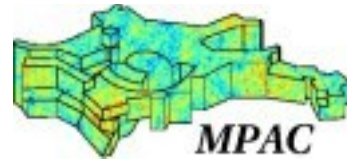


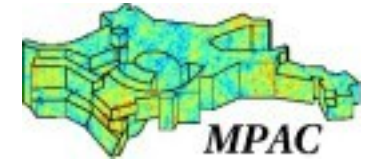
Java PBS GAT Adaptor

Uwe Doerl



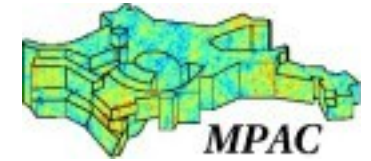
Max-Planck-Institut for Astrophysics

Potsdam 19.1.2006



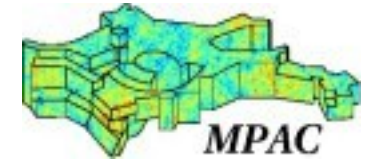
Überblick

- Planck Projekt und Schedule
- Mögliche Bibliotheken
- Struktur des Java GAT Adapter
- Probleme bei der Anbindung an PBS
- Wünsche an die Schnittstelle



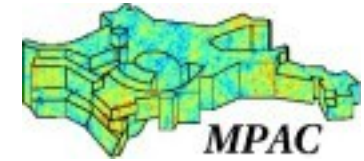
Planck Projekt und Schedule

- Framework zum Verwalten von Pipelines
- Astronomische Berechnungen
- Parallele Ausführung komplexer Berechnungen
- Einsatz verschiedener Scheduler
- mögliche Einbindung in Grids



Mögliche Bibliotheken

- DRMAA – noch sehr unfertig!
- JGrid – noch zu simple!
- GAT – weit fortgeschritten
- Eigene Entwicklung – keine Ressourcen



Struktur des Java GAT Adapter

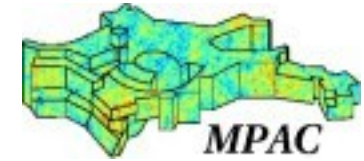
```
public class PbsJob extends Job {
    private PbsBrokerAdaptor mBroker;
    private JobDescription mDescription;
    private String mId;

    public PbsJob(PbsBrokerAdaptor broker, JobDescription description, String id) {
        mBroker = broker;
        mDescription = description;
        mId = id;
        state = INITIAL;
    }

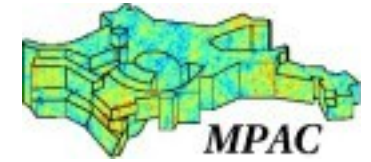
    public Map getInfo() throws GATInvocationException, IOException {
        ...
    }
    ...
}
```



```
public class PbsBrokerAdaptor extends ResourceBrokerCpi implements IParameter {  
    public PbsBrokerAdaptor(GATContext context, Preferences pref) throws GATObjectCreationException {  
        super(context, pref);  
        checkName("pbs");  
    }  
    public Job submitJob(JobDescription description) throws GATInvocationException, IOException {  
        SoftwareDescription sd = description.getSoftwareDescription();  
        ...  
        java.io.File temp = java.io.File.createTempFile("pbs", null);  
        try {  
            ParamWriter job = new ParamWriter(new BufferedWriter(new FileWriter(temp)), SUFFIX);  
            job.println("#!/bin/sh");  
            job.println("# qsub script automatically generated by scheduler");  
            job.addString("v", getEnvironment(sd.getEnvironment()));  
            job.addString("N", temp.getName());  
            ...  
            job.println(cmd.toString());  
            id = Executer.singleResult("/qsub " + temp);  
        }  
        finally { temp.delete(); }  
        return new PbsJob(this, description, id);  
    }  
    ...  
}
```

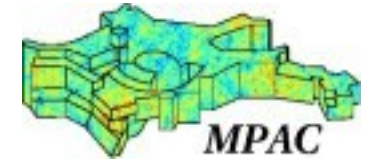


```
public class Executer implements IParameter {
    private static final String sPbsBin;
    private static String[] sExport;
    static {
        Environment env = new Environment();
        Vector exp = new Vector();
        exp.add(PBS_EXEC + "=" + env.getVar(PBS_EXEC));
        exp.add(PBS_HOME + "=" + env.getVar(PBS_HOME));
        exp.add("LD_LIBRARY_PATH=" + env.getVar(PBS_EXEC) + "/lib/");
        setExport(exp);
        sPbsBin = env.getVar(PBS_EXEC) + "/bin/";
    }
    public static String singleResult(String command) {
        String result = null;
        Process proc = Runtime.getRuntime().exec(sPbsBin + command, sExport);
        proc.waitFor();
        if (proc.exitValue() == 0) {
            BufferedReader br = new BufferedReader(new InputStreamReader(proc.getInputStream()));
            result = br.readLine();
            br.close();
        }
        return result;
    }
    ...
}
```



Probleme bei der Anbindung an PBS

- keine Job Datenbank
- kein Exit-Status der Scripte
- File Austausch nur innerhalb des Clusters
- kein stdin



Wünsche an die Schnittstelle

- universellere Software Parameter
- Einbindung der Hardware Parameter
- Reduktion der Zahl der Bibliotheken
- Einfachere Selection der Adapter
- Logging