Autoren: Andreas Hoheisel

E-Mail: andreas.hoheisel@first.fraunhofer.de

Dokumentname: Anforderungen an Grid-Knoten für das GWES-Workflow-Management

Arbeitspakete: AP 2.2 Prozessmanagement

beteiligte Partner: Fraunhofer FIRST

Versionsgeschichte

Version	Datum	Beschreibung	Autor
0.1	01.12.2009	Erster Entwurf	Andreas Hoheisel
0.2	01.12.2009	Glossar hinzugefügt	Andreas Hoheisel
0.3	04.12.2009	Informationen über gwes-command-line-operation.sh und Inhaltsverzeichnis hinzugefügt	Andreas Hoheisel
0.4	10.05.2010	Informationen zu activity-directory-template hinzugefügt und medigrid -> pneumogrid	Andreas Hoheisel
0.5	11.05.2010	Skript zum Ausführen von Kommandozeilenprogrammen verbessert (unterstützt jetzt auch Parameter mit Leerzeichen), siehe Anhang A	Andreas Hoheisel
1.0	27.01.2011	Ergänzungen bezüglich Integration von LSF und GWES Version 2.1 und ResourceUpdater 3.2.0	Andreas Hoheisel

Inhaltsverzeichnis

Anforderungen an Grid-Knoten für das GWES-Workflow-Management

Im Folgenden werden die technischen Anforderungen beschrieben, die notwendig sind, damit eine Grid-Ressource für die Workflow-Ausführung mit dem *Grid Workflow Execution Service* (GWES) Version 2.1 verwendet werden kann. Weiterführende Informationen zum GWES sind unter http://www.gridworkflow.org/gwes zu finden. Die Zielgruppe dieses Textes sind in erster Linie Systemadministratoren, die für die Konfiguration der Grid-Knoten zuständig sind (gemäß der Definition des Begriffs „Grid-Knoten“ im Glossar auf Seite 6).

Diese Anforderungen beziehen sich nicht nur auf das Projekt PneumoGRID, sondern gelten auch für das GWES-Workflow-Management in anderen D-Grid-Projekten, wie zum Beispiel BauVOGrid, MediGRID, MediInfoGrid und Services@MediGRID.

Die Anforderungen an Web-Service-Ressourcen (SOAP-Ressourcen) sowie die Installation von Anwendungs-Software und die Ausführung von Workflows ist nicht Bestandteil dieses Textes.

Anforderungen an Grid-Knoten als Checkliste:

- ☐ **Globus-Toolkit 4.0.x** gemäß D-Grid-Referenzinstallation (siehe http://dgiref.d-grid.de/wiki/Introduction). Insbesondere werden benötigt
  - GSISSH (zur Installation und zum Testen von Anwender-Software)
  - WS-GRAM (zum automatisierten Ausführen von Workflow-Aktivitäten)
  - RFT (zum Übertragen von Dateien)
- ☐ **/etc/grid-security/grid-mapfile**: Die DNs aller Mitglieder der entsprechenden VO müssen in der grid-mapfile auf einzelne lokale Nutzerkonten abgebildet werden (keine Sammelkonten aus Datenschutzgründen).
- ☐ **Firewalleinstellungen**: Zusätzlich zu den offenen Globus-Ports gemäß der D-Grid-Referenzinstallation werden in der Regel folgende offene Ports benötigt:
  - 80, 8080, 9050, 9081 (nur ausgehender Verkehr, z.B. zur Kommunikation zwischen ResourceUpdater und Ressourcen-Registry)
- ☐ **PBS** oder **LSF** als lokales Resource-Management-System, falls die Grid-Ressource Jobs auf einem Rechen-Cluster verteilt. Für das automatische Monitoring der Queues wird der Befehl `qstat -Q` (PBS) bzw. `bqueues` (LSF) verwendet. Für die Unterstützung von weiteren lokalen Resource-Management-Systemen (z.B. SGE, Condor, ...) kann gerne ein Angebot erstellt werden.
- ☐ **Java 5** (Version 1.5.x) oder **Java 6** (Version 1.6.x) – wird vom ResourceUpdater benötigt.

- ☐ **Gemeinsames VO-Verzeichnis für die Installation von Anwendungen (bevorzugt `/opt/pneumogrid`):**
  - Schreibrechte für alle Anwendungsentwickler der VO
  - Leserechte und Ausführrechte für die gesamte VO
  - Verfügbar unter selben Pfad auf dem Grid-Knoten (front end) und auf allen Cluster-Knoten
- ☐ **Gemeinsames VO-Scratch-Verzeichnis (bevorzugt `/opt/pneumogrid/tmp`):** In diesem Verzeichnis wird automatisch für jede Workflow-Aktivität ein Unterverzeichnis angelegt, das dann als temporäres Arbeitsverzeichnis verwendet wird und (Zwischen-) Ergebnisse enthält.
  - Schreibrechte und Leserechte für die gesamte VO
  - Verfügbar unter selben Pfad auf dem Grid-Knoten (front end) und auf allen Cluster-Knoten
  - In der Regel genügen etwa 500GB
- ☐ **Skript zur Ausführung von Kommandozeilenprogrammen:** Der GWES führt die Anwendungsprogramme auf den Grid-Knoten oder Cluster-Knoten nicht direkt aus, sondern verwendet hierfür das dort zu installierende Shellscript `gwes-command-line-operation.sh`. Dies hat den Vorteil, dass eine zusätzliche Protokollierung der Ausführung erfolgen kann und zudem vorbereitende Schritte, wie zum Beispiel die Erstellung eines temporären Arbeitsverzeichnisses, effektiver durchgeführt werden können. In der Regel wird dieses Shellskript vom GWES-Administrator unter `/opt/pneumogrid/tmp/gwes-command-line-operation.sh` installiert und ist in Anhang A auf Seite 7 gelistet.
- ☐ **Leeres Verzeichnis als Muster für Ausführungsverzeichnisse:** Da mit dem File-Stage-In-Mechanismus von WS-GRAM lediglich vorhandene Verzeichnisse kopiert, aber keine neuen Verzeichnisse angelegt werden können, benötigt der GWES ein Muster eines Verzeichnisses, welches er dann per File-Stage-In zum Anlegen von temporären Arbeitsverzeichnissen verwenden kann. Dieses Musterverzeichnis wird in der Regel vom GWES-Administrator als leeres Verzeichnis unter `/opt/pneumogrid/tmp/activity-directory-template/` angelegt.
- ☐ **Aufräumskript als cron-Job**: Um übrig gebliebene Arbeitsverzeichnisse in dem Scratch-Verzeichnis zu löschen, sollte in regelmäßigen Abständen (zum Beispiel einmal pro Woche als cron-Job) ein Aufräumskript ausgeführt werden. In der Regel wird dieses unter `/opt/pneumogrid/cleanup/gwes-cleanup-tmpdirs.sh` installiert. Da die zu löschenden Aktivitätsverzeichnisse unter verschiedenen Nutzerkennungen erstellt wurden, benötigt dieses Skript root-Rechte. Ein Beispiel eines solchen Skriptes ist in Anhang B auf Seite 9 gelistet.
- ☐ **ResourceUpdater Version 3.2.0:** Der GWES verwendet ein Monitoring-System, welches unabhängig von MDS oder anderen Grid-Informationssystemen funktioniert. Hierzu wird das Java-Programm „ResourceUpdater“ als Daemon auf der Grid-Ressource gestartet, der in regelmäßigen Abständen (ca. alle 10 bis 20 Sekunden) aktuelle Ressourceninformationen in der Ressourcen-Registry (D-GRDL-Datenbank) ablegt. Falls die Grid-Ressource sowohl Fork als auch PBS/LSF

unterstützt, müssen zwei Instanzen des RessourceUpdater installiert, konfiguriert und gestartet werden. Der ResourceUpdater wird in der Regel von dem GWES-Administrator automatisch installiert; kann jedoch auch von dem Administrator der Grid-Ressource von http://www.gridworkflow.org/kwfgrid/distributions/ heruntergeladen, installiert und konfiguriert werden.

Glossar

- **Grid-Knoten**: Ein im Grid exponierter Grid-Rechner, der für die Annahme von Grid-Jobs zuständig ist und diese entweder auf dem Grid-Knoten selber ausführt (Fork) oder den Job über ein *lokales Ressourcen-Management-System* (z.B. PBS oder LSF) an dahinter liegende *Cluster-Knoten* verteilt.
- **GWES**: *Grid Workflow Execution Service*. Dienst für die automatische Ausführung von IT-Prozessen auf verteilten Systemen; siehe auch http://www.gridworkflow.org/gwes
- **LRMS**: Lokales Ressourcen-Management-System, wie zum Beispiel PBS, LSF, SGE.
- **PBS**: *Portable Batch System*. Ein LRMS zum Verteilen von Jobs auf einem Rechen-Cluster. Siehe auch http://en.wikipedia.org/wiki/Portable_Batch_System
- **LSF:** *Load Sharing Facility*. Ein kommerzielles LRMS der Firma Platform Computing zum Verteilen von Jobs auf einem Rechen-Cluster. Siehe auch http://en.wikipedia.org/wiki/Platform_LSF
- **ResourceUpdater**: Ein Java-Programm, welches als Daemon auf Grid-Knoten ausgeführt wird, um die Auslastung der Ressourcen zu überwachen. Der ResourceUpdater für PBS ruft in regelmäßigen Abständen `qstat -Q` auf, um Informationen über die Auslastung der Queues zu erhalten; im Falle von LSF wird hierfür der Befehl `bqueues` verwendet. Der ResourceUpdater für Fork nutzt `df`, um Informationen über freien Plattenplatz zu erhalten und das `/proc/`−Dateisystem für Informationen über den aktuellen Arbeitsspeicher und die Netzwerkauslastung. Diese Informationen werden dann als D-GRDL formatiert in die Ressourcen-Datenbank der VO (basierend auf eXist-DB) gespeichert und dienen als Grundlage für das Meta-Scheduling und ResourceMatching. Es werden keine personenbezogenen Daten übertragen. Der ResourceUpdater benötigt keine root-Rechte. Siehe auch http://www.gridworkflow.org/fhrg/resourceupdater/docs/
- **VO**: *Virtuelle Organisation*, siehe auch http://de.wikipedia.org/wiki/Virtuelle_Organisation

Anhang A: `gwes-command-line-operation.sh`

```
#!/bin/bash
#######################################################################
# $Id: gwes-command-line-operation.sh 1372 2010-05-11 09:25:03Z
andreas.hoheisel@first.fraunhofer.de $
#
# Copyright 2010 Fraunhofer Gesellschaft, Munich, Germany,
# for its Fraunhofer Institute for Computer Architecture and Software
# Technology (FIRST), Berlin, Germany. All rights reserved.
# http://www.first.fraunhofer.de/
#
#######################################################################
#
# Script for wrapping command line operations.
# Copy this script to all your WS-GRAM resources to the directory specified
# by the property "gwes.gram.home.directory".
#
# @author Andreas Hoheisel
# @version $Id: gwes-command-line-operation.sh 1372 2010-05-11 09:25:03Z
andreas.hoheisel@first.fraunhofer.de $
#
#######################################################################

function usage {
  echo "### USAGE: $0 --WORKDIR <WORKING DIRECTORY> --EXEC <EXECUTABLE> [--STDOUT <STDOUT-
FN>] [--STDERR <STDERR-FN>] [--STDIN <STDIN-FN>] [<additional parameters>]"
}

function timestamp {
    DATE=`date '+%Y-%m-%d_%H:%M:%S_%N'`
    echo "DATE=${DATE}" >> exec.log
}

INDEX=1
while [ $# -gt 0 ]
do
  case $1 in
    --WORKDIR)
      WORKDIR=$2
      shift 2
      ;;
    --EXEC)
      EXEC=$2
      shift 2
      ;;
    --STDOUT)
      STDOUT=$2
      shift 2
      ;;
    --STDERR)
      STDERR=$2
      shift 2
      ;;
    --STDIN)
      STDIN=$2
      shift 2
      ;;
    *)
      param[${INDEX}]=$1
      let "INDEX += 1"
      shift
      ;;
  esac
done

if [ -z "$WORKDIR" ]; then
  echo "### Error: the working directoy has not been specified! ###"
  usage
  exit 99
fi

if [ -z "$EXEC" ]; then
  echo "### Error: the executable has not been specified! ###"
```

```
  usage
  exit 98
fi

if [ ! -x "$EXEC" ]; then
  echo "### Error: the executable \"$EXEC\" is not available or not executable! ###"
  usage
  exit 97
fi

if [ -z "$STDOUT" ]; then
  STDOUT="stdout.dat"
fi
if [ -z "$STDERR" ]; then
  STDERR="stderr.dat"
fi

mkdir -p $WORKDIR
cd $WORKDIR

timestamp
if [ ! -z "$STDIN" ]; then
  if [ ! -r "$STDIN" ]; then
    echo "### Error: the stdin \"$STDIN\" is not available or not readable! ###"
    usage
    exit 96
  fi
  # with stdin from file
  echo "COMMANDLINE=${EXEC} ${param[@]} > $STDOUT 2> $STDERR < $STDIN" >> exec.log
  ${EXEC} "${param[@]}" > "$STDOUT" 2> "$STDERR" < "$STDIN"
  echo "EXITCODE=$?" >> exec.log
else
  # without stdin from file
  echo "COMMANDLINE=${EXEC} ${param[@]} > $STDOUT 2> $STDERR" >> exec.log
  ${EXEC} "${param[@]}" > "$STDOUT" 2> "$STDERR"
  EXITCODE=$?
  echo "EXITCODE=${EXITCODE}" >> exec.log
fi
timestamp
exit ${EXITCODE}
```

Anhang B: `gwes-cleanup-tmpdirs.sh`

```
#!/bin/bash
# Script for removing old GWES working directories.
set -e
LOGDIR=/opt/pneumogrid/tmp/logs
LOG=${LOGDIR}/cleanup.log
TMPDIR=/opt/pneumogrid/tmp
DAYS=90
DATE=`date '+%Y-%m-%d_%H:%M:%S_%N'`
if [ ! -d "$LOGDIR" ]; then
    mkdir $LOGDIR
fi
echo "begin removing tmp directories ($DATE) ..." >> $LOG
find ${TMPDIR}/ -maxdepth 1 -type d -mtime +$DAYS \
  -name "*_????????-????-????-????-???????????_??????????" \
  -exec rm -vrf {} \; 2>> $LOG >> $LOG
echo "end removing tmp directories ($DATE)." >> $LOG
echo "--------------------------------------------------------------------------------" >> $LOG
```