Ibis as Master Key

Niels Drost

Computer Systems Group
Department of Computer Science
VU University, Amsterdam, The Netherlands

Today's Program

10.00: Introduction

11.00: Ibis as Master Key

12.00: Lunch

13.00: Ibis as Glue

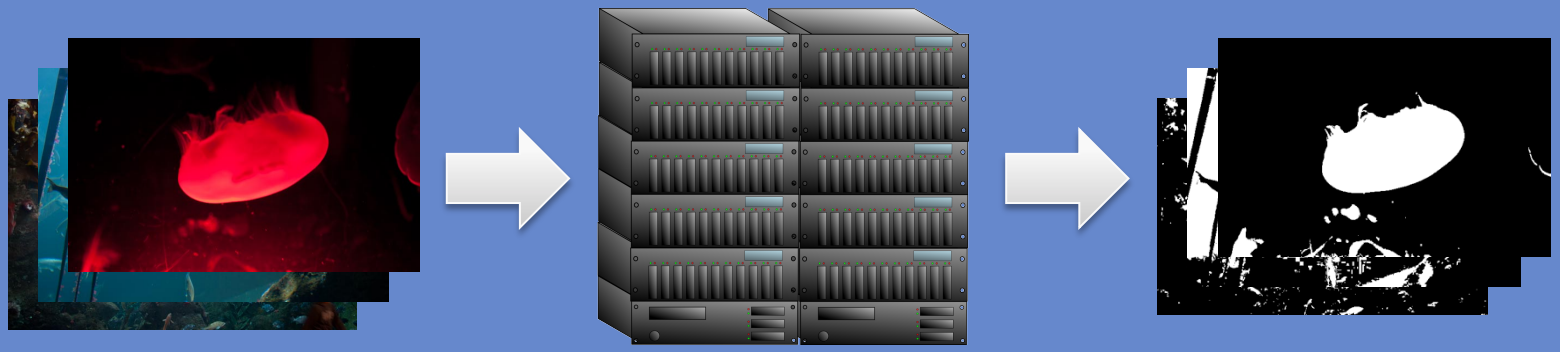
14.00: Coffee

14.15: Ibis Deploy and Coupled Codes

14.30: The Future of Ibis

Running Example

- Scientist needs to process a large number of independent data files using some application
- Offload to compute resource using Task Farming
- Today
  - Tool: ImageMagic “convert”
  - Data: Pictures



Deployment

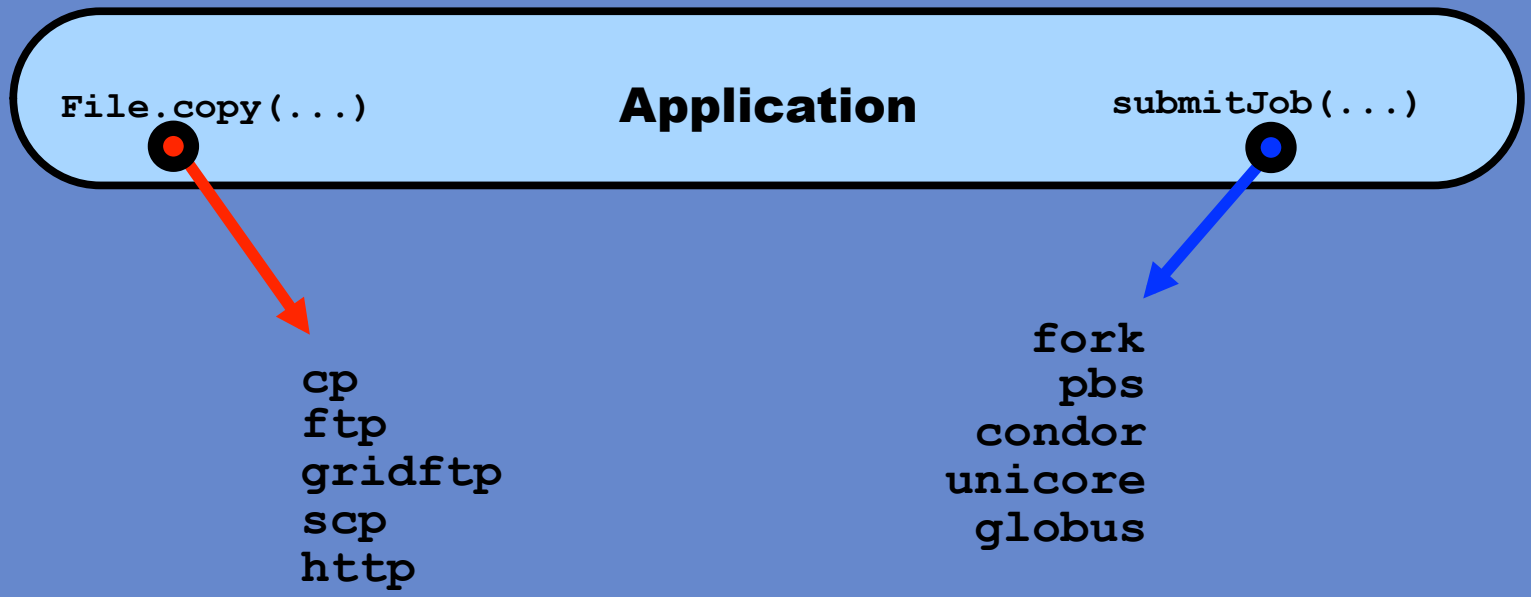
- How to get your application running in the Jungle
- For each resource used:
  - Find resource
  - Copy input files
  - Reserve resource
  - Run application
  - Copy back output files
- Access to remote resources using Middleware

Middleware

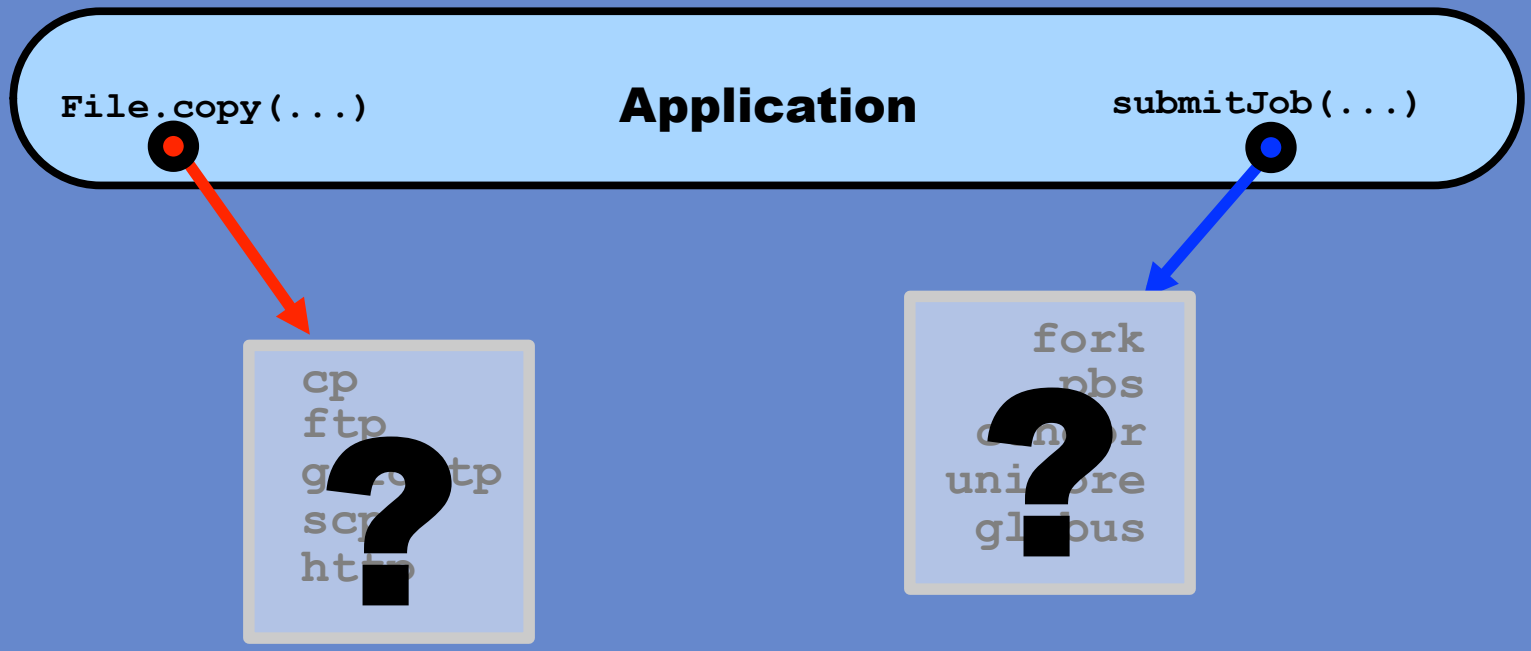
- Resources invariable use some sort of Middleware
- Provide remote access to resources
- File copy, running applications, etc
- Many different middleware available:
  - Globus (de facto standard, in 4 Flavors)
  - gLite, NAREGI, UNICORE, Legion
  - SSH (poor man's middleware)

Typical Jungle Application

Typical Jungle Application



Typical Jungle Application



Which Middleware do I use?

- A lot to choose from
- Some may not work on all sites
- Most are hard to use
- Interfaces change often
- Globus? (Obvious choice 3 years ago)

Globus File Copy (C++, 2004)

```
package org.globus.ogsa.gui;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.net.URL;
import java.util.Date;
import java.util.Vector;
import javax.xml.rpc.Stub;
import org.apache.axis.message.MessageElement;
import org.apache.axis.utils.XMLUtils;
import org.globus.*
import org.gridforum.ogsi.*
import org.gridforum.ogsi.holders.TerminationTimeTypeHolder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class RFTClient {
 public static void copy (String source_url, String target_url) {
  try {
   File requestFile      = new File (source_url);
   BufferedReader reader = null;
   try {
    reader = new BufferedReader (new FileReader (requestFile));
   } catch (java.io.FileNotFoundException fnfe) { }
   Vector requestData = new Vector ();
   requestData.add (target_url);
   TransferType[] transfers1      = new TransferType[transferCount];
   RFTOptionsType multirftOptions = new RFTOptionsType ();

   multirftOptions.setBinary          (Boolean.valueOf  (
        (String)requestData.elementAt (0)).booleanValue ());
   multirftOptions.setBlockSize       (Integer.valueOf  (
        (String)requestData.elementAt (1)).intValue     ());
   multirftOptions.setTcpBufferSize   (Integer.valueOf  (
        (String)requestData.elementAt (2)).intValue     ());
   multirftOptions.setNotpt           (Boolean.valueOf  (
        (String)requestData.elementAt (3)).booleanValue ());
   multirftOptions.setParallelStreams (Integer.valueOf  (
        (String)requestData.elementAt (4)).intValue     ());
   multirftOptions.setDcau(Boolean.valueOf(
         (String)requestData.elementAt (5)).booleanValue ());

  int i = 7;
  for (int j = 0; j < transfers1.length; j++)
  {
   transfers1[j] = new TransferType ();

   transfers1[j].setTransferId     (j);
   transfers1[j].setSourceUrl      ((String)requestData.elementAt (i++));
   transfers1[j].setDestinationUrl ((String)requestData.elementAt (i++));
   transfers1[j].setRftOptions     (multirftOptions);
  }

  TransferRequestType transferRequest = new TransferRequestType ();
  transferRequest.setTransferArray (transfers1);

  int concurrency = Integer.valueOf
                    ((String)requestData.elementAt(6)).intValue();

  if (concurrency > transfers1.length)
  {
   System.out.println ("Concurrency should be less than the number"
                       "of transfers in the request");
   System.exit (0);
  }
  transferRequest.setConcurrency (concurrency);

  TransferRequestElement requestElement = new TransferRequestElement ();
  requestElement.setTransferRequest (transferRequest);

  ExtensibilityType extension = new ExtensibilityType ();
  extension = AnyHelper.getExtensibility (requestElement);

  OGSIServiceGridLocator factoryService = new OGSIServiceGridLocator ();
  Factory factory = factoryService.getFactoryPort (new URL (source_url));
  GridServiceFactory gridFactory = new GridServiceFactory (factory);

  LocatorType locator = gridFactory.createService (extension);
  System.out.println ("Created an instance of Multi-RFT");

  MultiFileRFTDefinitionServiceGridLocator loc
                    = new MultiFileRFTDefinitionServiceGridLocator();
  RFTPortType rftPort = loc.getMultiFileRFTDefinitionPort (locator);
   ((Stub)rftPort)._setProperty (Constants.AUTHORIZATION,
                        NoAuthorization.getInstance());
   ((Stub)rftPort)._setProperty (GSIConstants.GSI_MODE,
                        GSIConstants.GSI_MODE_FULL_DELEG);
   ((Stub)rftPort)._setProperty (Constants.GSI_SEC_CONV,
                        Constants.SIGNATURE);
   ((Stub)rftPort)._setProperty (Constants.GRIM_POLICY_HANDLER,
                        new IgnoreProxyPolicyHandler ());

   int requestid = rftPort.start ();
   System.out.println ("Request id: " + requestid);

  }
  catch (Exception e)
  {
   System.err.println (MessageUtils.toString (e));
  }
 }
```

The Problem with Middleware

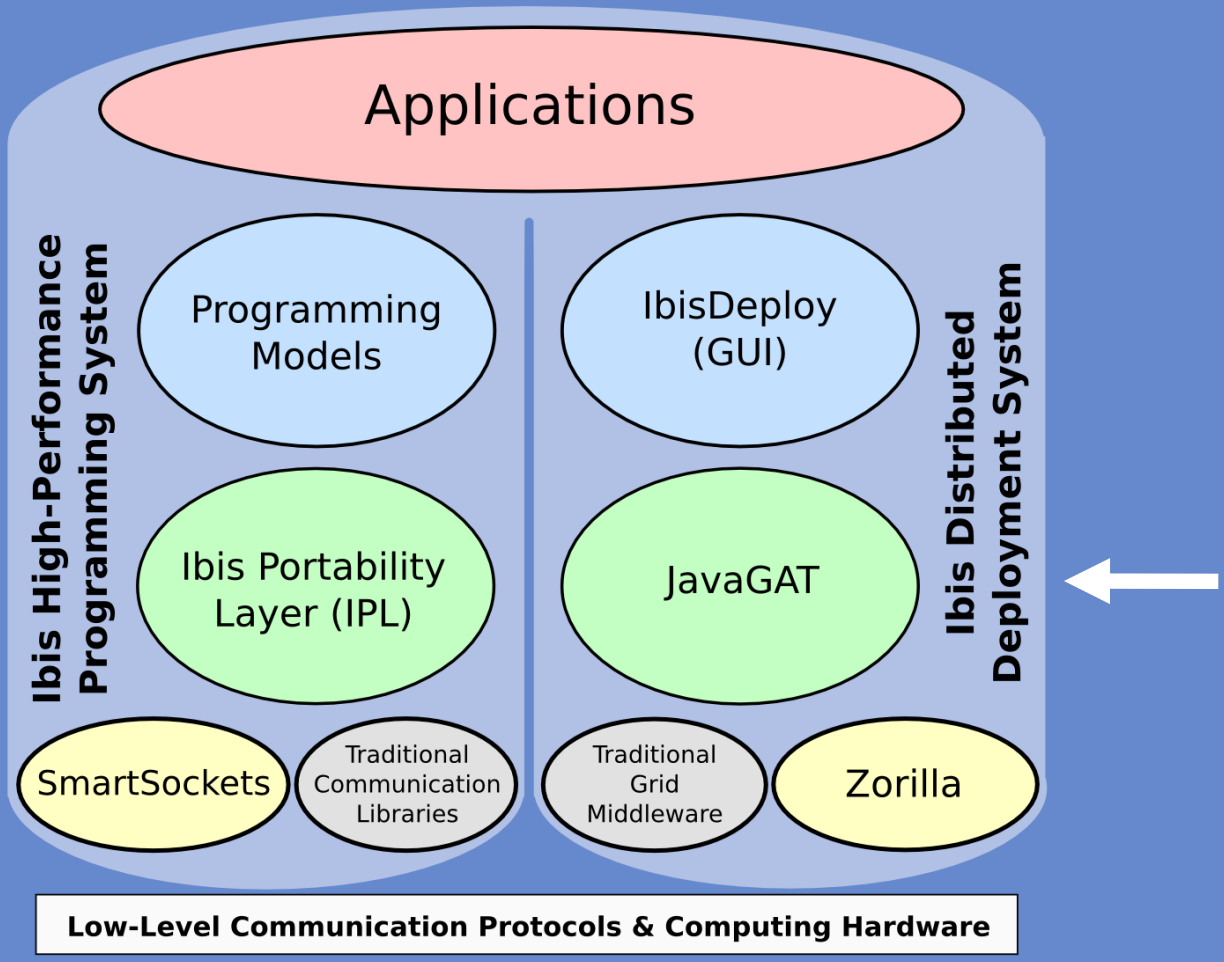
- There are too many different Middleware
- With wildly different interfaces
- Which are too low level

Using multiple different resources at the same time is neigh impossible using middleware directly

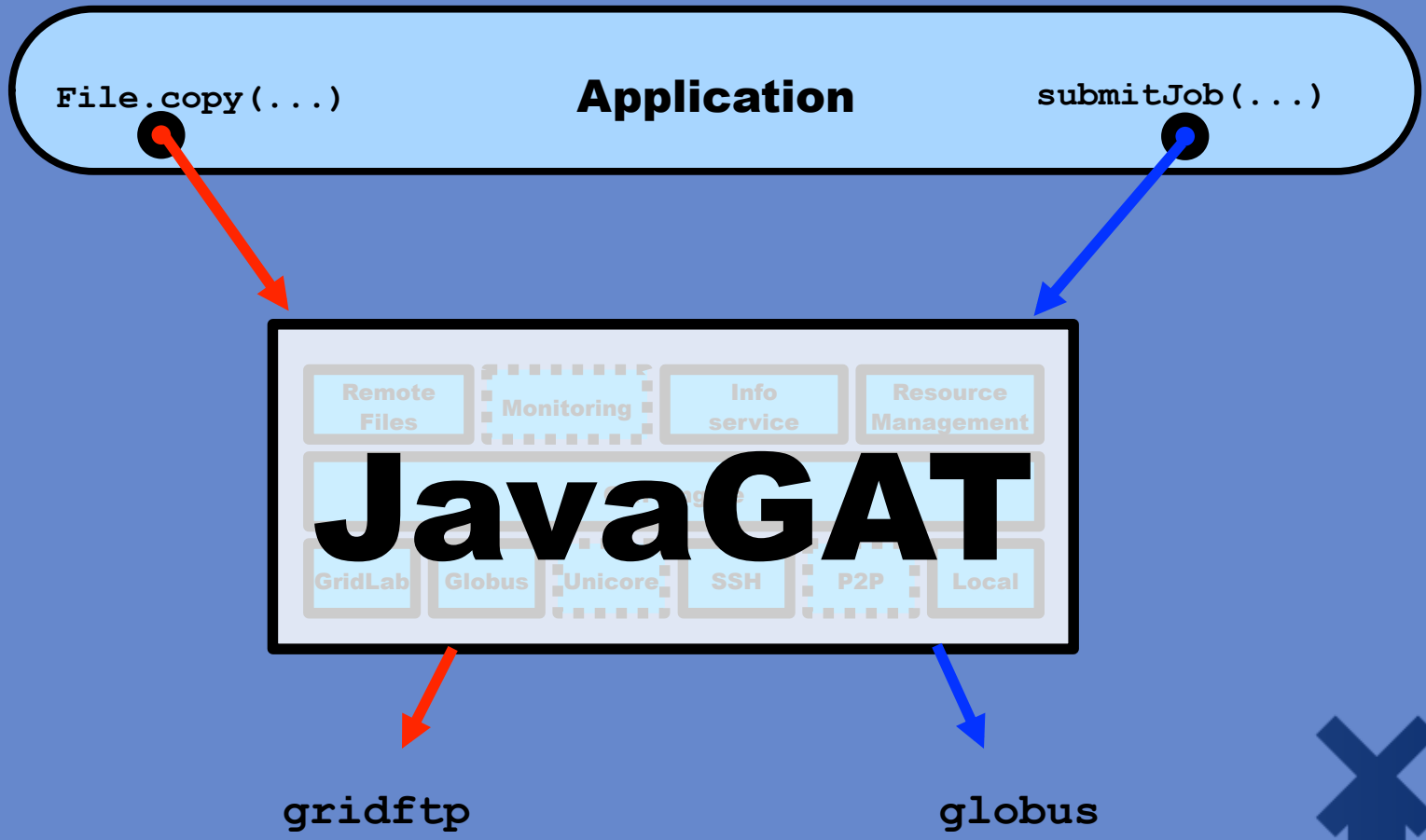
JavaGAT

- Java Grid Application Toolkit
- Layer between the application and the Middleware
- Simple API
- File Copy, Job submission, Monitoring
- Functionality provided by Adaptors
  - All major (and most minor) middleware supported
- Assumes middleware is buggy, can fail at all time, is not configured properly, etc

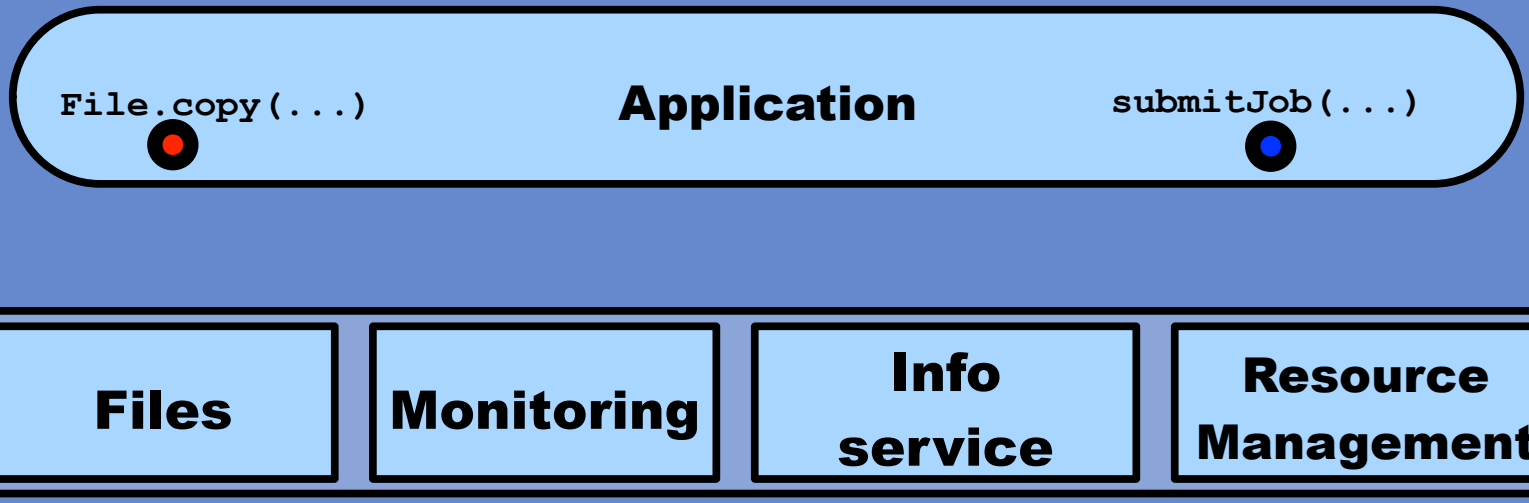
Where are we?


Applications
Ibis High-Performance Programming System
Programming Models
Ibis Portability Layer (IPL)
SmartSockets
Traditional Communication Libraries
Ibis Distributed Deployment System
IbisDeploy (GUI)
JavaGAT
Traditional Grid Middleware
Zorilla
Low-Level Communication Protocols & Computing Hardware

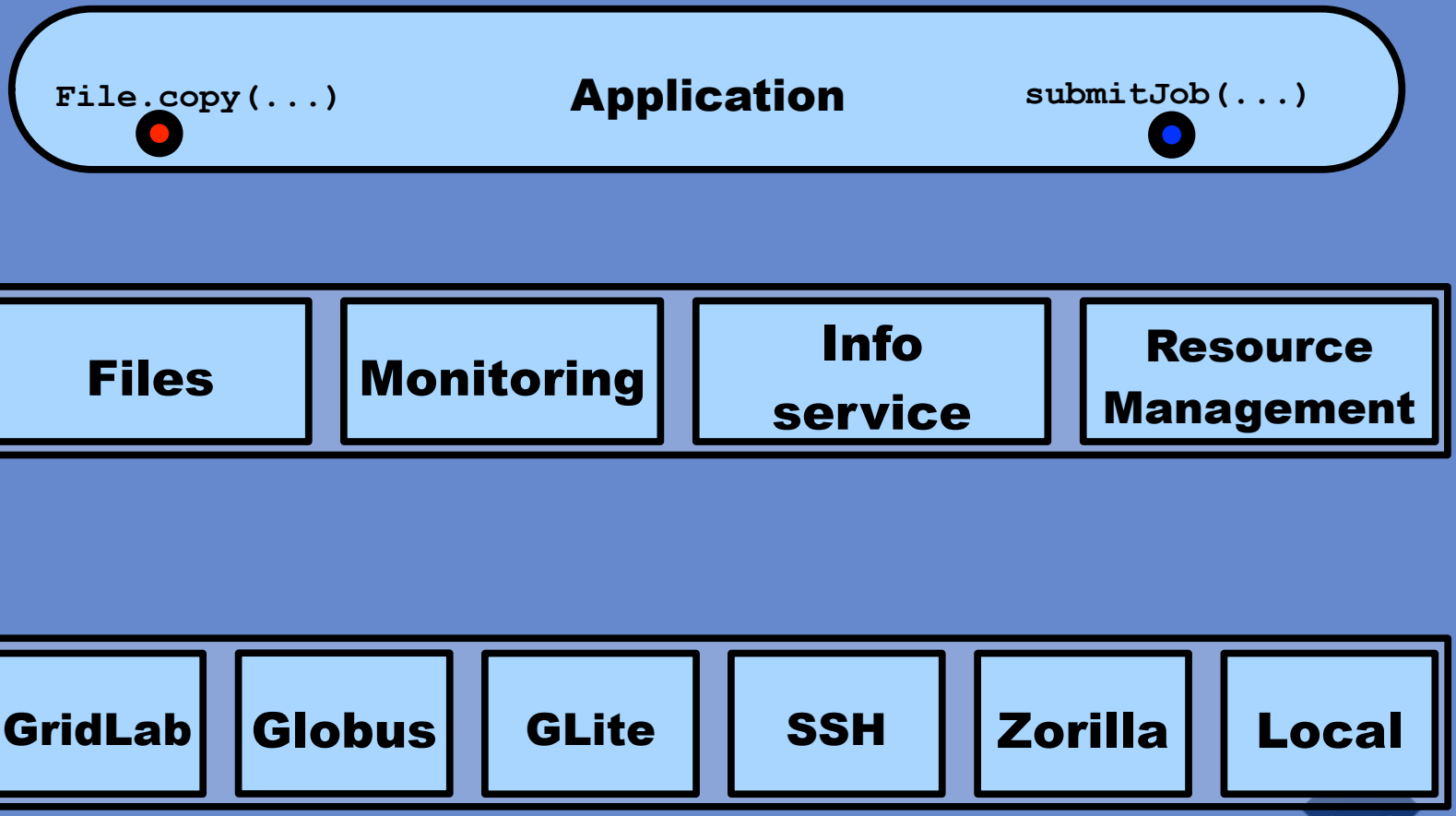
JavaGAT


File.copy(...)
Application
submitJob(...)
Remote Files
Monitoring
Info service
Resource Management
JavaGAT
GridLab
Globus
Unicore
SSH
P2P
Local
gridftp
globus

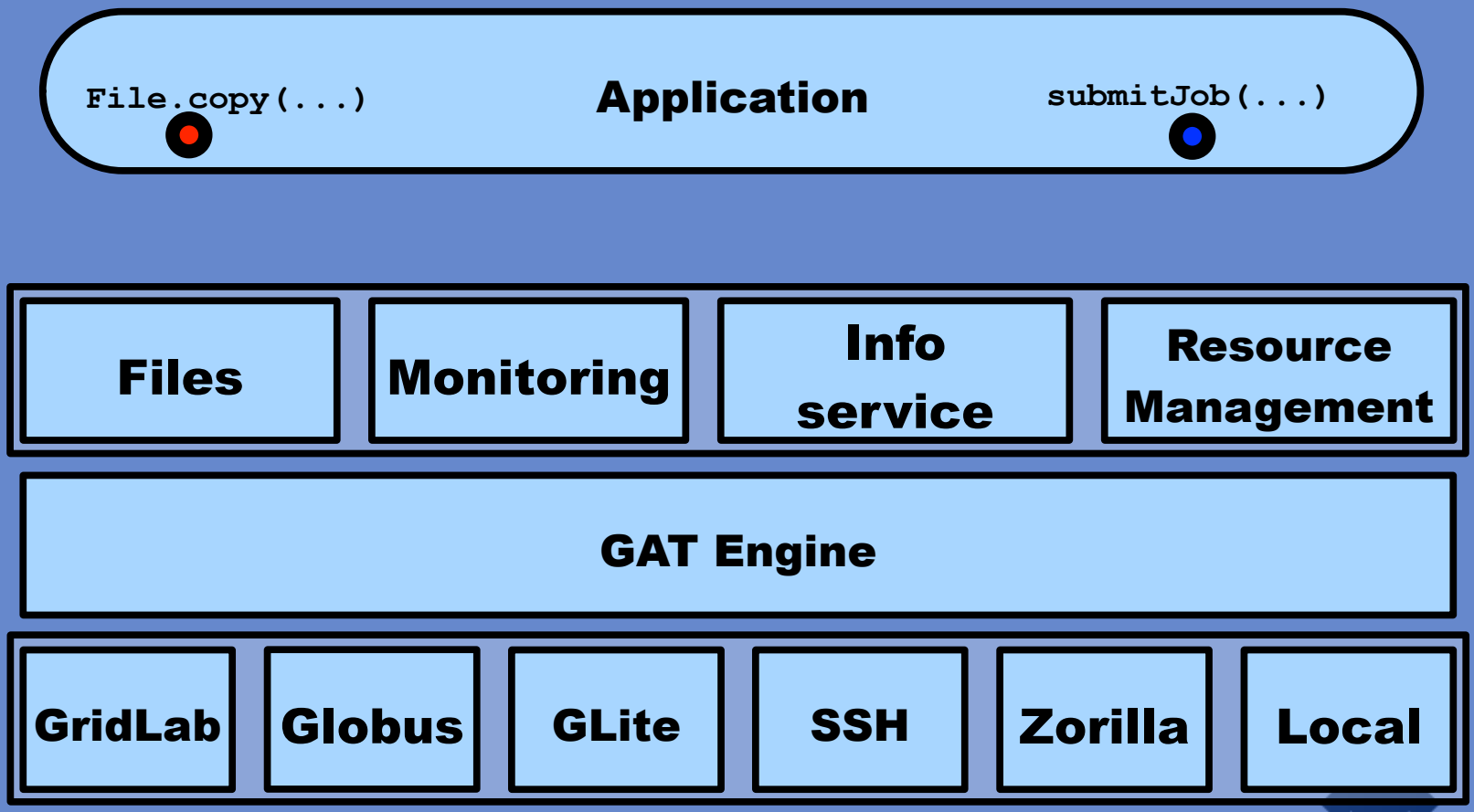
JavaGAT Engine



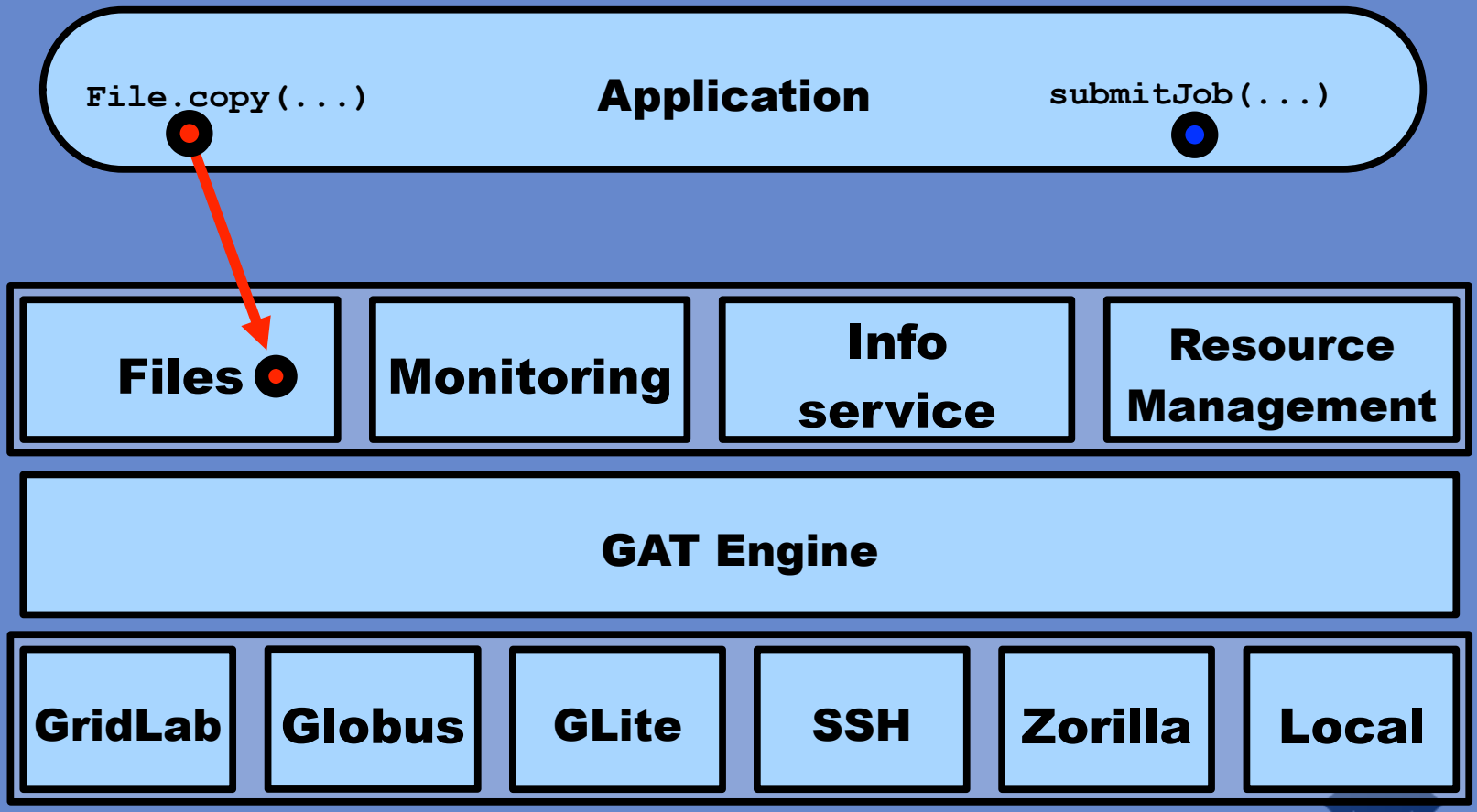
JavaGAT Engine



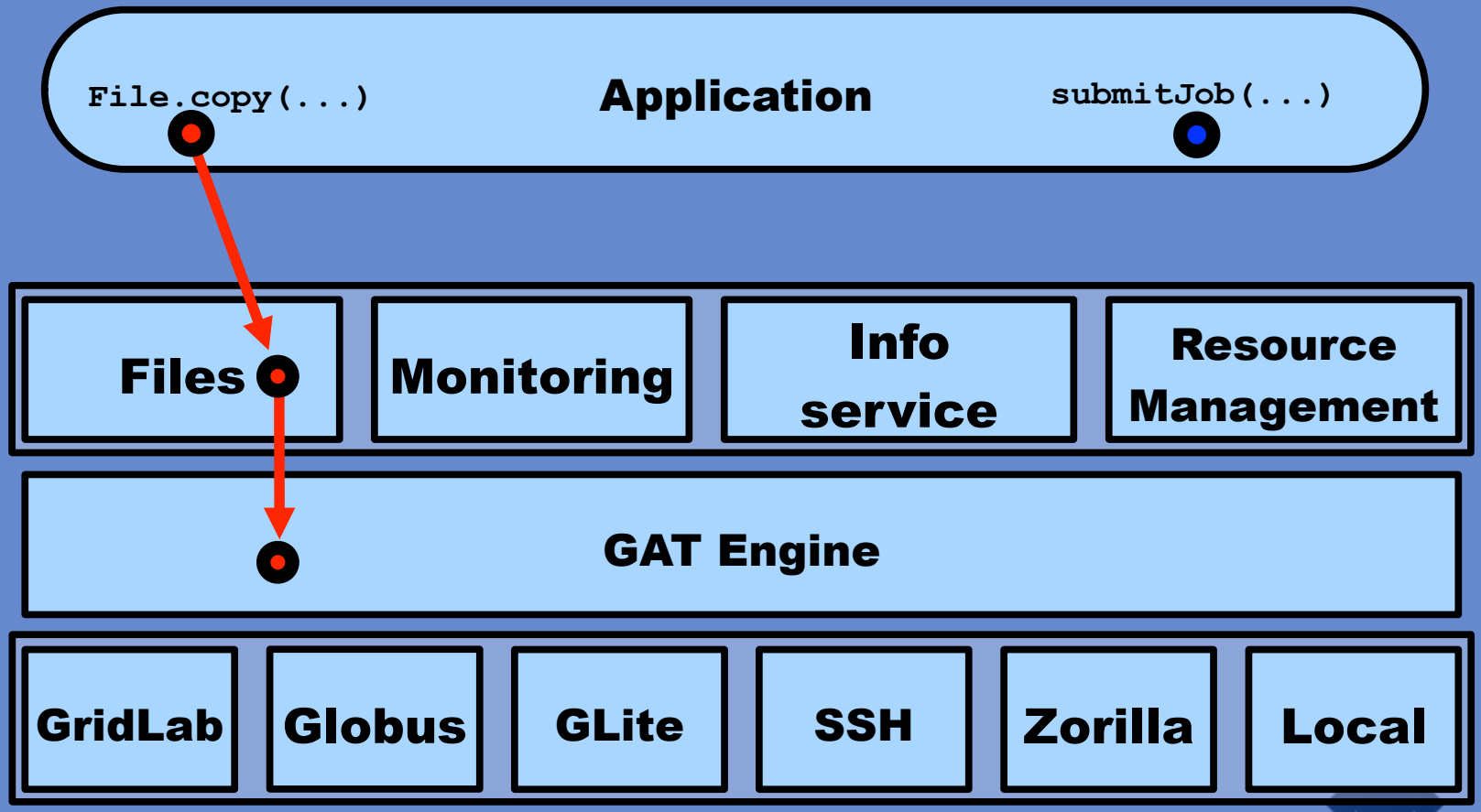
JavaGAT Engine



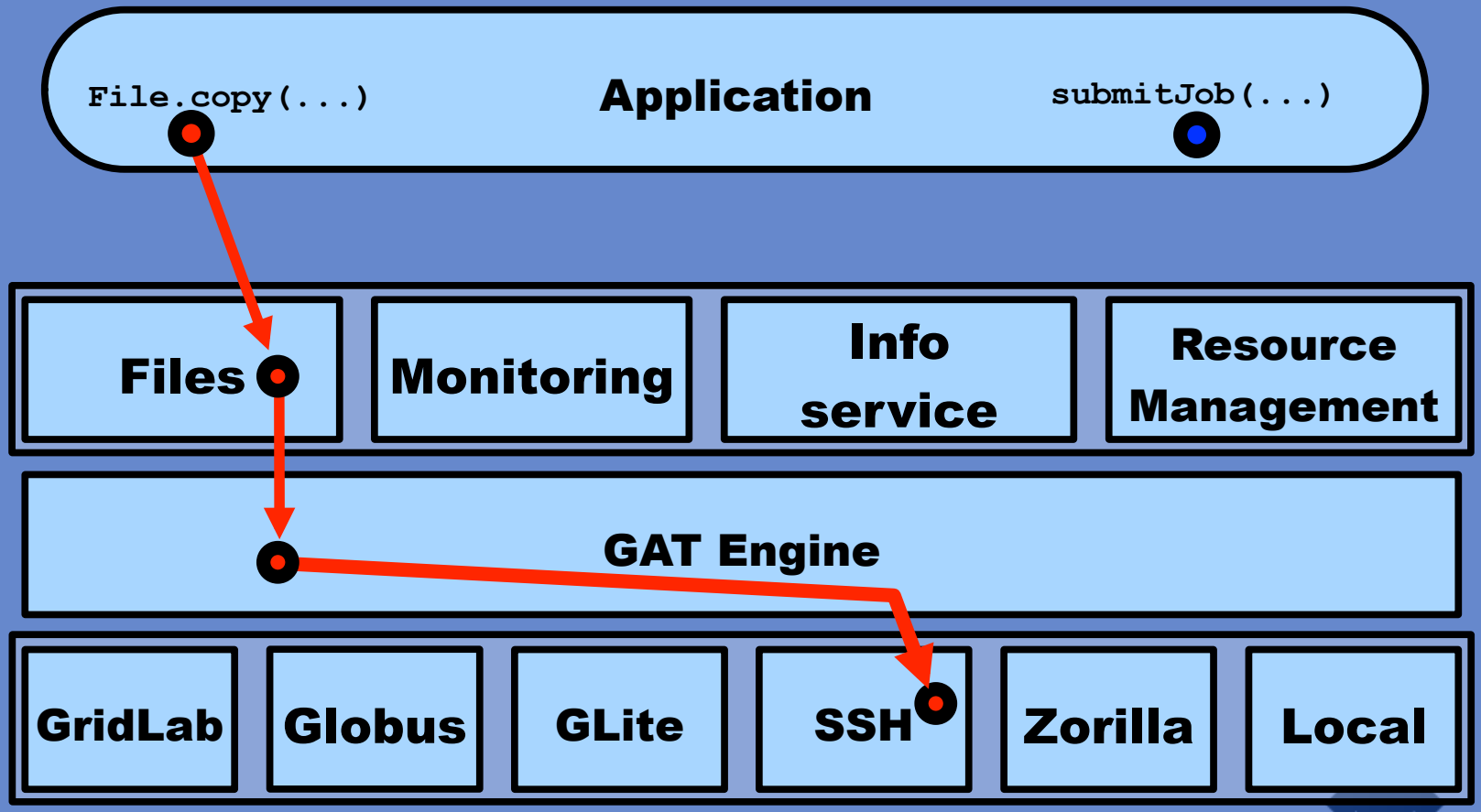
JavaGAT Engine



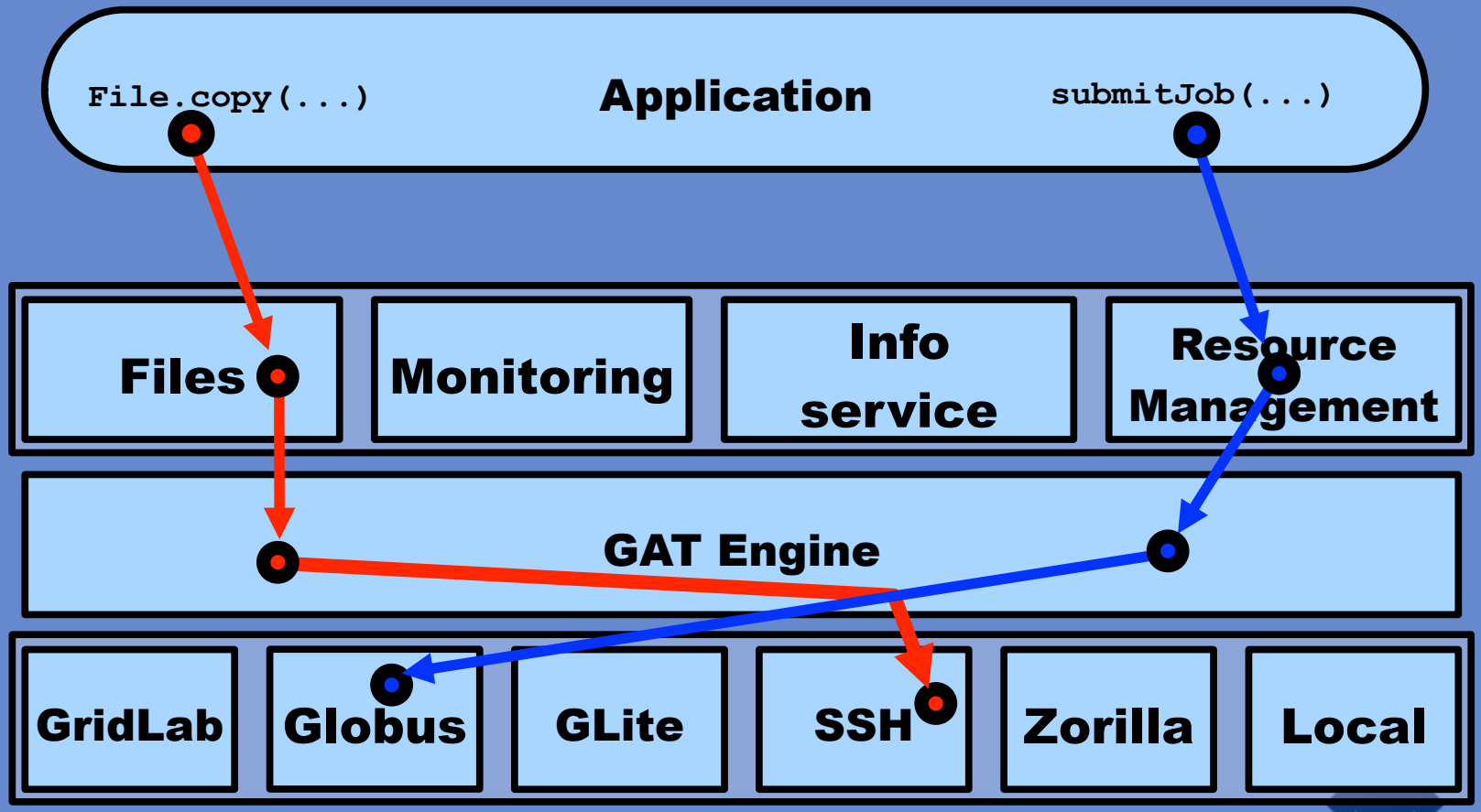
JavaGAT Engine



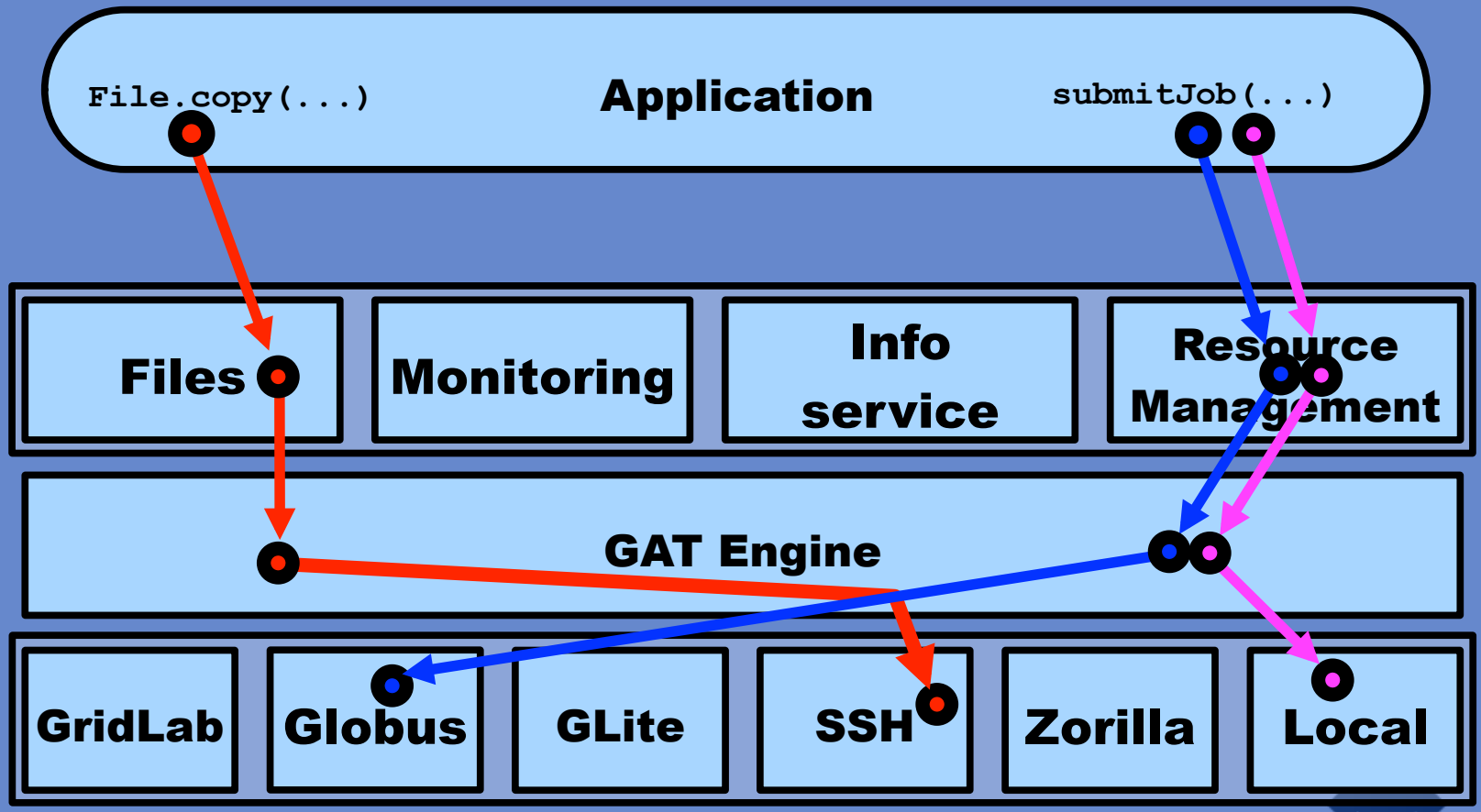
JavaGAT Engine



JavaGAT Engine



JavaGAT Engine



Globus File Copy (C++) (Revisited)

```
package org.globus.ogsa.gui;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.net.URL;
import java.util.Date;
import java.util.Vector;
import javax.xml.rpc.Stub;
import org.apache.axis.message.MessageElement;
import org.apache.axis.utils.XMLUtils;
import org.globus.*
import org.gridforum.ogsi.*
import org.gridforum.ogsi.holders.TerminationTimeTypeHolder;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class RFTClient {
 public static void copy (String source_url, String target_url) {
  try {
   File requestFile      = new File (source_url);
   BufferedReader reader = null;
   try {
    reader = new BufferedReader (new FileReader (requestFile));
   } catch (java.io.FileNotFoundException fnfe) { }
   Vector requestData = new Vector ();
   requestData.add (target_url);
   TransferType[] transfers1      = new TransferType[transferCount];
   RFTOptionsType multirftOptions = new RFTOptionsType ();

   multirftOptions.setBinary          (Boolean.valueOf  (
        (String)requestData.elementAt (0)).booleanValue ());
   multirftOptions.setBlockSize       (Integer.valueOf  (
        (String)requestData.elementAt (1)).intValue     ());
   multirftOptions.setTcpBufferSize   (Integer.valueOf  (
        (String)requestData.elementAt (2)).intValue     ());
   multirftOptions.setNotpt           (Boolean.valueOf  (
        (String)requestData.elementAt (3)).booleanValue ());
   multirftOptions.setParallelStreams (Integer.valueOf  (
        (String)requestData.elementAt (4)).intValue     ());
   multirftOptions.setDcau(Boolean.valueOf(
         (String)requestData.elementAt (5)).booleanValue ());

  int i = 7;
  for (int j = 0; j < transfers1.length; j++)
  {
   transfers1[j] = new TransferType ();

   transfers1[j].setTransferId     (j);
   transfers1[j].setSourceUrl      ((String)requestData.elementAt (i++));
   transfers1[j].setDestinationUrl ((String)requestData.elementAt (i++));
   transfers1[j].setRftOptions     (multirftOptions);
  }

  TransferRequestType transferRequest = new TransferRequestType ();
  transferRequest.setTransferArray (transfers1);

  int concurrency = Integer.valueOf
              ((String)requestData.elementAt(6)).intValue();

  if (concurrency > transfers1.length)
  {
   System.out.println ("Concurrency should be less than the number"
                       "of transfers in the request");
   System.exit (0);
  }
  transferRequest.setConcurrency (concurrency);

  TransferRequestElement requestElement = new TransferRequestElement ();
  requestElement.setTransferRequest (transferRequest);

  ExtensibilityType extension = new ExtensibilityType ();
  extension = AnyHelper.getExtensibility (requestElement);

  OGSIServiceGridLocator factoryService = new OGSIServiceGridLocator ();
  Factory factory = factoryService.getFactoryPort (new URL (source_url));
  GridServiceFactory gridFactory = new GridServiceFactory (factory);

  LocatorType locator = gridFactory.createService (extension);
  System.out.println ("Created an instance of Multi-RFT");

  MultiFileRFTDefinitionServiceGridLocator loc
              = new MultiFileRFTDefinitionServiceGridLocator();
  RFTPortType rftPort = loc.getMultiFileRFTDefinitionPort (locator);
   ((Stub)rftPort)._setProperty (Constants.AUTHORIZATION,
                        NoAuthorization.getInstance());
   ((Stub)rftPort)._setProperty (GSIConstants.GSI_MODE,
                        GSIConstants.GSI_MODE_FULL_DELEG);
   ((Stub)rftPort)._setProperty (Constants.GSI_SEC_CONV,
                        Constants.SIGNATURE);
   ((Stub)rftPort)._setProperty (Constants.GRIM_POLICY_HANDLER,
                        new IgnoreProxyPolicyHandler ());

   int requestid = rftPort.start ();
   System.out.println ("Request id: " + requestid);

  }
  catch (Exception e)
  {
   System.err.println (MessageUtils.toString (e));
  }
}
```

File Copy with JavaGAT

```
import org.gridlab.gat.GAT;
import org.gridlab.gat.URI;

public class Copy {

public static void main(String[] args) throws Exception {
        GAT.createFile(args[0]).copy(new URI(args[1]));
        GAT.end();
    }
}
```

File Copy with JavaGAT

```
import org.gridlab.gat.GAT;
import org.gridlab.gat.URI;

public class Copy {

    public static void main(String[] args) throws Exception {
        GAT.createFile(args[0]).copy(new URI(args[1]));
        GAT.end();
    }
}
```

- URI's used as file location
- Note: This is an actual program, not pseudo code!

Starting a Job

- The files are there, now start the application
- URI's used to denote location of resource
  - Machine: ssh://fs0.das4.cs.vu.nl
  - Globus: globus://fs0.das3.cs.vu.nl/jobmanager-sge
  - SGE through ssh: sshsge://fs0.das4.cs.vu.nl
- Application specified using SoftwareDescription
  - Executable, Arguments, etc
- Job Specified using JobDescription object
  - Number of nodes, number of processes, etc…

```java
public class RunJob {

    public static void main(String[] args) throws Exception {
        ResourceBroker broker = GAT.createResourceBroker(new URI(args[0]));

        SoftwareDescription sd = new SoftwareDescription();

        sd.setExecutable(args[1]);

        sd.setStdout(GAT.createFile("stdout.txt"));
        sd.setStderr(GAT.createFile("stderr.txt"));

        Job job = broker.submitJob(new JobDescription(sd));

        do {
            System.out.println("Current state: " + job.getState());
            Thread.sleep(1000);
        } while ((job.getState() != JobState.STOPPED)
                && (job.getState() != JobState.SUBMISSION_ERROR));

        GAT.end();
    }
}
```

Security

- In General, JavaGAT adaptors automatically pick-up on security mechanisms
  - .ssh/* files with SSH
  - .globus/* files with globus
- However, sometimes a user would like some more control, and specify security info manually
- GATContext, containing settings of JavaGAT
- SecurityContext, containing security info

Security Example

```
public static void main(String[] args) throws Exception {
    CertificateSecurityContext securityContext = new CertificateSecurityContext(
            new URI(System.getProperty("user.home") + "/.globus/userkey.pem"),
            new URI(System.getProperty("user.home") + "/.globus/usercert.pem"),
            getPassphrase());

    GATContext context = new GATContext();

    context.addSecurityContext(securityContext);

    ResourceBroker broker = GAT.createResourceBroker(context, new URI(
            args[0]));

    ...
```

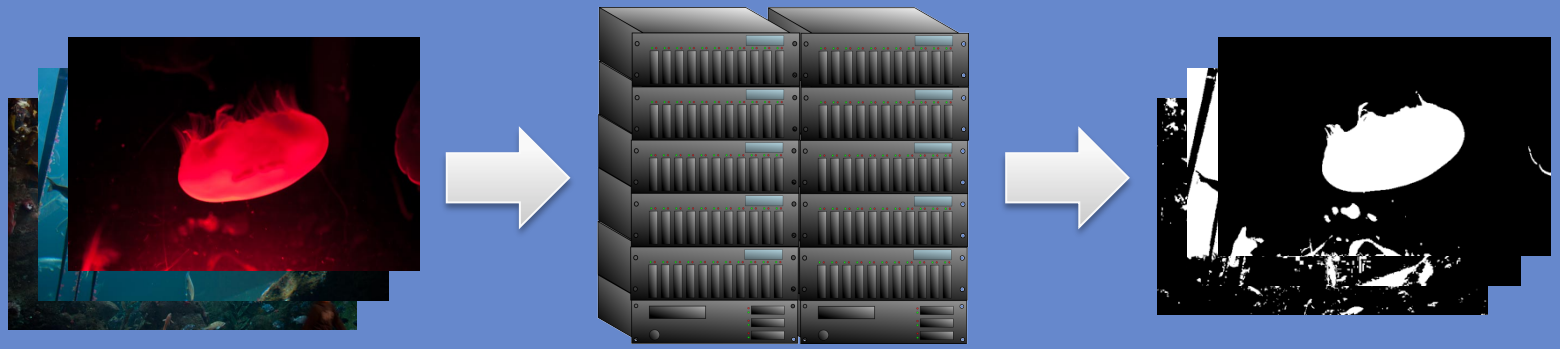
Next: File staging

- Keeping track of files cumbersome, especially for a lot of jobs
- File staging: Automatic copying of files associated with job to and from resource
- Sandbox automatically created for each job
  - Files put in this sandbox
  - Current working directory of Job.
  - Option to wipe when done for sensitive data
- Example: generic remote execution app

```java
public class RunJobWithStaging {
    // USAGE: machine executable input [arguments…] output
    public static void main(String[] args) throws Exception {
        ResourceBroker broker = GAT.createResourceBroker(new URI(args[0]));

        SoftwareDescription sd = new SoftwareDescription();
        sd.setExecutable(args[1]);
        sd.setStdout(GAT.createFile("stdout.txt"));
        sd.setStderr(GAT.createFile("stderr.txt"));

        sd.addPreStagedFile(GAT.createFile(args[2]));
        sd.addPostStagedFile(GAT.createFile(args[args.length - 1]));

        sd.setArguments(getArguments(args));
        Job job = broker.submitJob(new JobDescription(sd));

        do {
            System.out.println("Current state: " + job.getState());
            Thread.sleep(1000);
        } while ((job.getState() != JobState.STOPPED)
                && (job.getState() != JobState.SUBMISSION_ERROR));

        GAT.end();
    }
```

Task Farming Example

- Using the code explained so far we can create a simple task farming application already!
- Directory with Input files
- Directory with output files
- Executable, arguments, resource to use



```
public class SimpleTaskFarming {
    //USAGE: MACHINE EXECUTABLE INPUT_DIR [ARGUMENTS ...] OUTPUT_DIR
    public static void main(String[] args) throws Exception {
        ResourceBroker broker = GAT.createResourceBroker(new URI(args[0]));

        String executable = args[1];
        String inputdir = args[2];
        String outputdir = args[args.length - 1];

        String[] inputs = listInputs(inputdir, ".jpg");

        Job[] jobs = new Job[inputs.length];

        for (int i = 0; i < inputs.length; i++) {
            SoftwareDescription sd = new SoftwareDescription();
            sd.setExecutable(executable);

            File input = GAT.createFile(inputdir + File.separator + inputs[i]);
            sd.addPreStagedFile(input);
            File output = GAT.createFile(outputdir + File.separator + "out-" + input.getName());
            sd.addPostStagedFile(GAT.createFile(output.getName()), output);

            sd.setStdout(GAT.createFile("stdout-" + i + ".txt"));
            sd.setStderr(GAT.createFile("stderr-" + i + ".txt"));

            // Set the arguments and submit the job.
            sd.setArguments(prepareArguments(input.getName(), getArguments(args),
                    output.getName()));

            jobs[i] = broker.submitJob(new JobDescription(sd));
        }

        waitUntilFinished(jobs);
        GAT.end();
    }
```

```
public class SimpleTaskFarming {
    //USAGE: MACHINE EXECUTABLE INPUT_DIR [ARGUMENTS ...] OUTPUT_DIR
    public static void main(String[] args) throws Exception {
        ResourceBroker broker = GAT.createResourceBroker(new URI(args[0]));

        String executable = args[1];
        String inputdir = args[2];
        String outputdir = args[args.length - 1];

        String[] inputs = listInputs(inputdir, ".jpg");

        Job[] jobs = new Job[inputs.length];

        for (int i = 0; i < inputs.length; i++) {
            File output = GAT.createFile(outputdir + File.separator + "out-" + input.getName());
            sd.addPostStagedFile(GAT.createFile(output.getName()), output);

            sd.setStdout(GAT.createFile("stdout-" + i + ".txt"));
            sd.setStderr(GAT.createFile("stderr-" + i + ".txt"));

            // Set the arguments and submit the job.
            sd.setArguments(prepareArguments(input.getName(), getArguments(args),
                    output.getName()));

            jobs[i] = broker.submitJob(new JobDescription(sd));
        }

        waitUntilFinished(jobs);
        GAT.end();
    }
```

```
public class SimpleTaskFarming {
    //USAGE: MACHINE EXECUTABLE INPUT_DIR [ARGUMENTS ...] OUTPUT_DIR
    public static void main(String[] args) throws Exception {
        for (int i = 0; i < inputs.length; i++) {
            SoftwareDescription sd = new SoftwareDescription();
            sd.setExecutable(executable);

            File input = GAT.createFile(inputdir + File.separator + inputs[i]);
            sd.addPreStagedFile(input);
            File output = GAT.createFile(outputdir + File.separator
                                                   + "out-" + input.getName());
            sd.addPostStagedFile(GAT.createFile(output.getName()), output);

            sd.setStdout(GAT.createFile("stdout-" + i + ".txt"));
            sd.setStderr(GAT.createFile("stderr-" + i + ".txt"));

            // Set the arguments and submit the job.
            sd.setArguments(prepareArguments(input.getName(), getArguments(args),
                    output.getName()));

            jobs[i] = broker.submitJob(new JobDescription(sd));
        }

        waitUntilFinished(jobs);
        GAT.end();
    }
```

Multi Resource Task Farming

- Same as previous, with multiple resources
- Spread jobs equally over a set of resources
- Jungle Computing already ;-)
- Only 140 lines of code
  - With comments
  - Without error handling

```java
public class MultiSiteTaskFarming {
    // USAGE: MACHINE[,MACHINE, ...] EXECUTABLE INPUT_DIR [ARGUMENTS ...] OUTPUT_DIR
    public static void main(String[] args) throws Exception {
        String[] brokerURIs = args[0].split(",");

        ResourceBroker[] brokers = new ResourceBroker[brokerURIs.length];
        for (int i = 0; i < brokers.length; i++) {
            brokers[i] = GAT.createResourceBroker(new URI(brokerURIs[i]));
        }

        // ...

        for (int i = 0; i < inputs.length; i++) {

            ResourceBroker broker = brokers[i % brokers.length];

            jobs[i] = broker.submitJob(jobDescription);
        }

        waitUntilFinished(jobs);
        GAT.end();
    }
```

Multi Resource Task Farming

- Great! However:
  - Separate job for each task adds a lot of overhead in case tasks and/or files are small
  - Machines get more-and-more cores, may require multiple jobs per node to use it efficiently
  - Stand in the q for each job separately, may be a long time if the cluster is busy
- Not all applications are Task Farming!

Remarks

- JavaGAT standardized as SAGA
- Not all adaptors implement all functionality
  - But an other adaptor will be chosen automatically
- Not only for Grids, despite its name

Conclusion

- Ibis can be used as a “Master Key” to many different resources using the JavaGAT
- The JavaGAT offers a Simple yet powerful interface to a lot of different middleware
- JavaGAT compensates for the complexity and faultiness of current middleware
- JavaGAT can run any application, not just Java