HPCKP'14

AJO

Asynchronous Job Operator

Gabriel Verdejo Álvarez – RDlab
Fernando Galindo Pascual - RDlab

HPC Knowledge Portal
www.hpckp.org

CONTENTS

About Us

- RDlab: Research and Development Laboratory.
- Belongs to LSI department at UPC University.
- Created on fall 2010.
- Working with HPC since 2004.
- Gabriel Verdejo (gabriel@lsi.upc.edu) – IT manager
- Fernando Galindo (fgalindo@lsi.upc.edu) – IT Research Support

Why AJO?

- **Motivation**

  - Several groups use HPC system for research.

  - One group asked for a transparent way to access HPC system.

    European Project: Several Universities.

    Connect Web Service to HPC system for translations.

  - We have good experience due to S-GAE software.

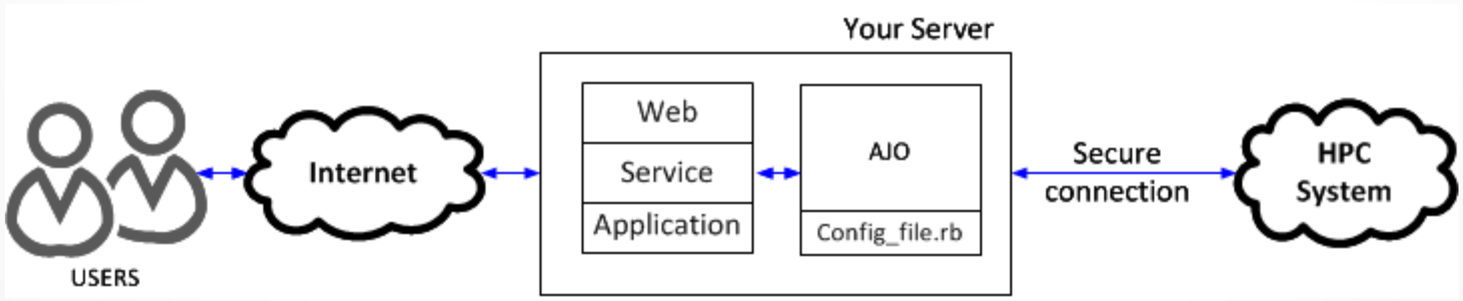
  - We are compromised with GNU software development.

The AJO Application: What is AJO?

- Transparent gateway between services and HPC systems.
- Users connect to the service, not the HPC system

  HPC as a service (AAS)

- http://rdlab.lsi.upc.edu/ajo



Who can benefit from AJO?

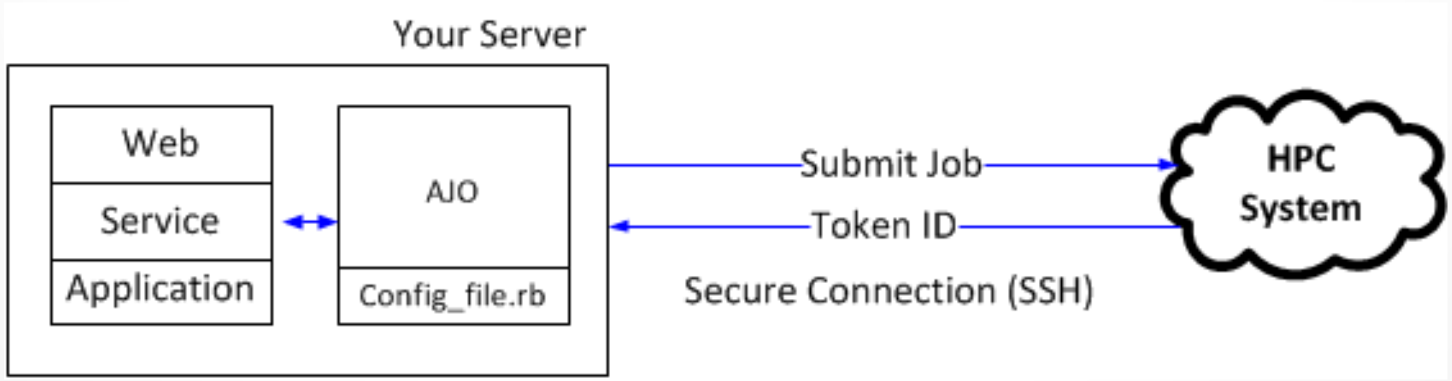
- HPC System based on DRMAA API.
- Any application that needs access to HPC system.
    - Web applications / services
    - Embedded software
    - Programmed in any language!
- Batch access, non-interactive.
    - Move local executions to HPC system
    - Scalable

The AJO Main Features

- Users connect to frontend, not the HPC system.
  - Do not even need to know there is an HPC system behind.
- Allows users to submit through the same username
  - No need to register different users on the HPC system.
- Unix Compliant
- Issues direct SSH commands to HPC system.
- Password-less SSH through private/public keys.
- **Functionalities**:
  - Submit, Query, Retrieve, List, Cancel, Erase, Help.

AJO functionalities: Submit

- Read configuration file
  - Get Files and folders to move to the HPC system
  - Get command to execute on HPC system
- Send job to HPC system
  - Sending is synchronous
- Generates and returns Token ID



AJO functionalities: The Token ID

- **Why?** Because…

  … execution is asynchronous.

  We need a way to query the job status later.

  … many jobs can be sent simultaneously.

  We need a way to identify jobs uniquely: token ID is unique.

- **What?**

  Alphanumeric sequence

- Securely generated and encrypted.

  Generated Ad-hoc on every submit.

  HPC system isolation: Token ID ≠ Job ID

  Encryption: Job ID can not be extracted from Token ID

AJO functionalities: Query

- Check the status of an already sent job.

  Is it finished?

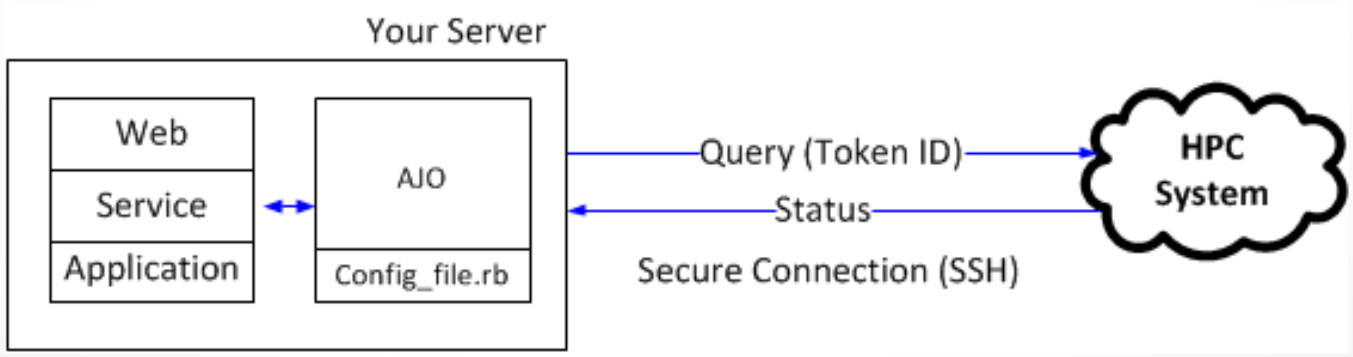
- Uses the token ID generated before.

- AJO is stateless

  Job execution is asynchronous

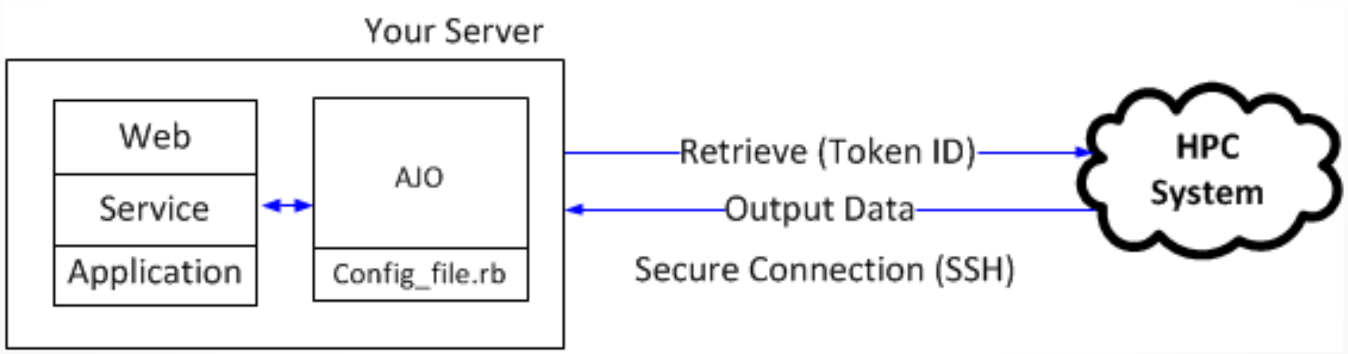
  Ajo does not "remember" which jobs has it sent.

  The user queries to know the status of the job.



AJO functionalities: Retrieve

- Opposite to submit:
  - Check if the job is finished
  - If it is, read configuration file to see what to retrieve
  - Pack data to retrieve and save it at the retrieval directory



AJO functionalities: Other

- **LIST**

  Show list of job identifiers submitted through AJO.
  Necessary due to the asynchronous nature of the system.
  Allows to know how many jobs are queued on the HPC system for AJO.

- **CANCEL**

  Cancel running job submitted through AJO.

- **ERASE**

  Erase information at server of job submitted through AJO.

- **HELP**

  Display help.

Example of Usage

- Written in Ruby, executed through CLI

  Can be called from any UNIX compatible platform

  UNIX-like parameter style: ajo –c <config_file> <options>

- **Submit**

  ```
  ajo –c config.rb  -s
  ```

- **Query**

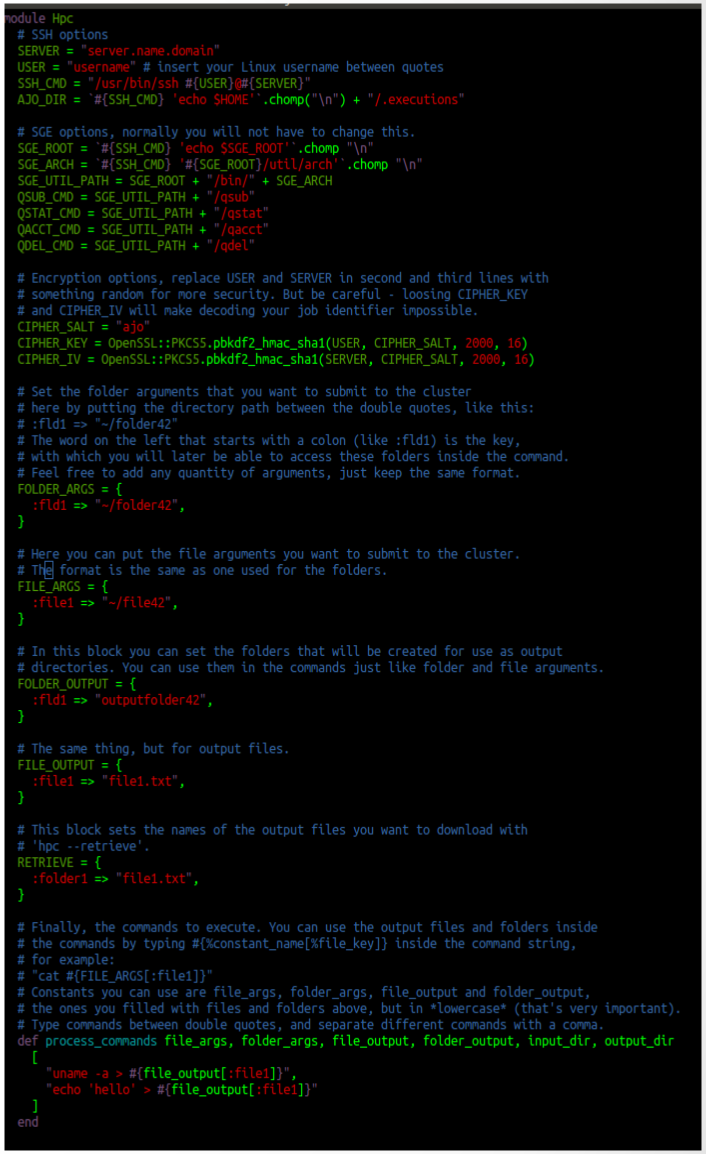
  ```
  ajo –c config.rb  -q token_id
  ```

- **Retrieve**

  ```
  ajo –c config.rb  -r token_id
  ```

The configuration file

- **Format**
  - Token = value
- **General Options**
  - Server name, username,…
- **Grid Engine options**
  - Binaries location.
- **Folder and File arguments**
  - FOLDER ARGS, FILE ARGS
  - FOLDER OUTPUT, FILE OUTPUT, RETRIEVE
- **Command processing block**
  - Commands to execute



```
module Hpc
  # SSH options
  SERVER = "server.name.domain"
  USER = "username" # insert your Linux username between quotes
  SSH_CMD = "/usr/bin/ssh #{USER}@#{SERVER}"
  AJO_DIR = `#{SSH_CMD} 'echo $HOME'`.chomp("\n") + "/.executions"

  # SGE options, normally you will not have to change this.
  SGE_ROOT = `#{SSH_CMD} 'echo $SGE_ROOT'`.chomp "\n"
  SGE_ARCH = `#{SSH_CMD} '#{SGE_ROOT}/util/arch'`.chomp "\n"
  SGE_UTIL_PATH = SGE_ROOT + "/bin/" + SGE_ARCH
  QSUB_CMD = SGE_UTIL_PATH + "/qsub"
  QSTAT_CMD = SGE_UTIL_PATH + "/qstat"
  QACCT_CMD = SGE_UTIL_PATH + "/qacct"
  QDEL_CMD = SGE_UTIL_PATH + "/qdel"

  # Encryption options, replace USER and SERVER in second and third lines with
  # something random for more security. But be careful - loosing CIPHER_KEY
  # and CIPHER_IV will make decoding your job identifier impossible.
  CIPHER_SALT = "ajo"
  CIPHER_KEY = OpenSSL::PKCS5.pbkdf2_hmac_sha1(USER, CIPHER_SALT, 2000, 16)
  CIPHER_IV = OpenSSL::PKCS5.pbkdf2_hmac_sha1(SERVER, CIPHER_SALT, 2000, 16)

  # Set the folder arguments that you want to submit to the cluster
  # here by putting the directory path between the double quotes, like this:
  # :fld1 => "~/folder42"
  # The word on the left that starts with a colon (like :fld1) is the key,
  # with which you will later be able to access these folders inside the command.
  # Feel free to add any quantity of arguments, just keep the same format.
  FOLDER_ARGS = {
    :fld1 => "~/folder42",
  }

  # Here you can put the file arguments you want to submit to the cluster.
  # The format is the same as one used for the folders.
  FILE_ARGS = {
    :file1 => "~/file42",
  }

  # In this block you can set the folders that will be created for use as output
  # directories. You can use them in the commands just like folder and file arguments.
  FOLDER_OUTPUT = {
    :fld1 => "outputfolder42",
  }

  # The same thing, but for output files.
  FILE_OUTPUT = {
    :file1 => "file1.txt",
  }

  # This block sets the names of the output files you want to download with
  # 'hpc --retrieve'.
  RETRIEVE = {
    :folder1 => "file1.txt",
  }

  # Finally, the commands to execute. You can use the output files and folders inside
  # the commands by typing #{%constant_name[%file_key]} inside the command string,
  # for example:
  # "cat #{FILE_ARGS[:file1]}"
  # Constants you can use are file_args, folder_args, file_output and folder_output,
  # the ones you filled with files and folders above, but in *lowercase* (that's very important).
  # Type commands between double quotes, and separate different commands with a comma.
  def process_commands file_args, folder_args, file_output, folder_output, input_dir, output_dir
    [
      "uname -a > #{file_output[:file1]}",
      "echo 'hello' > #{file_output[:file1]}"
    ]
  end
```

The configuration file (II)

- Example: Command processing block

```
def process_commands
          file_args, folder_args, folder_output, input_dir, output_dir
          [
                    "uname -a > #{file_output[:file1]}",
                     "echo 'hello' >> #{file_output[:file1]}"
          ]
end
```

Real Scenario

- TALP Research Center
- NLP group
- FAUST project

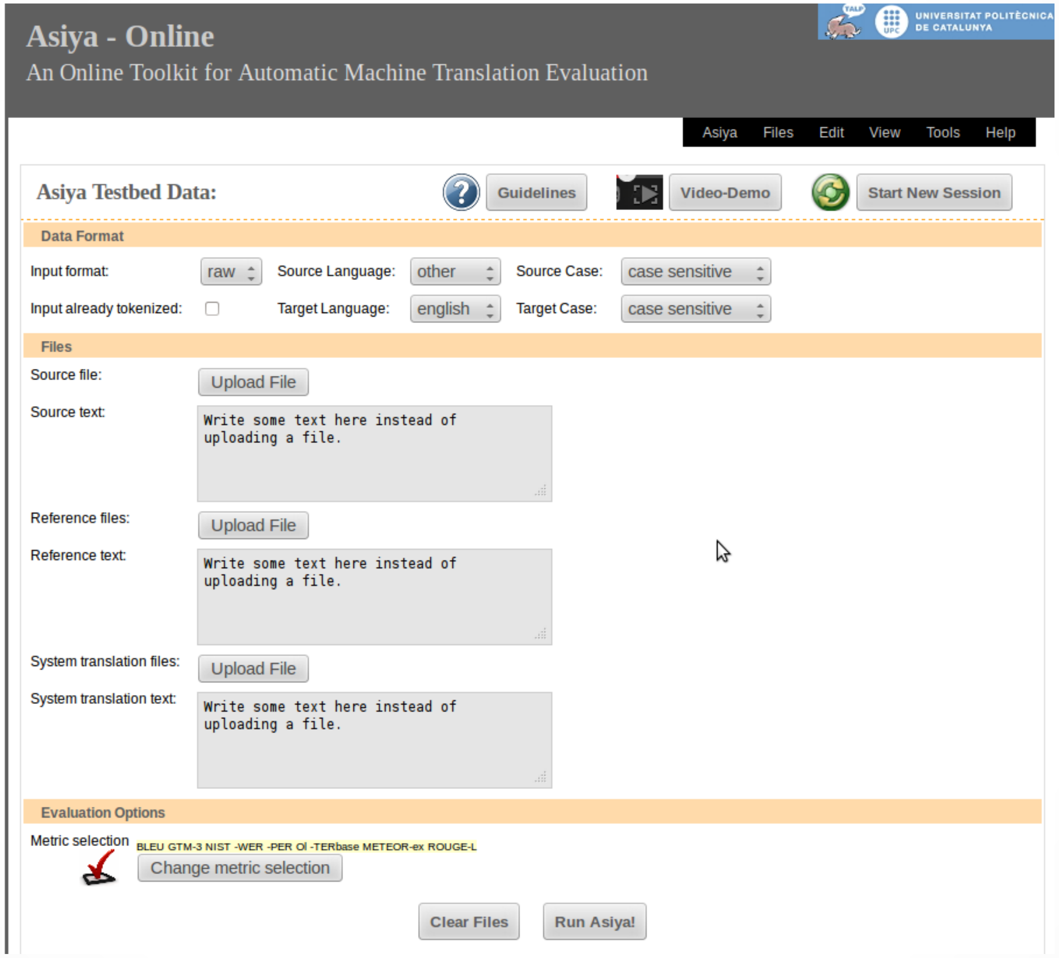
http://asiya-faust.lsi.upc.edu/

Asiya: a software for Automatic Machine Translation.

Asiya-Online: A web service that allows accessing the Asiya Framework for testing purposes, with no need to install packages.

Sends its processing to the HPC system through AJO.

Real Scenario (II)



Asiya - Online

An Online Toolkit for Automatic Machine Translation Evaluation

Asiya Testbed Data:

Data Format

Input format: raw
Source Language: other
Source Case: case sensitive
Input already tokenized: ☐
Target Language: english
Target Case: case sensitive

Files

Source file: Upload File

Source text: Write some text here instead of uploading a file.

Reference files: Upload File

Reference text: Write some text here instead of uploading a file.

System translation files: Upload File

System translation text: Write some text here instead of uploading a file.

Evaluation Options

Metric selection: BLEU GTM-3 NIST -WER -PER Ol -TERbase METEOR-ex ROUGE-L

HPCKP'14

AJO

Asynchronous Job Operator

http://rdlab.lsi.upc.edu/ajo