Ibis as Glue

Hands-on

Jason Maassen

Niels Drost

Computer Systems Group

Department of Computer Science

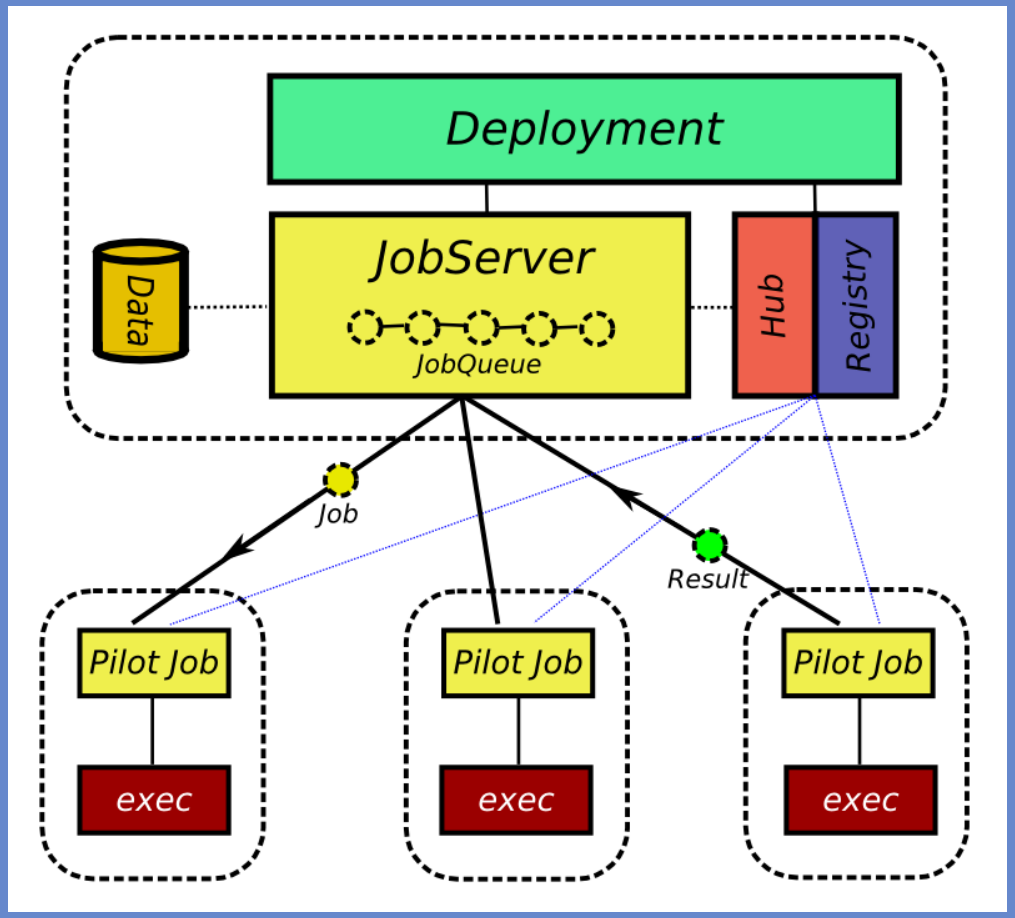
VU University, Amsterdam, The Netherlands

Ibis as Glue

- Use the IPL to communicate between parts of a computation
- In this Hands-on we will use the Pilot Job framework to illustrate this, and use IbisDeploy to start it

Pilot Job Framework

Design

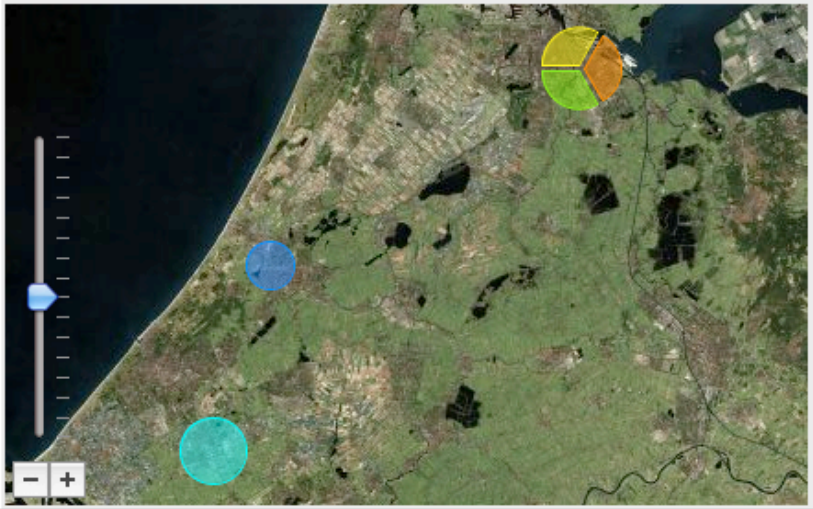


Ibis Deploy - demo-workspace

File Options Help

Experiments | Applications | Clusters | Network Overlay | Connection Overview | 3D Visualization

Experiment Editor



Pool Name: demo

Select Application

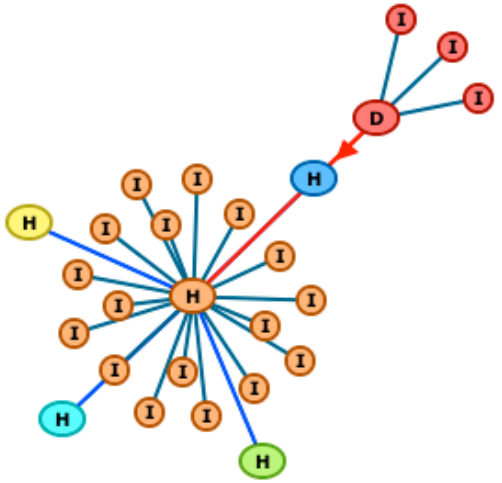
Server x 1

Select Cluster

VU x 1

Create | Create & Start

Experiment Monitor



	pool	name	job status	hub status	cluster	middleware	application	process count	resource count	output
☐	demo	Client	DEPLOYED	DEPLOYED	local	local	Client	1	1	output
☐	demo	Delft-Server	INITIALIZING	DEPLOYED	Delft	ssh	Server	16	16	output
☐	demo	Leiden-Server	INITIALIZING	DEPLOYED	Leiden	ssh	Server	16	16	output
☐	demo	Local-Server	DEPLOYED	DEPLOYED	local	local	Server	1	1	output
☐	demo	Multimedia...	INITIALIZING	DEPLOYED	MultimediaN	ssh	Server	16	16	output
☐	demo	UvA-Server	INITIALIZING	DEPLOYED	UvA	ssh	Server	16	16	output
☐	demo	VU-Server	DEPLOYED	DEPLOYED	VU	ssh	Server	16	16	output

Start All | Stop All | Remove All

Start Selected | Stop Selected | Remove Selected

Property files

- Grid file: description of all resources (clusters)
  - Hostname of frontend, JavaGAT adaptor used, Geolocation, etc
- Application file: description of applications
  - Main class, jars needed, arguments, etc
- Experiment file: description of experiment
  - “run application A on 32 nodes of cluster B”
  - Alternative: specify in GUI
- Workspace: directory containing all 3 files

Application File

```
# Application "JobServer"
JobServer.main.class = glue.multicore.JobServer
JobServer.libs = lib, deploy/lib-server
JobServer.arguments = --script blur.sh --input images
   --output output-images
JobServer.input.files = images, scripts/blur.sh
JobServer.output.files = output-image

# Application "Pilot"
Pilot.main.class = glue.multicore.PilotJob
Pilot.libs = lib, deploy/lib-server
```

Grid File

```
# Details of cluster "DAS4-VU"
#DAS4-VU.support.adaptor =
DAS4-VU.support.uri = ssh://fs0.das4.cs.vu.nl
#DAS4-VU.job.adaptor =
DAS4-VU.job.uri = ssh://fs0.das4.cs.vu.nl
DAS4-VU.file.adaptors = SftpTrilead, SshTrilead, Local
DAS4-VU.java.path = /usr/lib/jvm/java-openjdk/bin/java
DAS4-VU.job.wrapper.script = das4.script
#DAS4-VU.user.name =
#DAS4-VU.user.key =
#DAS4-VU.support.system.properties =
DAS4-VU.latitude = 52.3328
DAS4-VU.longitude = 4.8669
DAS4-VU.color = #FFBF80
```

Experiment File

```
# Details of job "job-00"
job-00.application.name = JobServer
job-00.process.count = 1
job-00.cluster.name = local
job-00.resource.count = 1
#job-00.runtime =
job-00.pool.name = pilotjob
#job-00.pool.size =

# Details of job "job-01"
job-01.application.name = Pilot
job-01.process.count = 1
job-01.cluster.name = local
job-01.resource.count = 1
#job-01.runtime =
job-01.pool.name = pilotjob
#job-01.pool.size =
```

Running the Example

- Download ibis-glue.zip from tutorial site
- Compile with ant
- Run
  - scripts/ibis-deploy-gui for GUI
  - scripts/ibis-deploy-cli for command line interface
- Use “-m” option for performance visualization in GUI