



# Ibis as Master Key

Hands-on

Niels Drost

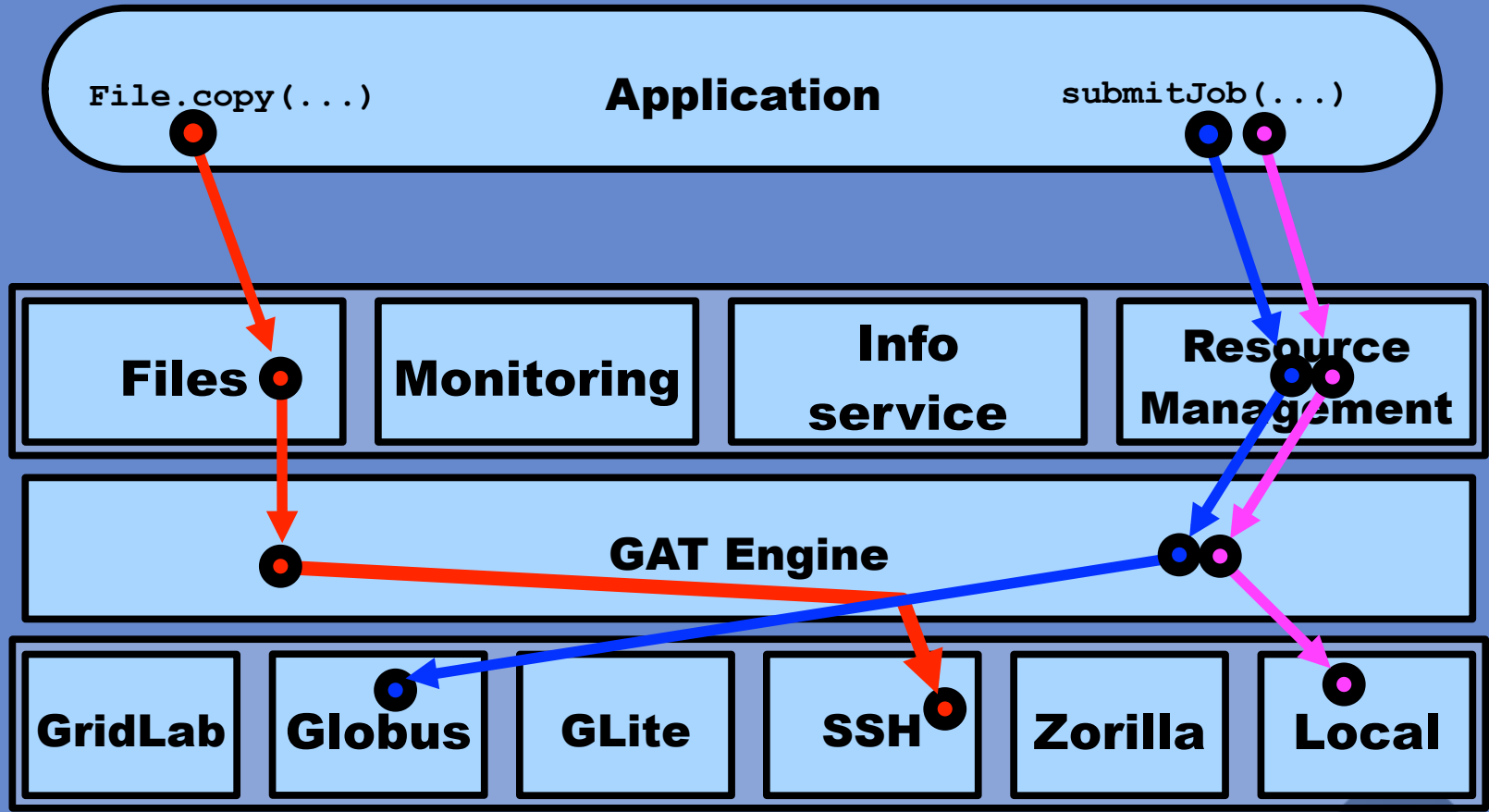
Computer Systems Group

Department of Computer Science

VU University, Amsterdam, The Netherlands



# JavaGAT Design



# Prerequisites

---

- These slides: [www.cs.vu.nl/ibis/tutorial.html](http://www.cs.vu.nl/ibis/tutorial.html)
- Sun/Oracle Java (1.6 or 1.7 both fine)  
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- Apache ANT:  
<http://ant.apache.org/bindownload.cgi>
- Eclipse (optional): <http://www.eclipse.org/>
- Login using ssh without a password



# Getting started with the Examples

---

- From tutorial page [www.cs.vu.nl/ibis/tutorial.html](http://www.cs.vu.nl/ibis/tutorial.html)
  - Download JavaGAT release
  - Download masterkey-examples.zip
- Unzip both files
- Set “javagat” property in “masterkey-examples/build.xml” to proper JavaGAT location
- Build using “ant”



# 1) File Copy

---

- Class: `masterkey.basic.Copy`
- Run using `scripts/run(.bat) CLASS ....`
- See `masterkey.examples`
- 1a. Try to copy to/from:
  - Local machine
  - DAS-3/4
  - Your own machine!
- 1b. Try to edit source to rename/delete/etc
  - See JavaGAT Javadoc on tutorial site



# 1) File Copy

```
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



## 2) Running jobs

---

- Class: `masterkey.basic.RunJob`
- 2) run some commands on
  - Localhost (`local://localhost`)
  - The DAS-4 VU frontend (`ssh://fs0.das4.cs.vu.nl`)
  - The DAS-4 nodes (`sshsgc://fs0.das4.cs.vu.nl`)
  - Your own machine!



```
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();  
    }  
}
```

# 3) Pre/Post-staging

---

- Class: `masterkey.basic.RunJobWithStaging`
- Run without arguments for usage!
- 3) Run some jobs with files. E.g.
  - `/usr/bin/sort` (see `numbers.txt`)
  - `/usr/bin/convert` (see `images` dir for some input)
  - ...



```
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();
    }
}
```

# 4) Jobs with extra Settings

---

- 4a) Certificates
  - Class: `masterkey.basic.RunJobWithCertificate`
  - Requires globus certificate, and local setup
- 4b) Environment
  - Class: `masterkey.basic.RunJobWithEnvironment`
  - Not all adaptors support this



# Using a Certificate

---

```
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]));

    ...
}
```



# Environment variables

---

```
// Create a software description containing the executable name, and
// two files for any output that is generated on stdout or stderr.
SoftwareDescription sd = new SoftwareDescription();
sd.setExecutable(args[1]);
sd.setStdout(GAT.createFile("stdout.txt"));
sd.setStderr(GAT.createFile("stderr.txt"));

Map<String, Object> environment = new HashMap<String, Object>();
environment.put("SOME_SETTING", "some-value");
sd.setEnvironment(environment);
```



# 5) Preferences

---

- Arbitrary additional settings not in API directly
- Simple key-value pairs
- Possible in almost all JavaGAT types (e.g. Files)
- See `javagat.properties` for a list per adaptor
- Specified using `GATContext`



# 5) Preferences

---

```
GATContext context = new GATContext();
context.addPreference("sshpbs.native.flags", "-l num_gpu=1");

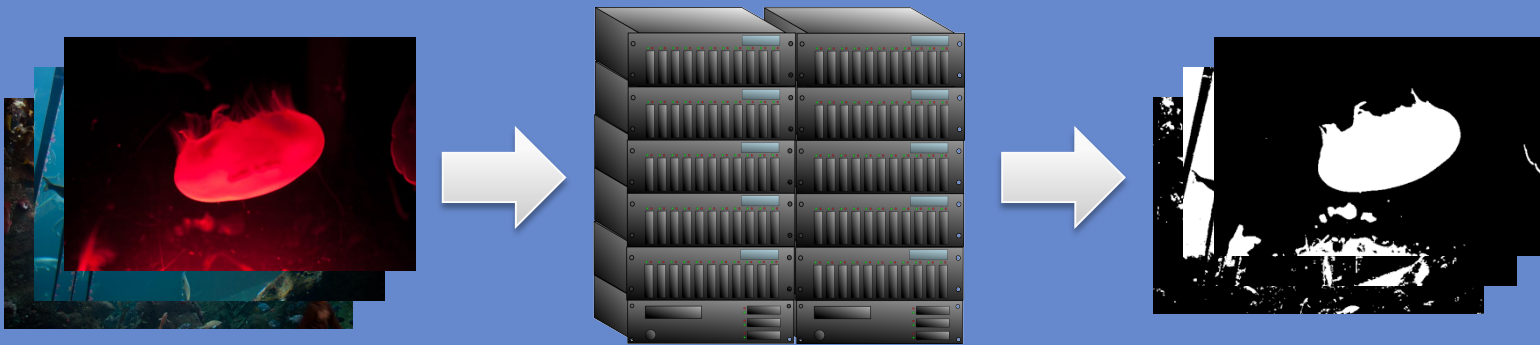
ResourceBroker broker = GAT.createResourceBroker(context, new URI(args[0]));
```

- Class: `masterkey.basic.RunJobWithPreferences`
- Submit to the DAS-4 to get a GPU node!



# Task Farming: Running Example

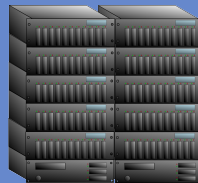
- Scientist needs to process a large number of independent data files using some application
- Offload to compute resource using **Task Farming**
- Today
  - Tool: ImageMagic “convert”
  - Data: Pictures



# 6) Task Farming

---

- Class
  - masterkey.basic.SimpleTaskFarming
  - masterkey.basic.MultiSiteTaskFarming
- 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();  
    }
```

```
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();
    }
}
```