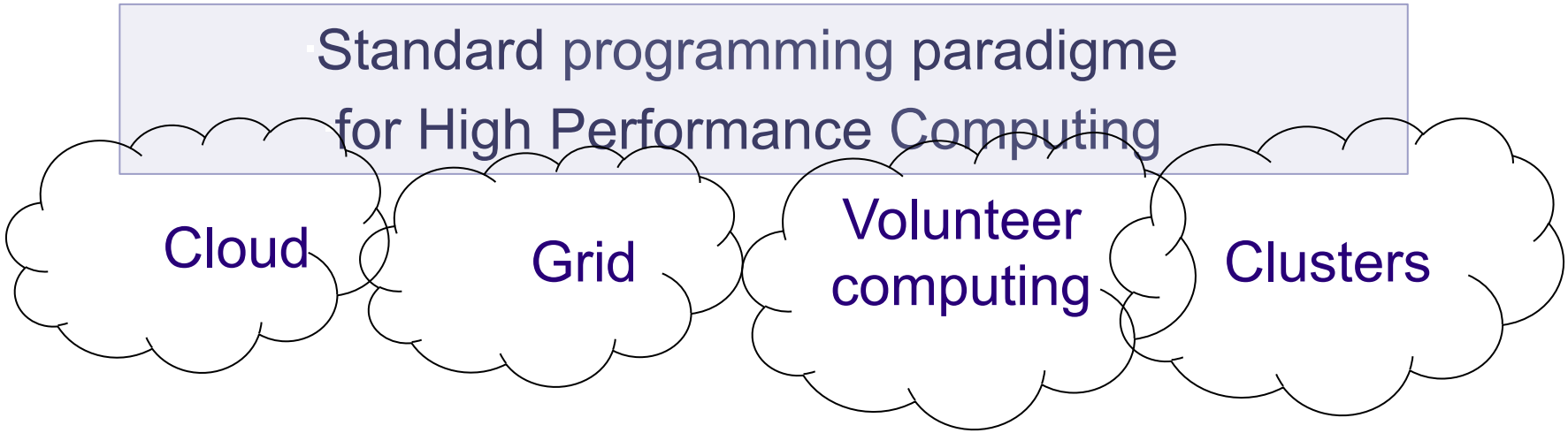


Outline

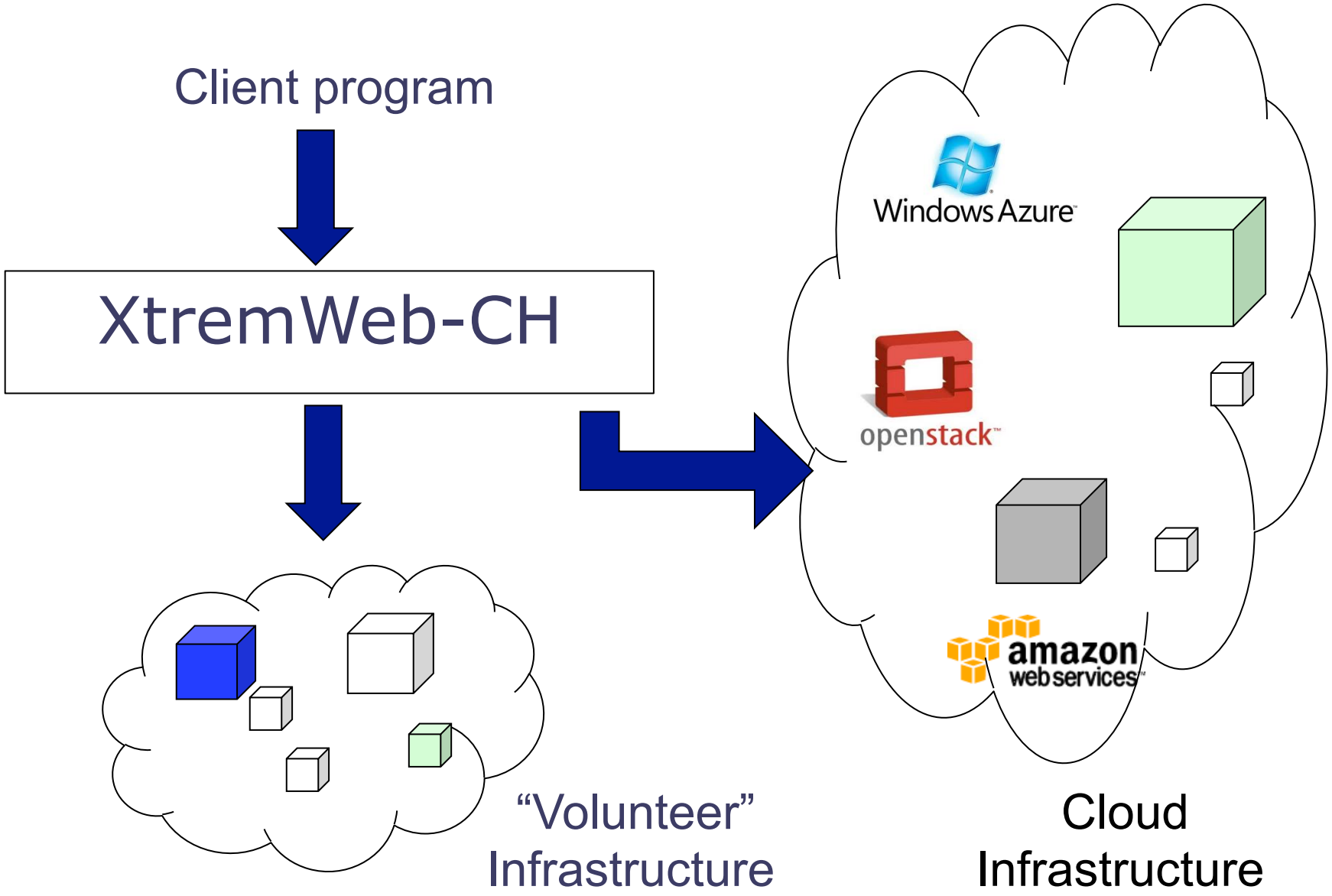
- Cloud vs. Grid & Volunteer Computing
- XtremWeb-CH (XWCH)
- Combining XtremWeb-CH with Cloud
- Conclusion

Why Combining Grid and Cloud?

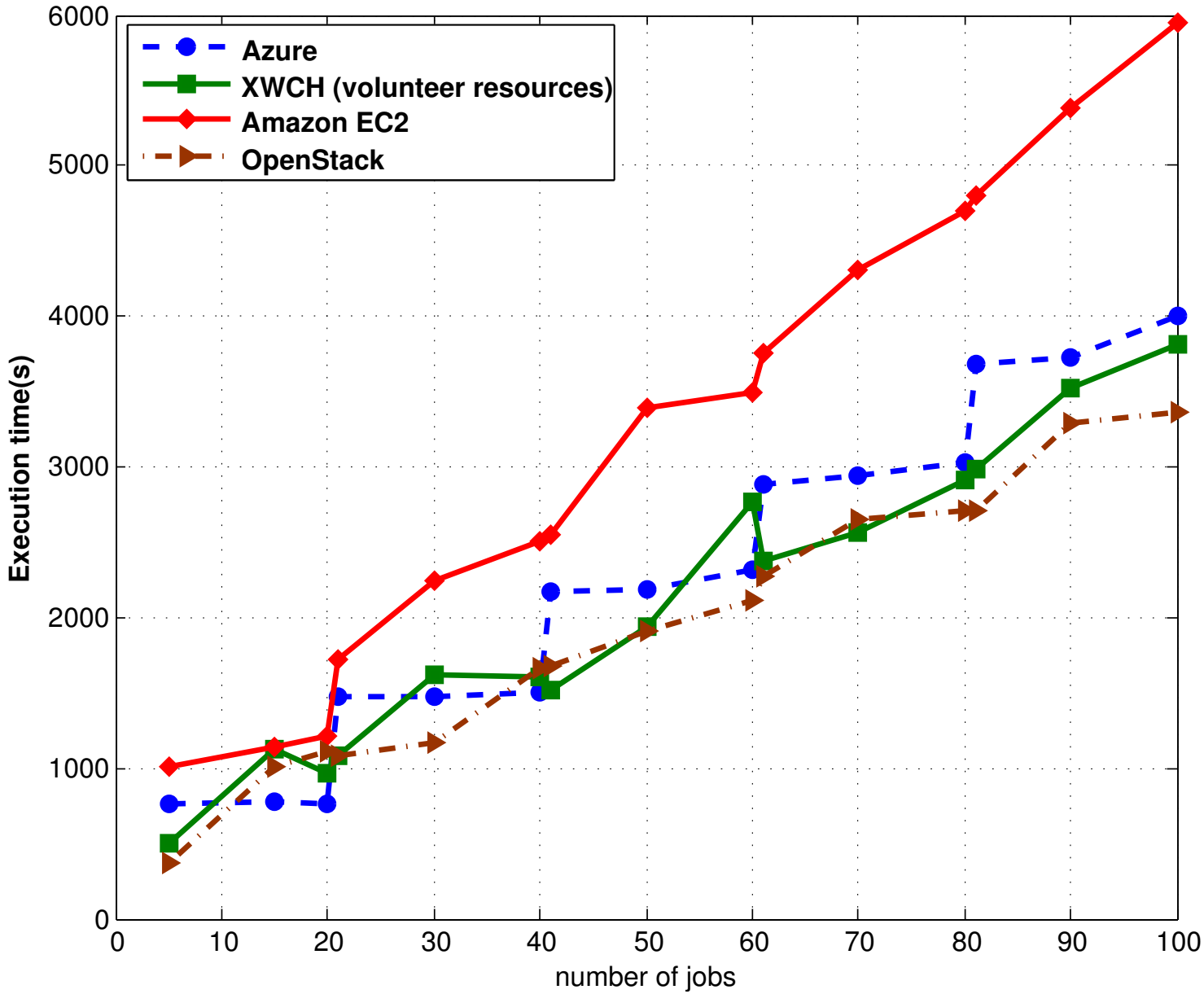
- Combine Cloud and Grid in order to:
  - enable a stable HPC platform where some of the nodes are governed by our universities, some by cloud infrastructure
  - take advantage of the HPC environments supported by Grid and VC middleware



Conclusion


Client program
XtremWeb-CH
Windows Azure
openstack
amazon web services
"Volunteer" Infrastructure
Cloud Infrastructure

Conclusion


Azure
XWCH (volunteer resources)
Amazon EC2
OpenStack
Execution time(s)
number of jobs
0
1000
2000
3000
4000
5000
6000
0
10
20
30
40
50
60
70
80
90
100