GlideinWMS Validation scripts

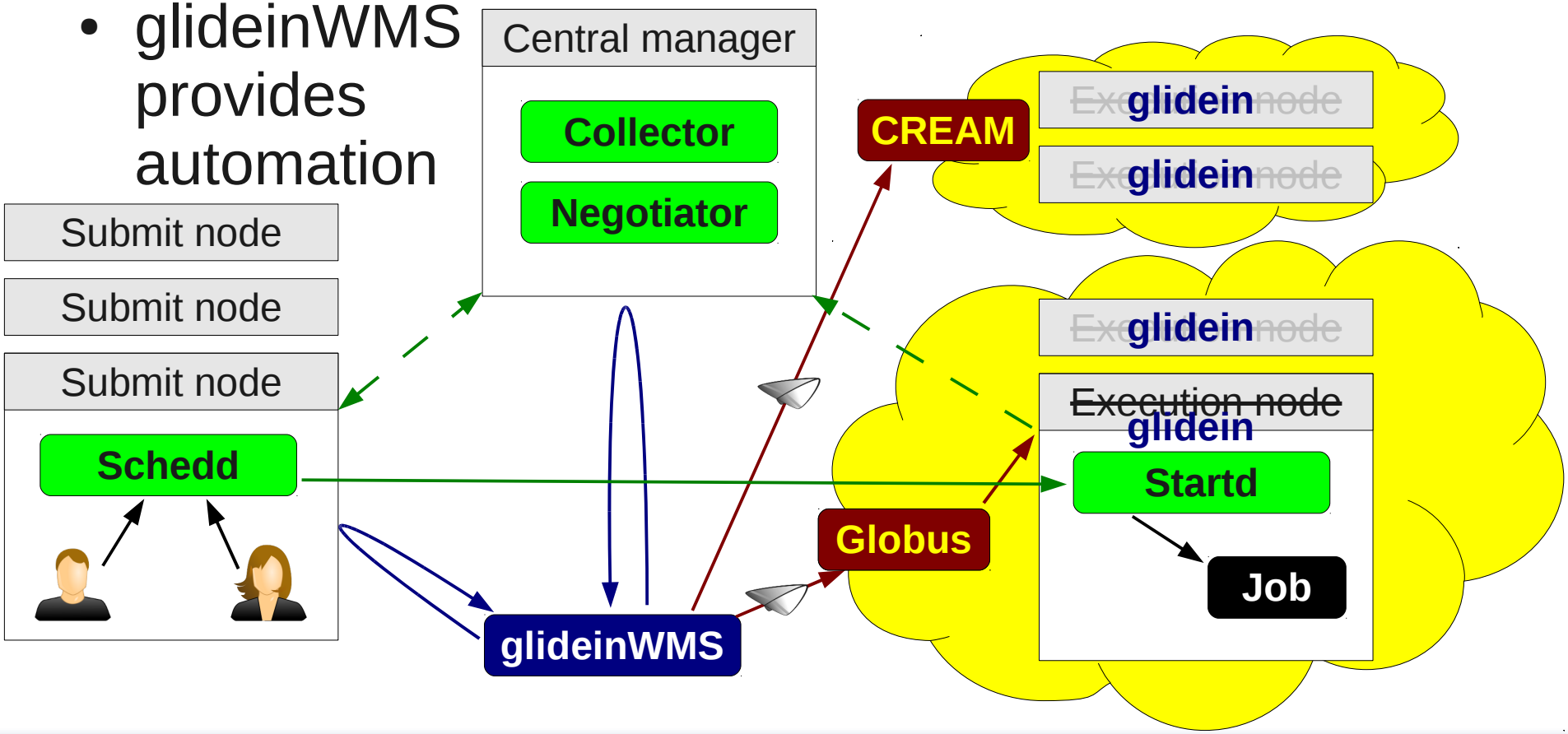
by Igor Sfiligoi (UCSD)

Overview

- Why validation scripts
- Anatomy of validation scripts
- Types of validation scripts

Reminder - Glideins

- A glidein is just a properly configured Condor execution node submitted as a Grid job
  - glideinWMS provides automation



Reminder – Glidein script

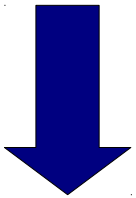
- Glidein startup script just a empty shell that:
  - Downloads scripts, parameters and Condor bins
  - Runs the scripts in order
  - Does the final cleanup
- Two types of script:
  - Node validation
  - Condor configuration and startup

If any of these fail,
Condor will never be started

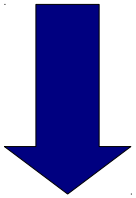
Once Condor starts,
glideinWMS is out of the way

As a consequence

If validation scripts finds a bad WN



Condor will not be started



No user jobs will ever fail here

Is validating at glidein startup a good idea?

- Advantages:
  - User jobs never land on “broken” nodes
  - Failures logged
- Limitations:
  - Tested only at glidein startup
    - If node “goes bad” after Condor startup, user jobs will still be fetched and will fail
- Problems:
  - Failed validation → wasted CPU
    - Some jobs may still succeed, even if validation failed

Users happy

Factory admins can act on this info, notifying sites (who can fix the problem)

Condor provides cron-like capabilities for this

Can be solved by passing the test and setting attributes

But this will hide problem from Factory

Anatomy of a validation script

Validation scripts 101

- Any executable will do!
  - There are no restrictions
  - Can be compiled binary or a shell script
- Exit code checked
  - ==0 - Success
  - !=0 - Failure
- And, to the first approximation, this is all

Validation scripts - I/O

- You may want to:
  - Get some input
  - Have some output
- Both handled through a dashboard file
  - Filename passed as the only argument to the validation scripts

Dashboard file

- Simple list of (key, value) pairs
  - One per line ← Newline not allowed in either key or value
  - Space separated ← Space not allowed in the key
- Hash (#) can be used for comments

```
GLIDEIN_Factory UCSD
GLIDEIN_Name Production_v4_2
GLIDEIN_Entry_Name CMS_T2_US_UCSD_gw2
GLIDECLIENT_Name UCSD-v5_3.main
GLIDEIN_WORK_DIR /data10/condor_local/execute/dir_22668/glide_B22745/main
GLIDEIN_Glexec_Use OPTIONAL
X509_CERT_DIR            /wn-client/globus/TRUSTED_CA
GLIDEIN_Site UCSD
# This was calculated on the fly
CCB_ADDRESS glidein-collector.t2.ucsd.edu:9822
```

http://tinyurl.com/glideinWMS/doc.prd/factory/custom_scripts.html#glidein_config

Reading input

- Dashboard file as the first argument
- Then just look for the key and split on space

```
# here is my dashboard file
glidein_config=$1

# I expect only one key and no space in the value
glexec_bin=`awk '/^GLEXEC_BIN /{print $2}'  $glidein_config`
if [ -z "$glexec_bin" ]; then
    exit 1
fi
…
exit 0
```

Writing output

- You can just append to the file
  - Just make sure it is properly formatted

```
# here is my dashboard file
glidein_config=$1


…
# tell condor to use glexec
echo 'GLEXEC_JOB True' >> $glidein_config

exit 0
```

- You should also make sure the same key is not already defined

Helper function

- glideinWMS provides a helper BASH function to avoid duplicate keys
  - External SH file, referenced as **`ADD_CONFIG_LINE_SOURCE`**
  - The function name inside is **`add_config_line`**

```
# here is my dashboard file (MUST be called glidein_config)
glidein_config=$1
# get helper function
add_config_line_source= \
   `awk '/^ADD_CONFIG_LINE_SOURCE /{print $2}' $glidein_config`
source $add_config_line_source
…
# tell condor to use glexec
add_config_line 'GLEXEC_JOB' 'True'
```

Influencing Condor behavior

- By default, keys in dashboard file ignored by Condor startup/configuration script
  - Anything you write into it, it is just for your consumption (e.g. for other scripts of yours)
- A special whitelist file lists the keys that should be passed to Condor
  - Referenced as **`CONDOR_VARS_FILE`**
  - Helper function available **`add_condor_vars_line`**

Again, source **ADD_CONFIG_LINE_SOURCE**

Condor Vars file

- Each line contains a key
- Seven fields, space (or tab) separated
  - Key
  - Type - I – Integer, S – String, C – Expr.
  - Default value - “-” for no default
  - Condor Name - “+” = Key name
  - Is it required? - Y|N
  - Should be exported to ClassAd? - Y|N
  - Should be exported to job environment?
    - “-” no, “+” Key name, “@” Condor Name

Useful when others have to define it

http://tinyurl.com/glideinWMS/doc.prd/factory/custom_scripts.html#condor_vars

Example

```
# here is my dashboard file (MUST be called glidein_config)
glidein_config=$1
# extract where to find the vars file
# (MUST be called condor_vars_file)
condor_vars_file= \
   `awk '/^CONDOR_VARS_FILE /{print $2}' $glidein_config`
# get helper function
add_config_line_source= \
   `awk '/^ADD_CONFIG_LINE_SOURCE /{print $2}' $glidein_config`
source $add_config_line_source
…
# This should already have been set
add_condor_vars_line "GLEXEC_BIN" "C" "-" "GLEXEC" "Y" "N" "-"
# tell condor to use glexec
add_config_line 'GLEXEC_JOB' 'True'
add_condor_vars_line "GLEXEC_JOB" "C" "True" "+" "Y" "Y" "-"
# tell user where is the TMPDIR
add_config_line 'GLEXEC_TMP' $TMPDIR
add_condor_vars_line "GLEXEC_TMP" "S" "-" "+" "Y" "Y" "+"
```

Error messages

- Your script found a problem
  - Now what?
- You definitely want to exit with errno !=0
- **But, please, also print an error message!**
  - With enough information to understand **why** the script failed
  - Will allow the Factory admins to act on it

Planned improvements

(still speculation at this point)

- Current error codes and messages arbitrary
  - Mostly good enough for manual debugging
  - But cannot really automatically act on them
- Want to add some more structure
  - Based on OSG Common Output Format proposal
    https://twiki.grid.iu.edu/bin/view/SoftwareTools/CommonTestFormat#Alain_s_proposal_Version_4_evolu
- In addition to exit code, scripts expected to write a status file
  - Which will be read and interpreted by the caller and propagated to the Factory

If file not present, will assume "Error unknown"

Now we can start thinking about automatically acting on errors!

Standardized error reasons

(preliminary - still speculation at this point)

- To allow for automated feedback, need standardized error reasons
- This is what I currently envision:
  - Config - e.g. Impossible combinations
  - Corruption - e.g. SHA1 check failed
  - WN Resource - e.g. Disk full or glexec not found
  - Network - e.g. Cannot talk to VO Collector
  - VO Proxy - e.g. Proxy too short
  - VO Data - e.g. VO SW not installed

Examples

(preliminary - still speculation at this point)

```
<?xml version="1.0"?>
<OSGTestResult id="glideinWMS.check_disk" version="7.5.4">

  <result>
    <status>OK</status>
    <metric name="diskspace" ts="2012-01-12T15:02:20"
            uri="local">/tmp/glidein_15432/</metric>
  </result>
  <detail>Enough disk space found.</detail>
</OSGTestResult>
```

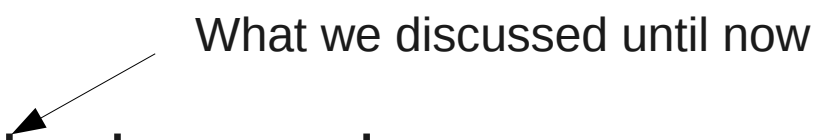
```
<?xml version="1.0"?>
<OSGTestResult id="glideinWMS.check_proxy" version="7.5.4">

  <result>
    <status>FAILED</status>
    <metric name="failure" ts="..." uri="local">VO Proxy</metric>
    <metric name="proxy" ts="2012-01-12T15:02:21"
            uri="local">/tmp/glidein_15432/proxy/a.proxy</metric>
  </result>
  <detail>Proxy had less than 12h left.</detail>
</OSGTestResult>
```

Validation script types

Why should you use VS?

- Of course:
  - Check for obviously broken nodes
- But also:
  - To discover and advertise dynamic information
  - Non-trivial configuration
  - Site-specific customizations



Dynamic information

- Some information dynamic by nature
  - E.g. location of VO software
- You want to discover at run-time where it is located
  - And fail, if you cannot find it!
  - Makes life easier for the users
- Once discovered, good practice to advertise it
  - In either/both the ClassAd and/or job environment

Example

```
# check if CMSSW installed locally
if [ -f "$CMSSW" ]; then
   source "$CMSSW"
   If [ -z “$CMSSW_LIST” -o -z "$CMSSW_LOC" ]; then
      echo "Corrupted CMSSW at $CMSSW!\n" 1>&2
      exit 1
   fi
else
   echo "CMSSW not found!\n" 1>&2
   exit 1
fi

# publish to user job env
add_config_line "CMSSW_LOC" "$CMSSW_LOC"
add_condor_vars_line "CMSSW_LOC" "S" "-"  "+" "Y" "N" "+"
# publish to Condor
add_config_line "CMSSW_LIST" "$CMSSW_LIST"
add_condor_vars_line "CMSSW_LIST" "S" "-"  "+" "Y" "Y" "-"
exit 0
```

Non-trivial configuration

(Not really a “validation” script)

- You may want to generate some data on the fly
  - e.g. a random seed

```
let s=$RANDOM%123+17
add_config_line "MY_SEED" “$s”
add_condor_vars_line "MY_SEED" "I" "-"  "+" "Y" "N" "+"
```

- And sometimes it is just inconvenient to specify some values in the frontend XML file
  - e.g a long list

```
l="1"
for ((i=2; $i<100; i++)); do
  l="$l:$i"
done
add_config_line "MY_LIST" “$l”
add_condor_vars_line "MY_LIST" "S" "-"  "+" "Y" "N" "+"
```

Site specific customization

- Currently, the frontend XML file does not allow site-specific customizations
  - Unless you want to have a group per site!

Limiting, since only one level of groups

  - And there is the option for you to arrange for the Factory to provide it for you

Maintenance will be a mess

- You can code the per-site config into a “validation script”

Still not ideal, but may be better than the alternative

Especially, if you can apply a rule with few exceptions

Example

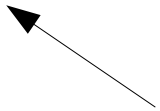
```
glidein_config=$1
site=`awk '/^GLIDEIN_CMSSITE /{print $2}'  $glidein_config`
country=`echo $site| awk '{print substr($1,8,2)}'`
if [ "$country" == "US" ]; then
  myvar="OSG"
elif [ "$country" == "IT" -o "$country" == "FR" ]; then
  myvar="EGI"
else
  echo "Cannot run in $country" 1>&2
  exit 1
fi

add_config_line "MY_VAR" "$myvar"
add_condor_vars_line "MY_VAR" "I" "-"  "+" "Y" "N" "+"
```

Limitations

Limits of validation scripts

- Whatever is discovered on the WN is
  - Used by the script for its own testing
  - At best, propagated to glidein ClassAd or job env
- The discovered info cannot be used for Frontend matchmaking purposes!
  - At best, for Negotiator matchmaking
- As a result, you may be requesting glideins that will never run any user jobs
  - Wither fail validation or do not match



If condition common to all WNs

What can you do?

- How do you notice it?
  - If validation errors
    - The Factory admins will likely contact you
  - If Negotiator not matching jobs
    - You will need to discover it yourself
- What to do afterwards?
  - Tune the script
  - Manually blacklist a site is your frontend XML
  - Or convince the Factory admins to advertise VO specific info

Maybe you were just too aggressive?

Pretty much a hack!

Can be hard to maintain long term

The End

Pointers

- The official glideinWMS project Web page is http://tinyurl.com/glideinWMS
- glideinWMS development team is reachable at glideinwms-support@fnal.gov
- The OSG glidein factory is reachable at osg-gfactory-support@physics.ucsd.edu

Acknowledgments

- The glideinWMS is a CMS-led project developed mostly at FNAL, with contributions from UCSD and ISI
- The glideinWMS factory operations at UCSD is sponsored by OSG
- The funding comes from NSF, DOE and the UC system