Science gateway feedback

Bruce Becker
Meraka Institute, CSIR

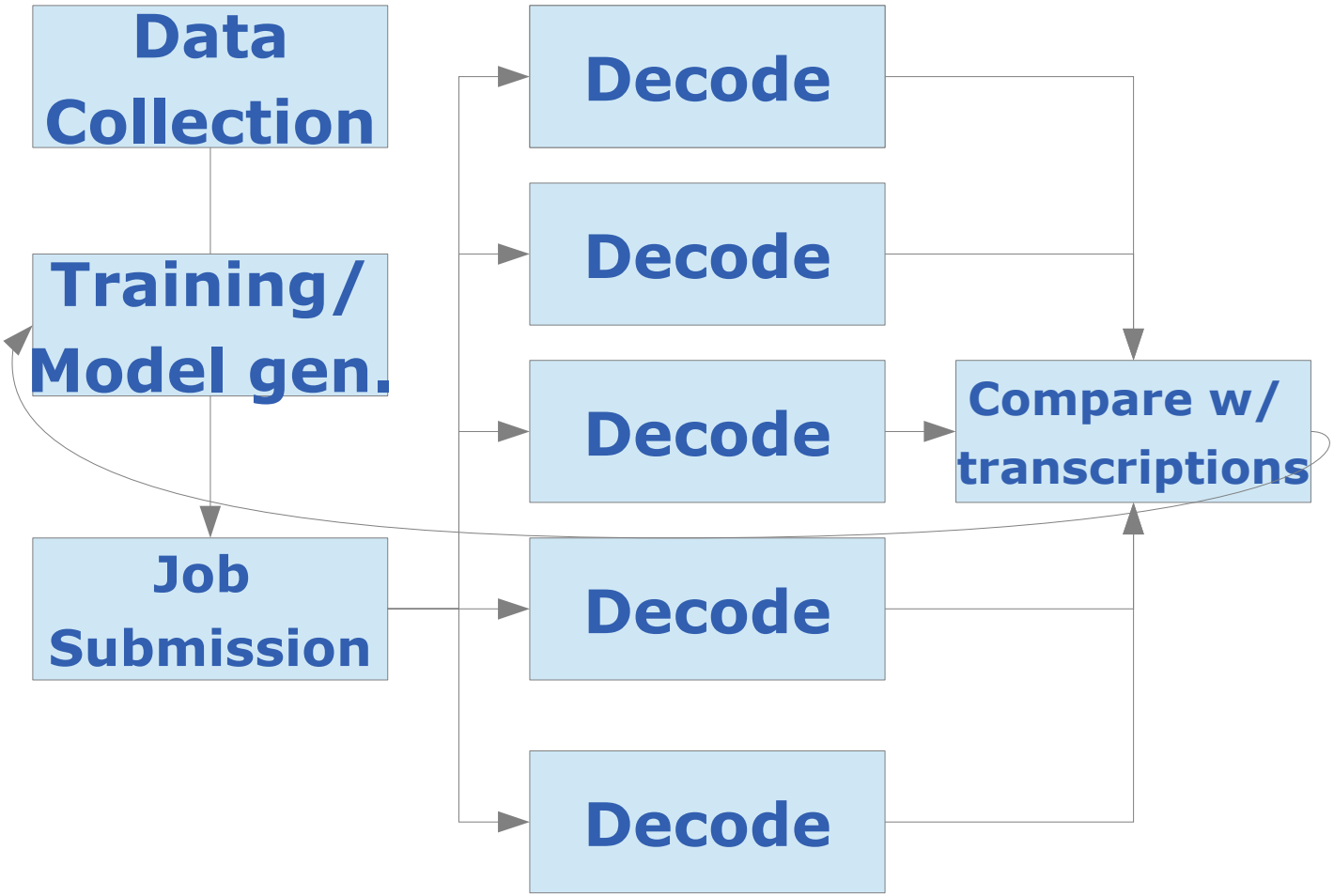
Grid School for porting to Science Gateways
UNAM, Mexico.

State of the application

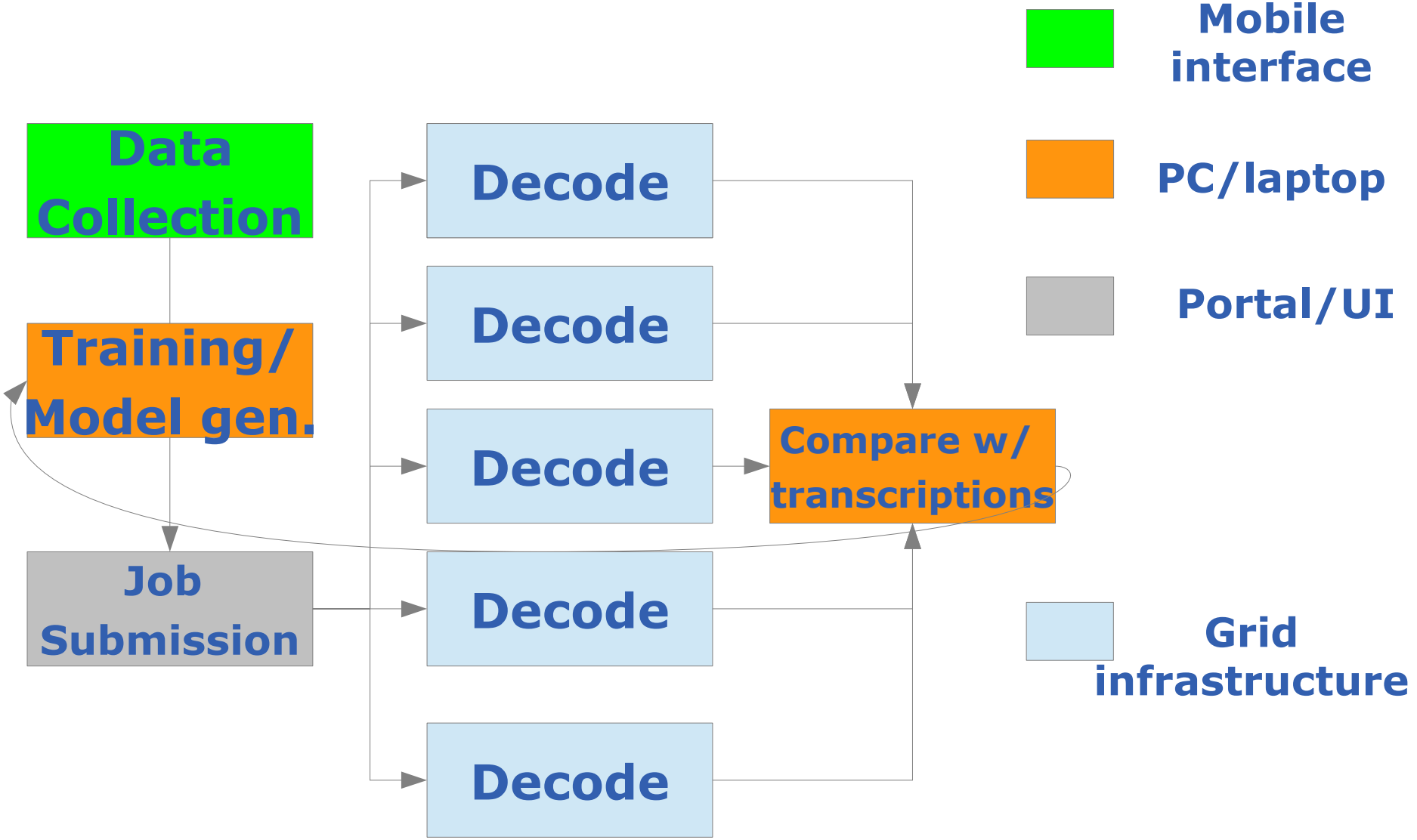
- **Hidden Markov Toolkit:**
  - Widely used in the Automatic Speech Recognition Domain
  - Open source toolkit, developed by University of Cambridge:
    - http://htk.eng.cam.ac.uk/download.shtml
  - Very few dependencies, can be compiled in few minutes with standard gcc-4.x compiler.
- **HTK in South Africa**
  - Application heavily used by Human Language Technologies group in Meraka Institute
  - Requested for porting to the grid in 2010, in use by researchers since then
  - Application deployed on all SAGrid sites, maintained by SAGrid Software Manager with tag : VO-sagrid-HTK-3.4.1 VO-sagrid-HTK

HTK use cases

- **1) Training models: ~ 1 day on a normal PC**
  - Provide input audio features and correct transcriptions to produce an accurate model of speech
  - Mutually exclusive speakers are required
  - Requires a special, restricted data set
- **2) Test performance on different speakers – ~ 1 day per speaker**
  - Decoding process (HTK workhorse) using model
  - Compare decoding results with transcriptions
- **3) Model perfection / iteration**
  - Repeat 1), 2) with different model parameters to find best theoretical response :
  - M*N days of CPU


Data Collection
Training/ Model gen.
Job Submission
Decode
Decode
Decode
Decode
Decode
Compare w/ transcriptions

Where is work being done ?



Goals of this event

- **Primary goals of this event :**
  - Create simple web interface to generic HTK usage
  - Test working example of HTK on a different infrastructure (Gisela)
  - Demonstrate multi-infrastructure usage of HTK
- **Secondary goals :**
  - ***Understand*** the Science Gateway concept
  - ***Study*** applicability in **general** to grid applications in South Africa
  - ***Propose*** model for extensive usage of application gateways in SAGrid
  - ***Provide*** feedback to GILDA development team on possible areas of improvement

Primary Goals

- **HTK Science Gateway Status :**
  - Modified the basic portlet (hostname) to submit HTK job with input parameter file to perform **decoding**
  - HTK compiled on the fly – no porting to grid necessary.
  - Input data retrieved at runtime from LFC
    - However, data is in SAGrid VO – permissions need to be changed - problem
    - Data could be retrieved from other places – http server, etc
- **HTK basic decoding portlet is working – VERY simple.**

Secondary goals : Understand

- **Science Gateway concept differs from “normal” grid UI in more than just the graphic user interface**
- **Auth/AuthZ:**
  - Single proxy used to interact with the grid services (application robot)
  - Robot must be added to the VO which is enabled on the infrastructure

OK !

  - Single users identified and authorised by internal db, IdF – need existing infrastructure for this
- **Functionality:** Gateway is developed with specific functions in mind, not general purpose

Secondary Goals : Study

- **Gateway Development process is more concrete: based on specific components:**
  - JSP/HTML for display
  - JSR specifications for interaction and workflow (ACTION_XX, VIEW_XX ; processAction, doView)
  - Java API to grid libraries
  - MySQL database for state tracking
- **What did we learn:**
  - Without a clear understanding of the desired ***functionality***, it is very difficult to develop a satisfactory portlet.
  - Portlets should be constrained to one specific function
  - A good understanding of ***each*** component is necessary for the developer to provide a satisfactory portlet
  - Non-trivial knowledge of non-standard code (GridEngine, helper classes) helps, but

Secondary Goal: Propose

- **In SAGrid, we have adopted a procedure development process to standardise common tasks across the infrastructure**
- **Propose to use this process to develop a common procedure for developing application gateways:**
  - Determine deployment mechanism – permanently installed and tagged on sites, or compile using glide-in, etc
  - Define functionality of the application – what atomic actions are undertaken by the researcher ?
  - Determine on which actions to delegate to the grid, and which to perform locally on the portal
  - Determine what the user needs to do and what can be automatically done by the portal (regular job submission, job retrieval, result visualisation, etc)
- **In summary : Use the gateway more as a Research Environment than a "pretty UI".**
- **Need to investigate more functionality of LifeRay and existing standard portlets**

Feedback to GILDA: Training Material

- **For someone coming from an "old" grid environment, development followed a simple cycle :**
  1) Determine application compilation configuration and dependencies
  2) Prepare deployment and validation script
  3) Deploy, tag sites
  4) (optional) Prepare demo JDL and script for user
  5) User is left with a lot of freedom
- **Suggestion1: spend more time on background knowledge requirements**
  - Provide "standard" liferay training on developing simple, nontrivial portlets, e.g.
    - "Hello world" with file upload, parameter input, feedback
    - Up-front description of user tracking database – fields, what they mean, etc
    - Simple example to exchange data between JSP/Java code/db

Feedback to GILDA: Code

- **The SAGA and GridEngine libraries are not documented, developer sees a “black box” - very hard to extend basic portlet functionality with confidence.**
- **Suggestion 2: documentation**
  - Provide a link on the GILDA wiki to automatically-generated code reference (using e.g. doxygen), with class inheritance and method descriptions
- **Suggestion 3: separate production and training repositories**
  - Starting with MultiInfrastructure portlet can be confusing and defeat the didactic aim of the school
  - StandAlone code already assumes integration into portlet... can be confusing at first sight.
  - Separate tutorial and examples into atomic steps
    - True standalone (only SAGA)
    - Portlet with single infrastructure code
    - Multi-infrastructure code

Benefit of Science Gateway

- **For someone coming from an "old" grid environment, the user had all the freedom -**
  - Free to write their own JDL and scripts
  - Can lead to:
    - Major disruption of infrastructure – flooding jobs, many waiting jobs, etc.
    - Bad user experience – incompatible requirements, overloaded services, bad resource selection, etc
- **With a gateway, the user can have restricted functionality**
  - Clearer understanding of what the grid can do, in terms of user-interaction
  - Gateway maintainer can tune the application submission parameters "properly"
- **Need to consider very carefully what to do on behalf of the user, and what to allow the user to do**

- **Finalise the Decoding Portlet**
  - Provide input fields for grammar and model upload at submit time
  - Implement simulation of parametrised job submission, using submission loop.
  - Provide hook for output files to be included in output sandbox
- **Implement other action portlets**
  - Model generation (quite easy)
  - Data collection (need audio stream interface)
- **Implement workflow in LifeRay, using standard portlets → create usable Virtual Research Environment**
  - Need to extend the user tracking database !
- **Understand better the OpenSocial implementation in LifeRay, to allow users to remain within the Gateway, while using other services to collaborate**

- **General issues:**
  - How to integrate with other general services ? GGUS tickets, trac tickets, etc.
  - Visualisation of results – standard java/jsp visualisation libraries would be nice...
- **Where is the “Grid Dashboard” portlet ?**
  - Users see almost no information on the state of the infrastructure...
  - While we understand the reasoning behind this, would like to be able to provide user with a “what-to-expect” portlet :
    - how many jobs are running on the grid
    - What services are up/down, etc
    - Consider interacting with GOCDB ?

End