ASL

Project Intro + Bash Tutorial

Organization

- Tutorials on Tuesdays
- Project Q&A sessions on Thursdays
  - Starting from next week
- Mapping Student -> TA is published on course web page - find your TA there

Dryad Cluster

- 16 machines (dryad01 – dryad16)
- Use it for testing
  - Shared for all students
- You can connect anytime with ssh

The Project

- No Team-Work
- Use
  - Java
  - PostgreSQL
  - Ant (no Eclipse/Netbeans/IntelliJ project files for submission)
- Not allowed are:
  - Use of external libraries (except PostgreSQL jdbc driver and log4j)
  - Running anything else on the DB-machine than PostgreSQL (no Java-Proxy)

Testing & Scripting

- Your end product is not allowed to use anything but JDK+PostgreSQL
- BUT: you are free to use any other library/framework for testing and benchmarking
- E.g. you can use JUnit for testing and bash scripts for automation

Your Infrastructure

- Your private machine (or ETH machines in the PC rooms), for development and simple testing
- Dryad cluster provided by the Systems Group for *distributed* testing
- The Amazon Computing Cloud for benchmarks

Bash Tutorial

(darkoma@inf.ethz.ch)

Comments from previous years

- “I had to stay up all night to run experiments!”
- “I cannot work on this from home...”
- “It took us more than 40hr./week to work on this milestone.”
- Solution is simple

Automate the experiments!

Bash Basics

- Bash is the command line interface used by most Linux systems
- On Windows: Use Putty to connect to a remote machine (Cluster and/or Amazon)
- Most Bash-commands are simple programs that are executed
- Recommended for automating experiments
  - Feel free to use other scripting languages (Python, ...)

Running Bash Commands

```
command arg1 arg2
```

```
   # starts a command prompt returns as soon as the command finished
```

```
command arg1 arg2 &
```

```
   # runs the command in the background
```

```
command arg1 arg2 > out
```

```
   # runs the command, write standard out to file called "out"
```

```
command arg1 arg2 | command2 arg
```

```
   # runs the command and "pipes" the output into command2
```

Conditionals in Bash

```
MYVAR=foo
# set variable MYVAR to foo
export MYVAR=bar
# set MYVAR, child processes will see it as well
if [ $MYVAR = foo ]
then
  echo "MYVAR is foo"
elif [ $MYVAR = bar]
then
  echo "Bar"
else
  echo "Somethingisodd"
fi
```

Loops in Bash

```
for i in $(ls)
do
  echo item:$i
done

for i in `seq 1 10`
do
  echo $i
done

COUNTER=0
while [ $COUNTER -lt 10]
do
  echo The counter is $COUNTER
  let COUNTER=COUNTER+1
done
```

Running across multiple machines

- Use SSH and SCP
  - `ssh darkoma@optimus.ethz.ch`
    - Login to machine called optimus.ethz.ch as user darkoma
  - `ssh darkoma@optimus.ethz.ch "ls"`
    - Execute command "ls" on optimus as user darkoma
    - This is great for scripting!!!
  - `scp project.jar darkoma@optimus.ethz.ch:~/asl`
    - Copy project.jar to ~/asl on optimus
- Hints:
  - To use ssh and scp in scripts use public key for passwordless authentication

Enabling Passwordless Login

- To enable passwordless login on local machine:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- To enable passwordless login on remote machine:

```
ssh-copy-id <username>@<machine>
```

- If there is no id_rsa.pub:

```
ssh-keygen
```

- And don't give a password

Running experiments while away

- What if my VPN connection fails while running the experiments
  - Use `screen` [1]
- Bad 1:

  darkoma@localmachine: ~$ ./runExperiment.sh
  - Connection is lost → experiment fails
- Bad 2:

  darkoma@localmachine: ~$ ssh remotemachine

  darkoma@localmachine: ~$ ./runExperiment.sh
  - Connection is lost → experiment fails

  Good:

  darkoma@localmachine: ~$ ssh remotemachine

  darkoma@remotemachine: ~$ screen -S experiment

  darkoma@remotemachien: ~$ ./runExperiment.sh
  - Connection is lost:

  darkoma@localmachine: ~$ ssh remotemachine

  darkoma@remotemachine: ~$ screen -dR experiment

[1] http://www.gnu.org/software/screen/manual/screen.html#Overview

Bash Demo

- Sample script that:
  1. Checks if passwordless login to server and client is working
  2. Copies jar file to server and client machines
  3. Runs server and waits for it to start listening to connections
  4. Starts clients on client machines
  5. Waits for clients to finish
  6. Copies log files from client machine
  7. Deletes log files from client and server machines
  8. Processes log files
  9. Plots the result with gnuplot
- Will be made available on the course web page

Compiling and running PostgreSQL

Compiling PostgreSQL

```
mkdir /mnt/local/username
# create a new directory
# download postgresql source and copy it with scp to the remote machine
tar xjf postgresql-9.3.5.tar.bz2
cd postgresql-9.3.5/
./configure --prefix="/mnt/local/username/postgres"
# the prefix tells the system where you want to install postgres
make
# this compiles postgresql - it might take a while
make install
# install it
```

Running PostgreSQL

```
LD_LIBRARY_PATH=/mnt/local/username/postgres/lib

export LD_LIBRARY_PATH

# make sure Linux can find the required shared libraries

/mnt/local/username/postgres/bin/initdb -D /mnt/local/username/postgres/db/

# this will create a newdatabase

# make sure to create it on a local disk and NOT in NFS (not your home directory)

/mnt/local/username/postgres/bin/postgres -D /mnt/local/username/postgres/db/ -p
PORTNUMBER -i -k /mnt/local/username/

# PORTNUMBER should be a random number bigger than 1024 - it might fail if another user
runs postgres on that port already - just use another port number

# The - k parameter lets postgres write the Unix socket into another directory (default is
/tmp)

# if this is not set, postgres might fail because another user wrote a unix socket
already.
```