Early Application Experience with Grid Application Toolkit (GAT)

- Stevens Le Blond
- Ana-Maria Oprescu
- Chen Zhang

Motivations

- A real-life application for the Almere Grid (City Grid)
- First step, deploy it for DAS2
- We could not manage with RMI
- Finally, discovered GAT as a possible solution
- Report on our experience with the GAT

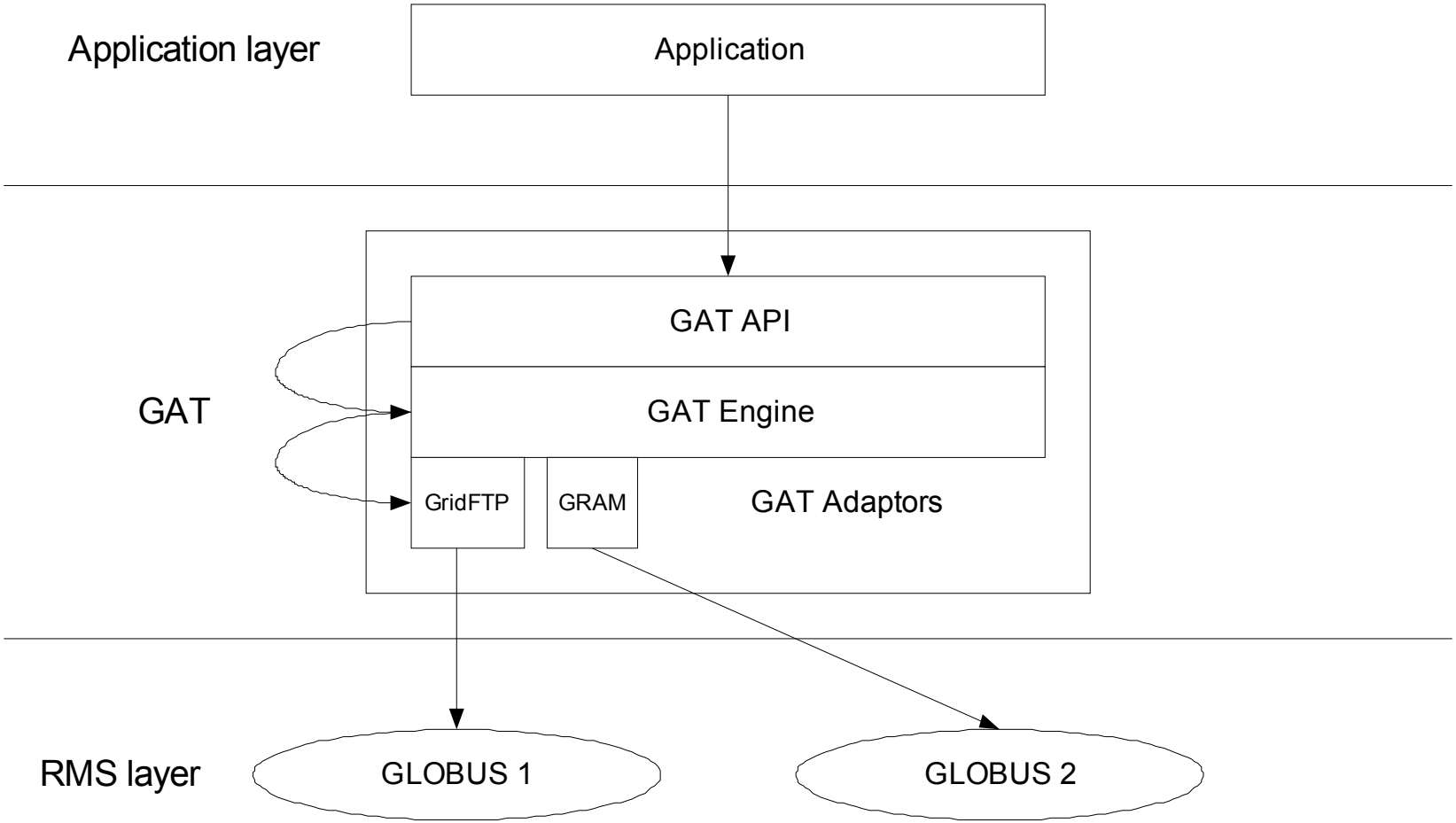
Overview

- A Parallel MPEG Encoder
- The (Java-)GAT
- Implementing the MPEG Encoder
- The Good, The Bad and The Ugly
- Conclusions and Future Work

A Parallel MPEG Encoder

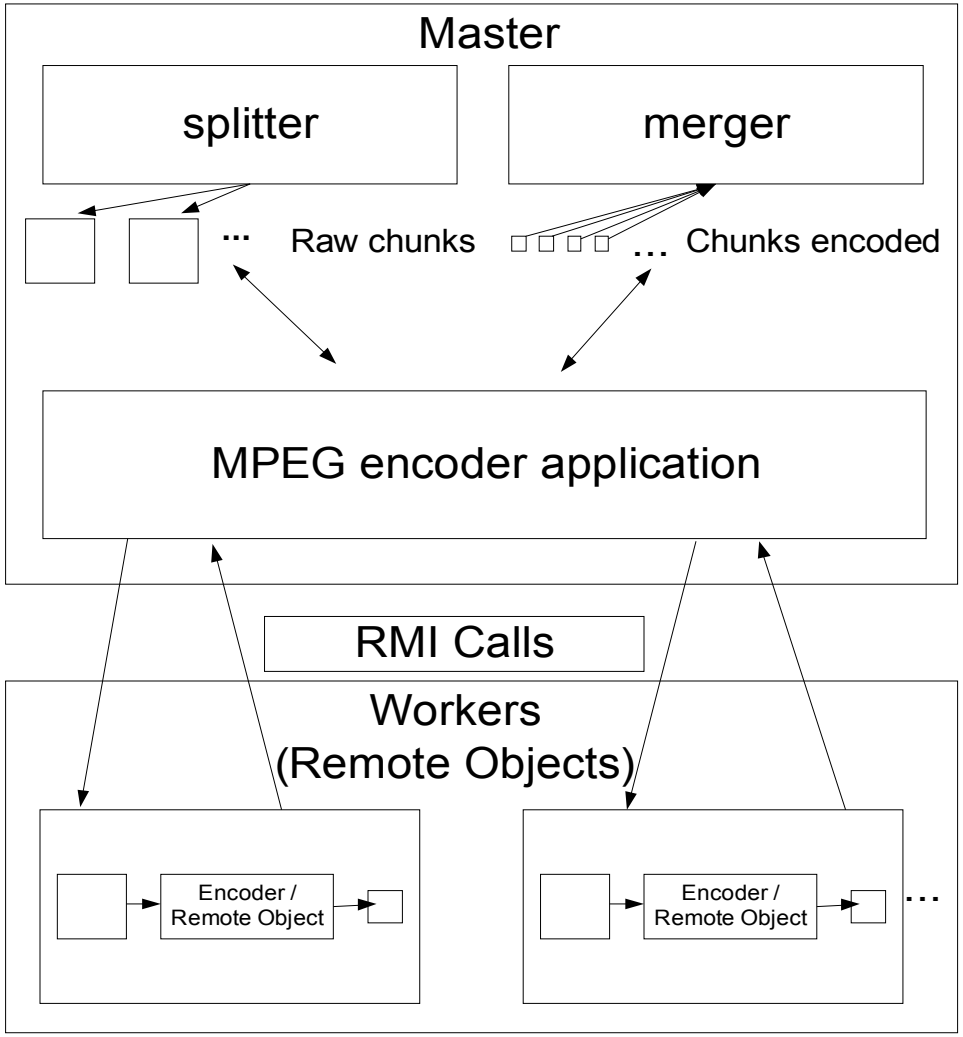
- Why an MPEG Encoder
  - From raw avi files to avi files
  - Smaller video files, same quality
  - Video/audio protocols
- Why parallelize it
  - Takes a lot less time (ex. hours vs. 10 min. for a 4GB file)

The (Java-)GAT

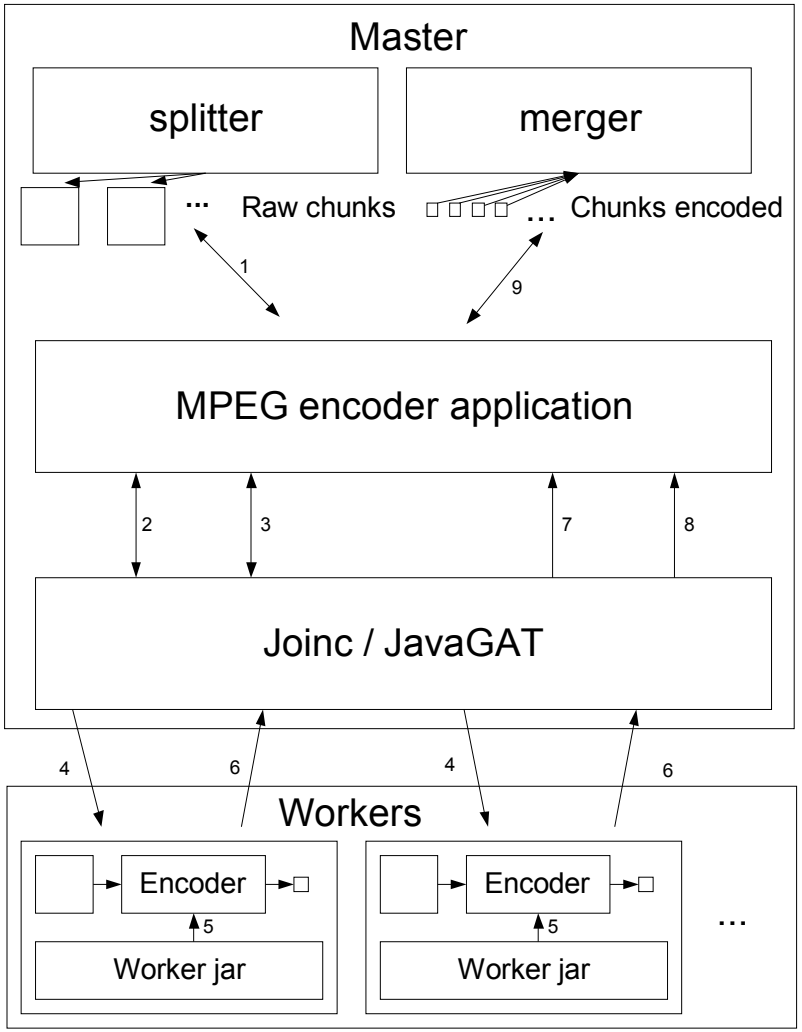

Application layer
Application
GAT API
GAT
GAT Engine
GridFTP
GRAM
GAT Adaptors
RMS layer
GLOBUS 1
GLOBUS 2

Implementing the MPEG Encoder

with RMI



with (Java-)GAT



The Good, The Bad and The Ugly

Experience with the GAT

- The good – what really works nicely
- The bad – what we feel is missing
- The ugly – what works, but could be done better

The Good

- Grid programming becomes easier
  - Files
  - Jobs
- Flexible
  - Provides reasonable default behavior
  - Allows selective control at different levels
- Transparent within external limitations
  - Location
  - OS

The Good (Files)

```
import org.gridlab.gat;
String          fileName  =  “file;
GATContext      context   =  new.GATContext ();
Preferences     prefs     =  new Preferences ();
//prefs.put(“FileAdaptor.type”, “ssh”);
URI             fileURI   =  new URI ( “any:///” +fileName );
File            file      =  GAT.createFile ( context, prefs, fileURI );
```

The Good (Jobs)

Job software description

```
// first, populate org.gridlab.gat.io.File preStagedFileList [],
// postStagedFileList [] accordingly
import org.gridlab.gat;
import org.gridlab.gat.resources;
SoftwareDescription sd = new SoftwareDescription ();
sd.setPreStaged   (preStagedFileList);
sd.setPostStaged (postStagedFileList);
sd.setStdin  (GAT.createFile(ctx, prefs, new URI (“any:///stdinFile”)));
sd.setStdout(GAT.createFile(ctx, prefs, new URI (“any:///stdoutFile”)));
sd.setStderr(GAT.createFile (ctx, prefs, new URI (“any:///stderrFile”)));
sd.setLocation(new URI (“any:///executableFilePath”));
```

The Bad

- Lack of brokerage/scheduling awareness
  - Programmer has to deal with brokerage/scheduling issues
    → literal host names in the code
  - Still no good grid resource management system available
    → GAT can't help this, but the user sees it as a GAT problem

The Ugly

- Credentials
  - grid-proxy-init
  - Multiple CAs strategies for certificate protection
- Lack of abstraction for remote/local specifications
  - Would be nice to have something like (semantic specification):

```
preferences.put("ResourceBroker.type", "remote");
broker  = GAT.createResourceBroker(ctx, prefs);
```

The Ugly (cont.)

- The id does not survive the execution of a job
  - the task incarnation, although it is a globally unique identifier
- Semantic issues
  - Like removeListener invoked in a processEvent handler: should this be allowed or not? Anyway, the decision should be specified in the API

Conclusions and Future Work

- If the middleware is faulty, users perceive it as a GAT problem
- Port & Test on Almere Grid