Programming distributed medical applications with XWCH2

Paper presentation for HealthGrid 2010.

June 30th 2010

Mohamed Ben Belgacem,
University of Geneva

UNIVERSITÉ DE GENÈVE

L'avenir est à créer

h e p i a

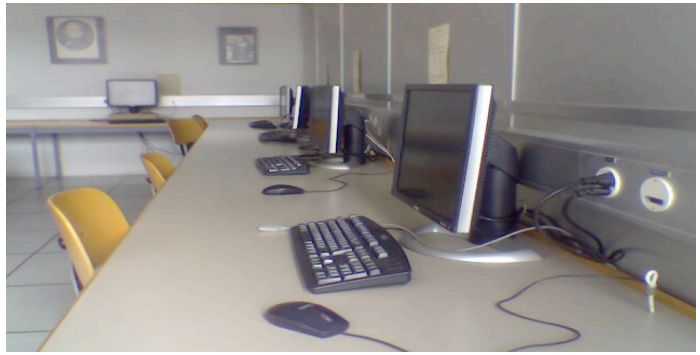
Haute école du paysage, d'ingénierie et d'architecture de Genève

Outline

- Introduction to XtremWeb-CH (XWCH)
- Architecture of XWCH2
- How to interface and Gridify?
- Applications
- Measurements

XtremWeb-CH

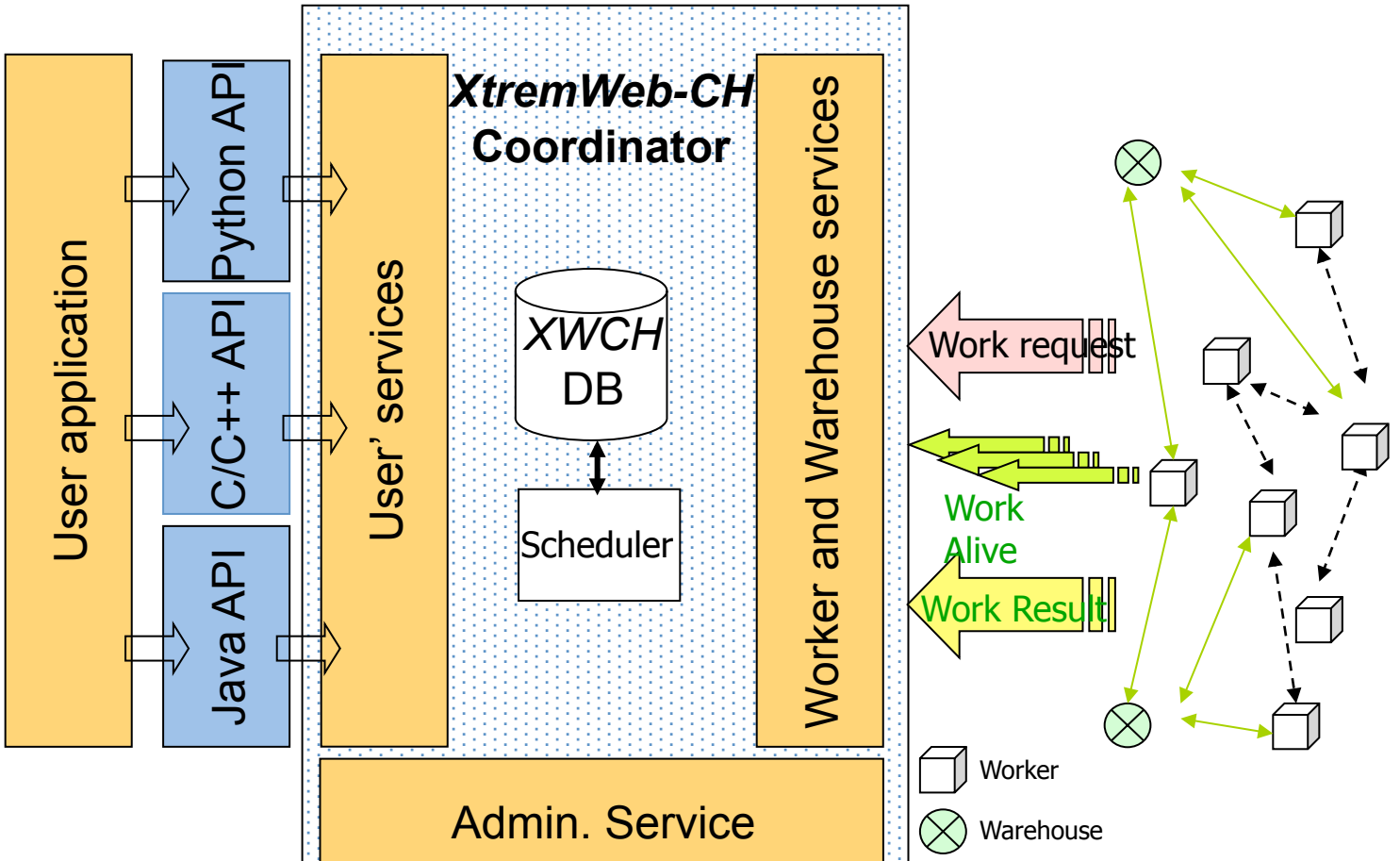
http://www.xtremwebch.net



History

- A volunteer computing and P2P computing inspired platform.
- 2003: Developed at “hepia”, originally based on ideas of the *XtremWeb* project (www.xtremweb.net).
- 2008: development of XWCH2 from scratch
- The software architecture was completely redesigned
- Major improvements have been brought to XWCH2 in order to obtain a reliable and efficient system:
  - Dynamic task generation
  - Flexible data sharing (data replication)
  - Easy to install, maintain and use.

XWCH architecture



Interfacing XWCH

API's & bridges

Warehouses

Coordinator

Workers

XWCH has

- API's for Java, C/C++
- a CLI
- a bridge for NorduGrid ARC

Interfacing XWCH: a simple job

Modify MinimalDemo.java

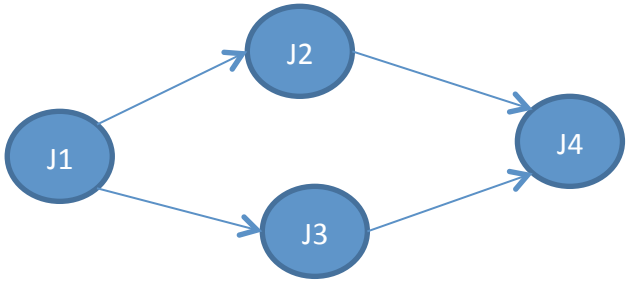
```
// Initialisation of the connection
c = new XWCHClient(ServerAddress, ".", IdClient);
c.init();
String appid = c.AddApplication("Hello World application");
String ModuleId1 = c.AddModule("Module1");
String refWind = c.AddModuleApplication (ModuleId1,
                                                    BinaryPath_Module1_win,
                                          PlateformEnumType.WINDOWS);
Fireref frefin=c.AddData("Input.zip");

String job0 = c.AddJob("First Job", //Job description
                       appid, //Application ID
                       ModuleId1, // Module identifier
                       CmdLine_for_job0, //Command line
                       frefin.toJobReference(),
                       list_files_out_job0,// file to
                       file_out_id_job0,// output file name
                       "SAME_JOB:job_id, REPLICATION:2"
                );
c.getJobResult(job0,…);
...
```

Interfacing XWCH: a simple job

Compile and run

```
$ javac -classpath .:XWCHClientAPI_V2.0.jar MinimalDemo.java
$ java -classpath .:XWCHClientAPI_V2.0.jar MinimalDemo
application added
module added
local files written
local files added 71a6a05b-b560-429a-95e3-
e90b9e919875:93a9be3644101e793ef285295fdfa297
Submitting job
waiting for job
9e6172db-d778-429a-ba39-0cb385b75bc7 WAITING
9e6172db-d778-429a-ba39-0cb385b75bc7 READY
9e6172db-d778-429a-ba39-0cb385b75bc7 COMPLETE
done, getting results fb703148-b9ef-4ccb-917c-eb34525699e7
EndApplication(): 68772b7c-f719-427c-b4be-7f850c09acaa --> [OK]
```

Interfacing XWCH: a simple DAG



NB: XWCH supports many sophisticated options, like DAG jobs, configurable replication, different executables for different platforms. All these are just default values here.

```
XWCHClient c =new XWCHClient (server, clientId);
String  J1,J2,J3,J4;
String appid=c.AddApplication("MyApp");
[…]
J1=c.AddJob (appid,inputfile1,moduleid,command1, "out",…);
While ( c.getJobStatus (J1) ≠ COMPLETED ) { ; }

While (int i=2;i<=3;i++){
        Ji= c.AddJob(appid,…);
}
While ( c.getJobStatus(J2) ≠ COMPLETED  &&  c.getJobStatus(J 3) ≠ COMPLETED ) { ; }
J4=c.AddJob (appid,inputfile4,moduleid,command4, "fout",…);
c.GetJobResult(J4,…);
c.EndApplication(appid);
```

Interfacing XWCH: other ways

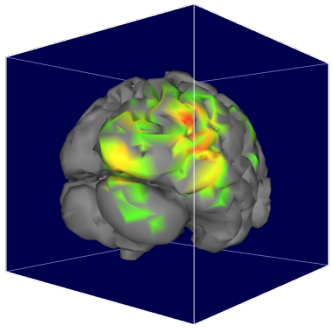
CLI:
$> /opt/xwch/xwch -f myjob.xwjdl

myjob.xwjdl:

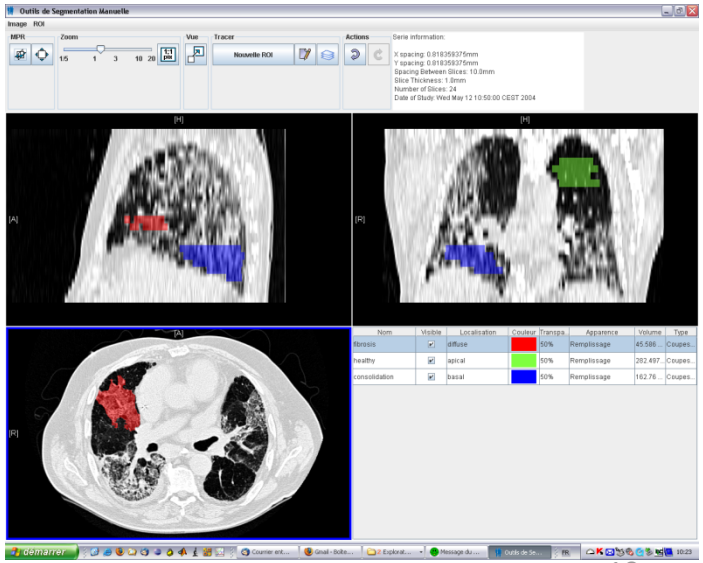
```
type=XWCHJob
xwserver=http://xtremwebch.hesge.ch:8080
client_ID =yourid
applicationname =APP_CLI_TEST
modulename =MODULE_CLI_TEST
datainputfilename =data.zip
binaryinputfilename =binary.zip
downloadresult=1
requirements=(os=LINUX || os=WINDOWS) && (freememory=512)
```

Applications

•*NeuroWeb* : build neuronal maps extracted from brain measurements



•*MedGift : medical image analysis and retrieval application*



NeuroWeb

- Objective: Reconstruction of the electromagnetic brain map

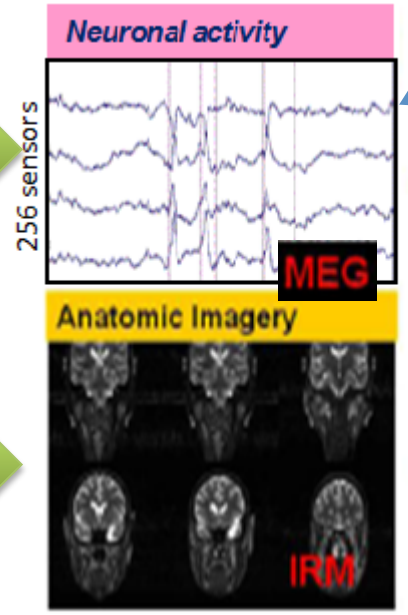
•A large scale optimization problem

# of sensors: 256 (dt = 1 ms)

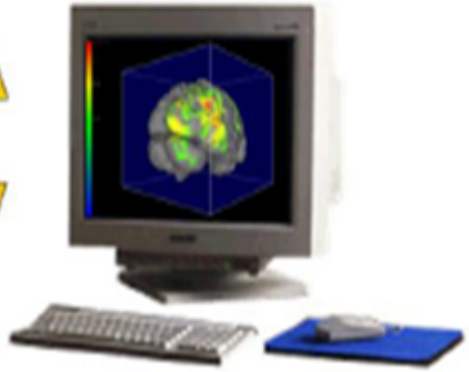
# of neurons: 60'000

MEG Scanner

MRI Scanner



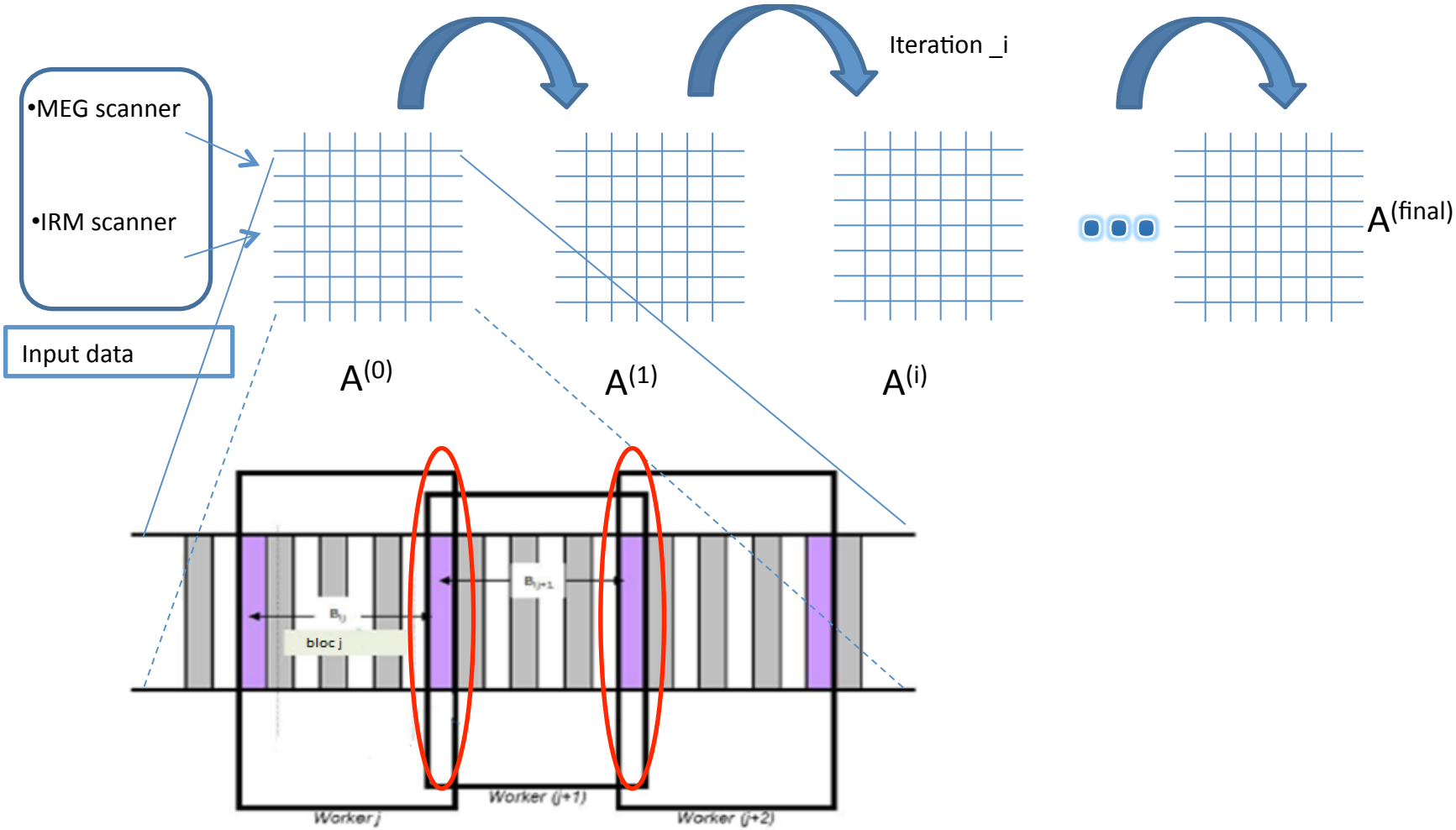
provides temporal information (functional data)



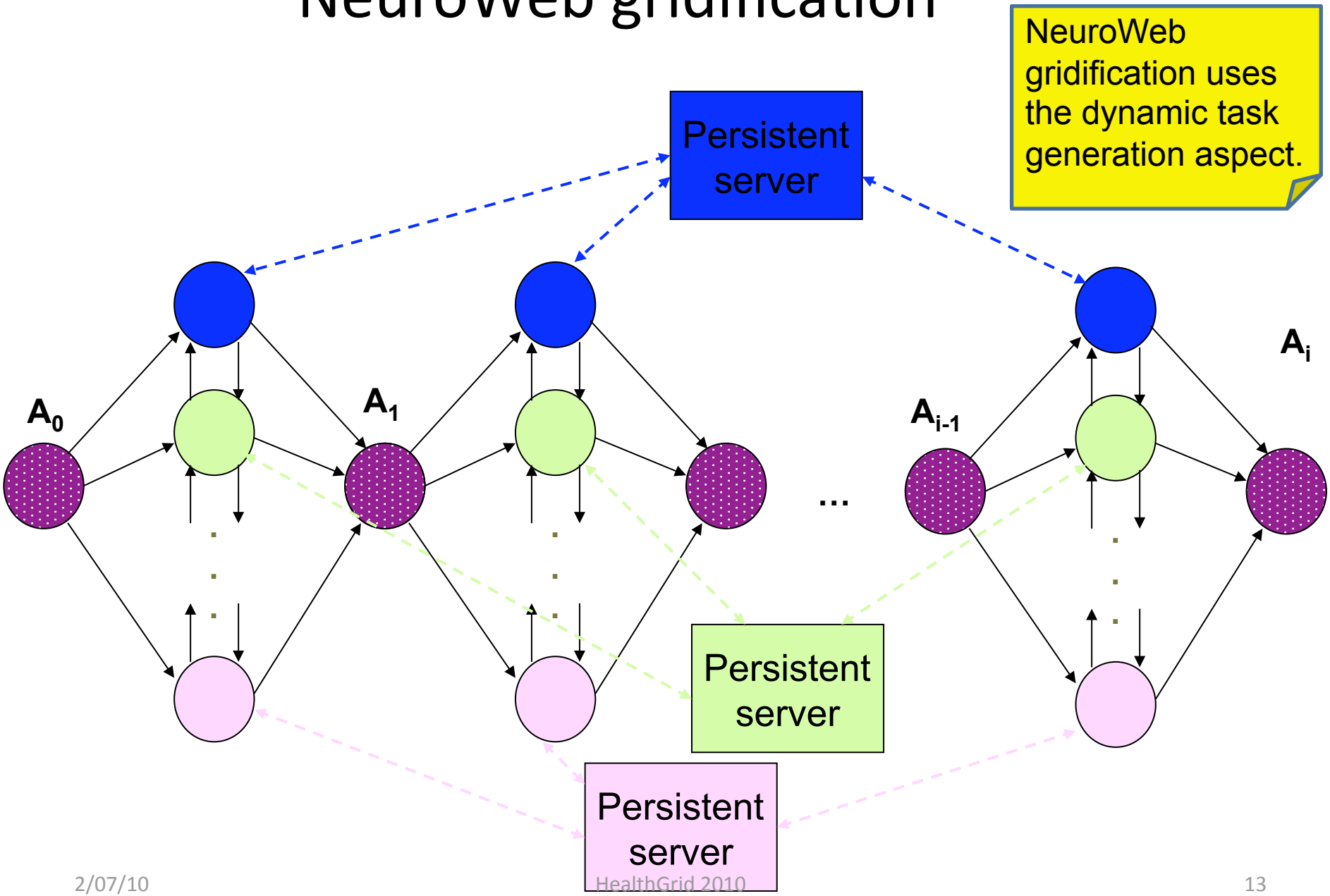
NeuroWeb

Brain activity is modeled by a matrix:

- Row: the “activity” of 1 neuronal region
- Column: the activity of all regions at a given time



NeuroWeb gridification



Applications

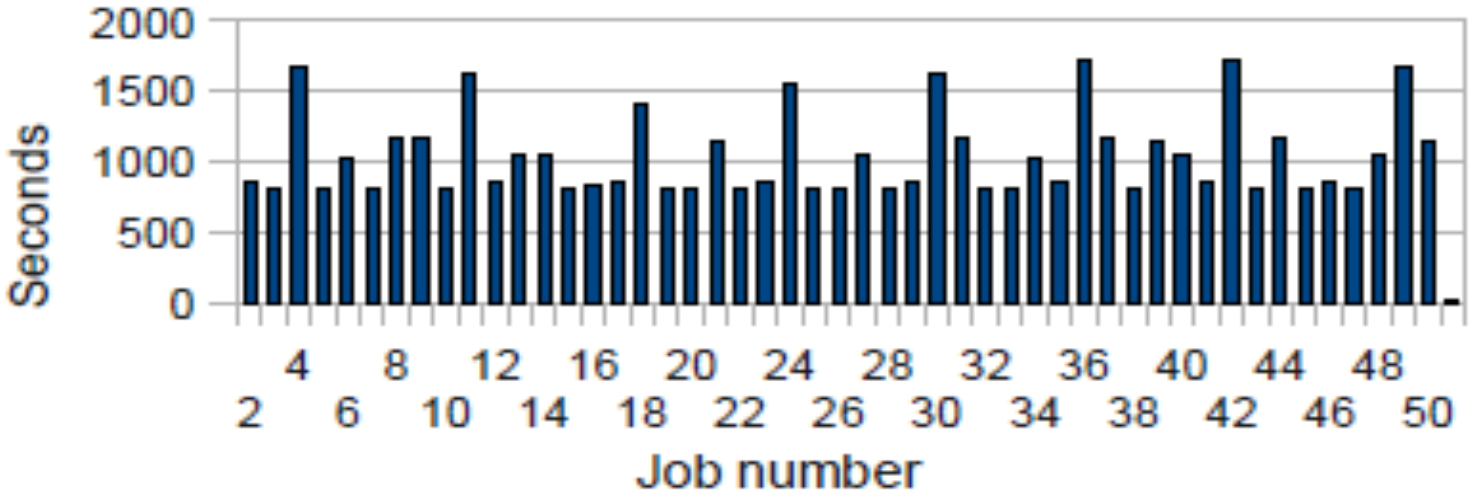
MedGift (Gnu Image Finding Tool)

- Content-based image retrieval is used as a diagnostic aid in hospitals
- *Gift* is a content-based image indexing and retrieval software developed by UniGe in the late 1990's
- *Gift* utilizes techniques from collection of colour and texture features
- *Gift* extracts these features and stores them in an inverted file

MedGift gridification

MedGift gridification uses the XWCH Command Line Interface (XWCH_CLI).

- Embarrassingly parallel jobs
- No communication between jobs
- Used data : 50 000 images, originally from radiological journals
- Gridified using the XWCH_CLI.
- Results (4 hours 53 minutes) are comparable to those achieved by the ARC Grid middleware (4 hours 25 minutes)

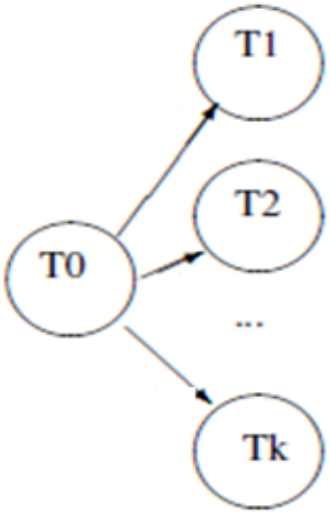


Measurements

Replication impact

A fork-join like application is used to show the impact of the data replication on the platform:

- 150 parallels tasks with different size for T0 output result.
- For each output size:
  - 1st execution with 1 replication
  - 2nd execution with 9 replications



Summary and future work

•XWCH2 benefits: easy to setup, firewall friendly, easy to program.

• Improve the scheduler response time

• Provide more simple programming modal

•"Decent" performance (upcoming paper, we compare with Condor).

•"Grid friendly": ARC bridge exists, but does not support Runtime Environments (will require lots of time to implement).