ECE 454/750: Distributed Computing
Tutorial 3: Assignment 1

TA: Hua Fan

h27fan@uwaterloo.ca

May 26, 2015 05:30-06:20, QNC 2502

Outline

In this tutorial we will discuss:

- Public Key Authentication (windows, linux)
- Set CPU affinity
- Submission Instructions
- Design document
- Build Instructions
- Sample test case
- Q & A

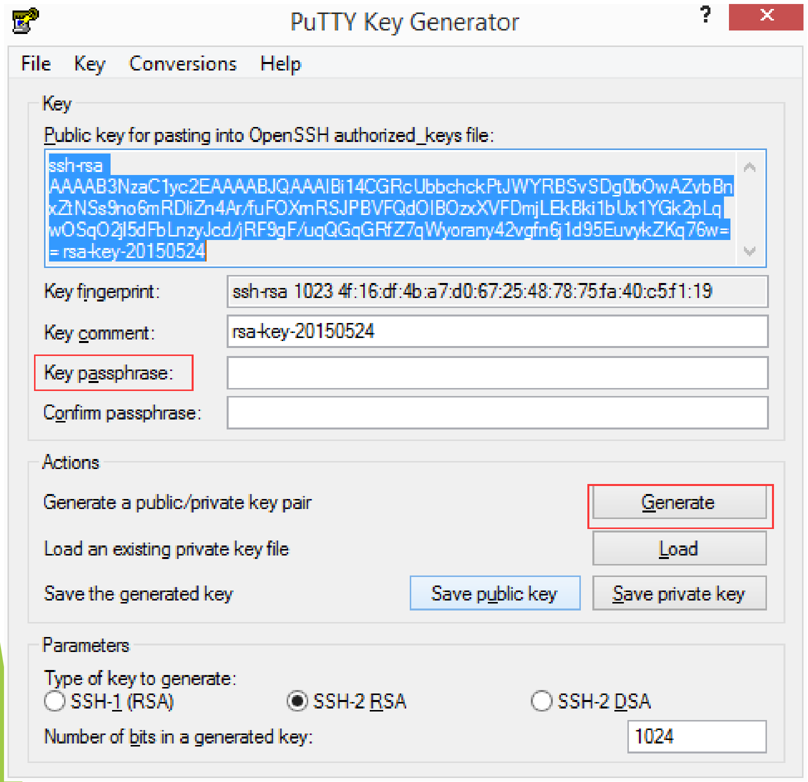
- Slides will be uploaded in Learn later.
- Discussion forum: piazza.com/uwaterloo.ca/summer2015/ece454750

Public Key Authentication With PuTTY*

- No need to send passwd to the server you want to ssh.
- Send public key to remote server; the local machine stores both the public key and the private
- Step 1: generate public/private key pair by PuTTYgen
  - Click Generate button
  - Save private key by button
  - Save public key in blue block in the following picture(highlight with the mouse)

* material derived from: http://www.ualberta.ca/CNS/RESEARCH/LinuxClusters/pka-putty.html

Step 1: generate public/private key pair



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIBi14CGRcUbbchckPtJWYRBSvSDg0bOwAZvbBn
xZtNSs9no6mRDliZn4Ar/fuFOXmRSJPBVFQdOIBOzxXVFDmjLEkBki1bUx1YGk2pLq
wOSqO2jl5dFbLnzyJcd/jRF9gF/uqQGqGRfZ7qWyorany42vqfn6j1d95EuvykZKq76w=
= rsa-key-20150524
```

Key fingerprint: ssh-rsa 1023 4f:16:df:4b:a7:d0:67:25:48:78:75:fa:40:c5:f1:19

Key comment: rsa-key-20150524

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

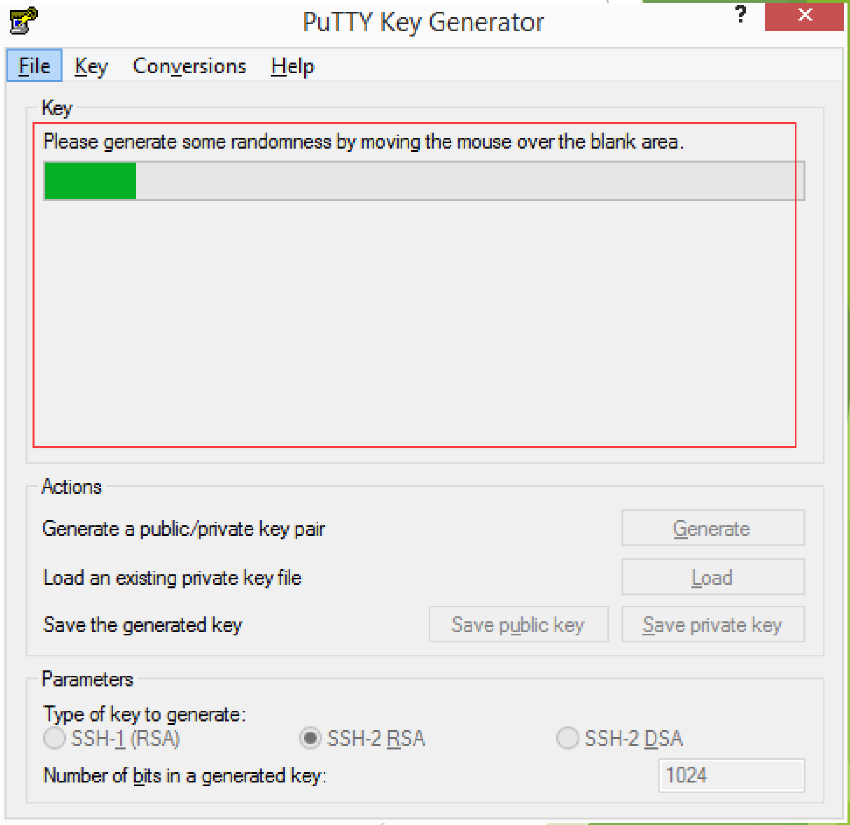
Load an existing private key file

Save the generated key

Parameters

Type of key to generate: SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 1024



Key

Please generate some randomness by moving the mouse over the blank area.

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

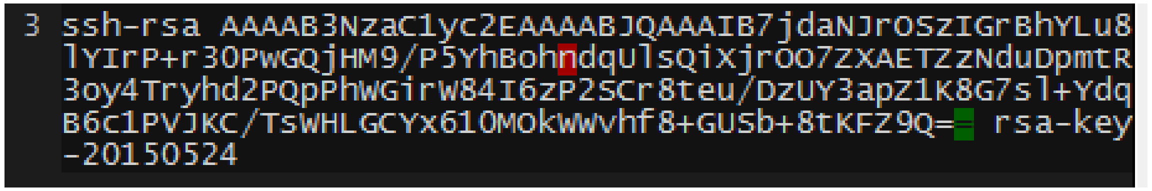
Parameters

Type of key to generate: SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 1024

Step 2: install the public key on the remote host

- Append the context of public key to file ~/.ssh/authorized_keys on the remote host(linux)
- One way: edit that file by text editor

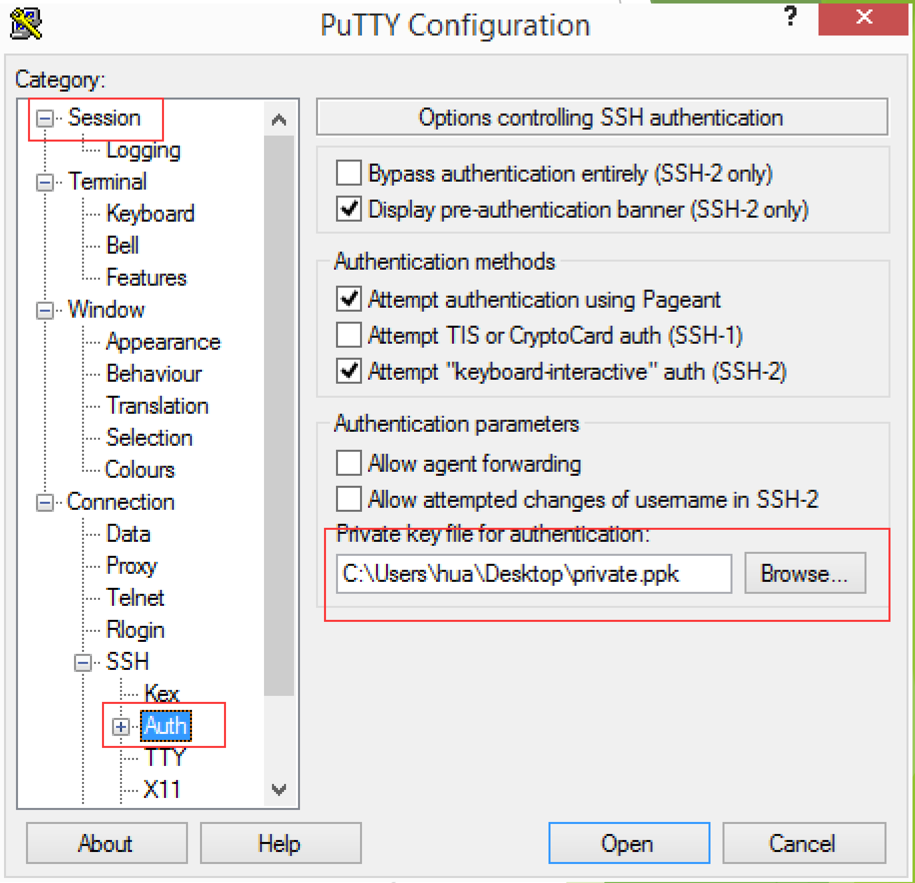


```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAIB7jdaNJrOSzIGrBhYLu8
lYIrP+r3OPwGQjHM9/P5YhBohndqUlsQixjrOO7ZXAETZzNduDpmtR
3oy4Tryhd2PQpPhwGirW84I6zP2SCr8teu/DzUY3apZ1K8G7sl+Ydq
B6c1PVJKC/TsWHLGCYx61OMOkWWvhf8+GUSb+8tKFZ9Q== rsa-key
-20150524
```

- Another way: transfer the public key file to remote host
  - cat public.key >> ~/.ssh/authorized_keys

Step 3: Setting private key for a session

- Connection->SSH->Auth:
  - Select the private key file.
- Save session: session->Save



Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

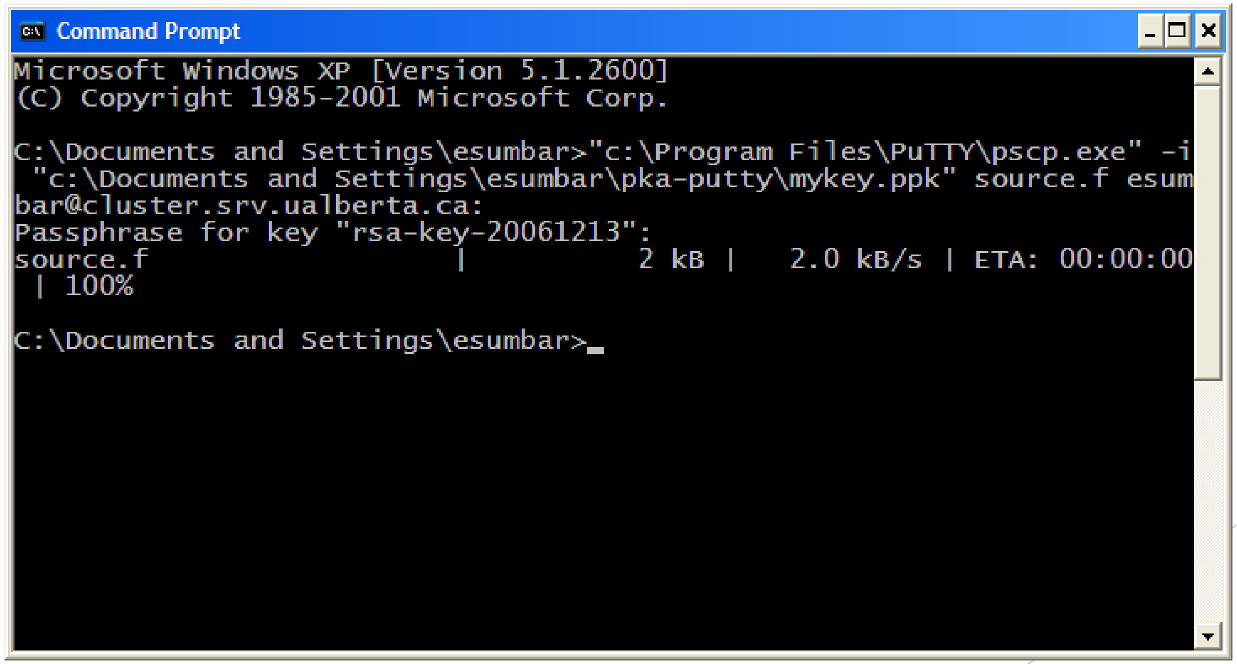
- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:

C:\Users\hua\Desktop\private.ppk

Additional usage

- File transfers with key authentication
  - pscp.exe –i flag to specify private key.
  - Usage(similar to scp on linux): pscp.exe –i private.key file user@host:path

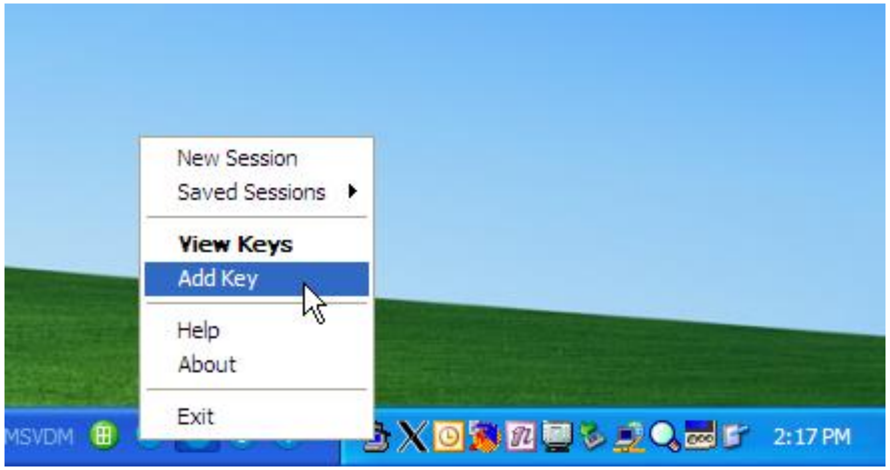


```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\esumbar>"c:\Program Files\PuTTY\pscp.exe" -i
 "c:\Documents and Settings\esumbar\pka-putty\mykey.ppk" source.f esum
bar@cluster.srv.ualberta.ca:
Passphrase for key "rsa-key-20061213":
source.f                  |          2 kB |   2.0 kB/s | ETA: 00:00:00
 | 100%

C:\Documents and Settings\esumbar>_
```

* Figure from: http://www.ualberta.ca/CNS/RESEARCH/LinuxClusters/pka-putty.html

Additional usage

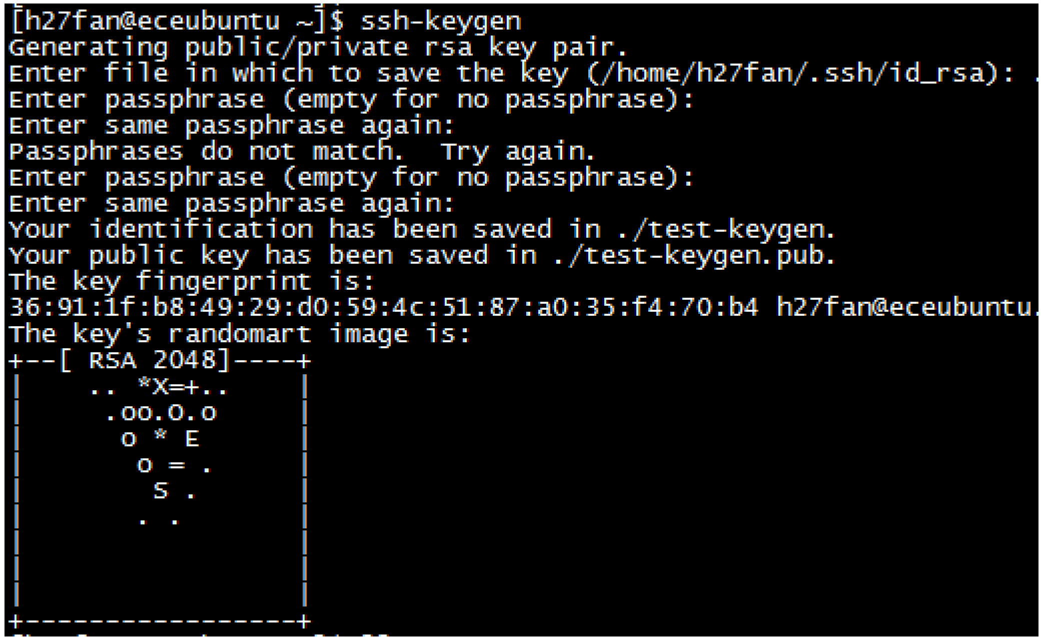
- PuTTY authentication agent, pageant
  - If you don’t want type passphrase every time.
  - Right-click on the icon and choose "Add Key" as illustrated below.



* Figure from: http://www.ualberta.ca/CNS/RESEARCH/LinuxClusters/pka-putty.html

Public Key Authentication With ssh-keygen (Linux)

- Step 1: generate key pair, execute: ssh-keygen



```
[h27fan@eceubuntu ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/h27fan/.ssh/id_rsa): .
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match.  Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./test-keygen.
Your public key has been saved in ./test-keygen.pub.
The key fingerprint is:
36:91:1f:b8:49:29:d0:59:4c:51:87:a0:35:f4:70:b4 h27fan@eceubuntu.
The key's randomart image is:
+--[ RSA 2048]----+
|    .. *X=+..    |
|     .oo.O.o     |
|      o * E      |
|       o = .     |
|        S .      |
|       . .       |
|                 |
|                 |
|                 |
+-----------------+
```

Linux ssh authen. (cont.)

- Step 2: install public key to remote host ~/.ssh/authorized_keys (the same as before)
- Step 3: Place private key in local machine ~/.ssh/
  - If use the default path to save keys, there on need for this step
- Step 4: chmod 600, so no other users can read them
  - chmod 600 ~/.ssh/id_rsa*
  - chmod 600 ~/.ssh/authorized_keys

Set CPU affinity by taskset

- Why use multi-core machines?
  - Parallel execution on multi-cores by multi-threading programing
  - Physic speed limit of CPU cores.
  - Hardware support: eceubuntu 6 cores; ecelinux1 8 cores
- Why need CPU affinity?
  - Reduce cache miss
  - Establish hardware resource bound
- Taskset
  - *taskset –c cpu-list <application>*
  - *e.g. taskset –c 0,1,2,3 <application>*

Submission Instructions

- You should place all you files in one folder, called a1ece454750. The folder should contain:
  - build.xml
  - design document in pdf file.
  - All source code(java, thrift files) organized at your own choice, but files built by an **ant** command.
- You should tar and gzip the folder by:
  - tar czf a1ece454750.tar.gz a1ece454750
  - we will untar and gunzip by: tar xzf a1ece454750.tar.gz
- Don’t submit any .class files or gen-java/*.java files, which can be generated by your source code.

Design document

- Short : 1-page
- The reasoning of your choice of sync/async RPCs, multi-threading servers, and protocols.
- Please be consistent with you choice on all function implementations(sync/async, protocols), otherwise it will be difficult for us to test.
- How do you achieve load balancing?
- How do you detect and deal with crash failures?
- How do you ensure that a BE node that joins the cluster can receive requests from the FE layer within 1 second of startup?

Build Instructions

- cd a1ece454750, where build.xml placed in, and build your solution by executing: ant
- Build must jar the class files and auxiliary resources and output as ece454750s15a1.jar. The jar file must place in the folder a1ece454750. Refer how tutorial.jar is generated in tutorial 1&2.
- You can safely have these assumptions:
  - Build on eceubuntu
  - Thrift compiler 0.9.1 located in: /usr/bin/thrift
  - There is a folder, lib, sharing the same parent folder with a1ece454750. The lib folder (referred as ../lib/ in build.xml) contains:
    - libthrift-0.9.1.jar
    - jbcrypt.jar
    - commons-codec-1.6.jar commons-logging-1.1.1.jar httpcore-4.2.4.jar log4j-1.2.14.jar slf4j-api-1.5.8.jar commons-lang3-3.1.jar httpclient-4.2.5.jar junit-4.4.jar servlet-api-2.5.jar slf4j-log4j12-1.5.8.jar (from thrift-0.9.1/lib/java/build/lib)
    - Other libs/source code are not allowed unless approved by course instructor and TA.

Test runnable

- Execute under folder a1ece454750 : cd a1ece454750
- Test on both ecelinux and eceubuntu. Instructions for cross-compiling to java 1.6 were given in tutorial 1.
- java -cp "ece454750s15a1.jar:../lib/*" ece454750s15a1.FEServer -host ecelinux1 –pport 8123 –mport 9123 –ncores 2 -seeds ecelinux1:10123,ecelinux2:10123,ecelinux3:10123
- The port number for the seed corresponds to the management interface.

Some notes on testing

- We will use our own client for testing, and this client will know the host names and port numbers of the FE nodes. In other words, students do not have to implement a mechanism by which clients would discover the FE nodes.
- We will implement different kinds of clients and detect your RPC mode(sync or async) and protocol choice by them.
- Some FE nodes will be started as seeds, meaning that as seed: the host and management port of the FE node appear in the list of seed nodes.
- Every FE and BE node is given the same list of seed nodes.

Sample test case

- Basic:
  - 1. Start a FE as seed on ecelinuxX taking one core
  - 2. Start a BE on ecelinuxX taking another core
  - 3. Test each interface, expecting results as specification.
  - 4. Start another non-seed FE "on the fly" on ecelinuxY
  - 5. sleep 1 second, new FE should be able to forward requests to the BE node.
  - 6. Test step 3 again against FE on ecelinuxY
- Note
  - X,Y stands for 1, 2, 3, 5
  - Test on servers as early as possible. Don't leave it to the last day. There are many students in the class and the infrastructure will become very busy shortly before the deadline.
  - Avoid testing your code using eceubuntu to prevent overload, use ecelinux machines instead.

Sample: balanced load

- Start a FE as seed on ecelinux1 taking 4 cores
- Start a BE on ecelinux2 taking 2 cores
- Start another BE on ecelinux2 taking 2 cores
- Client send requests
- Request performance counters on the servers
- Expecting: load is roughly split in half between the two BE nodes

Sample: imbalanced load

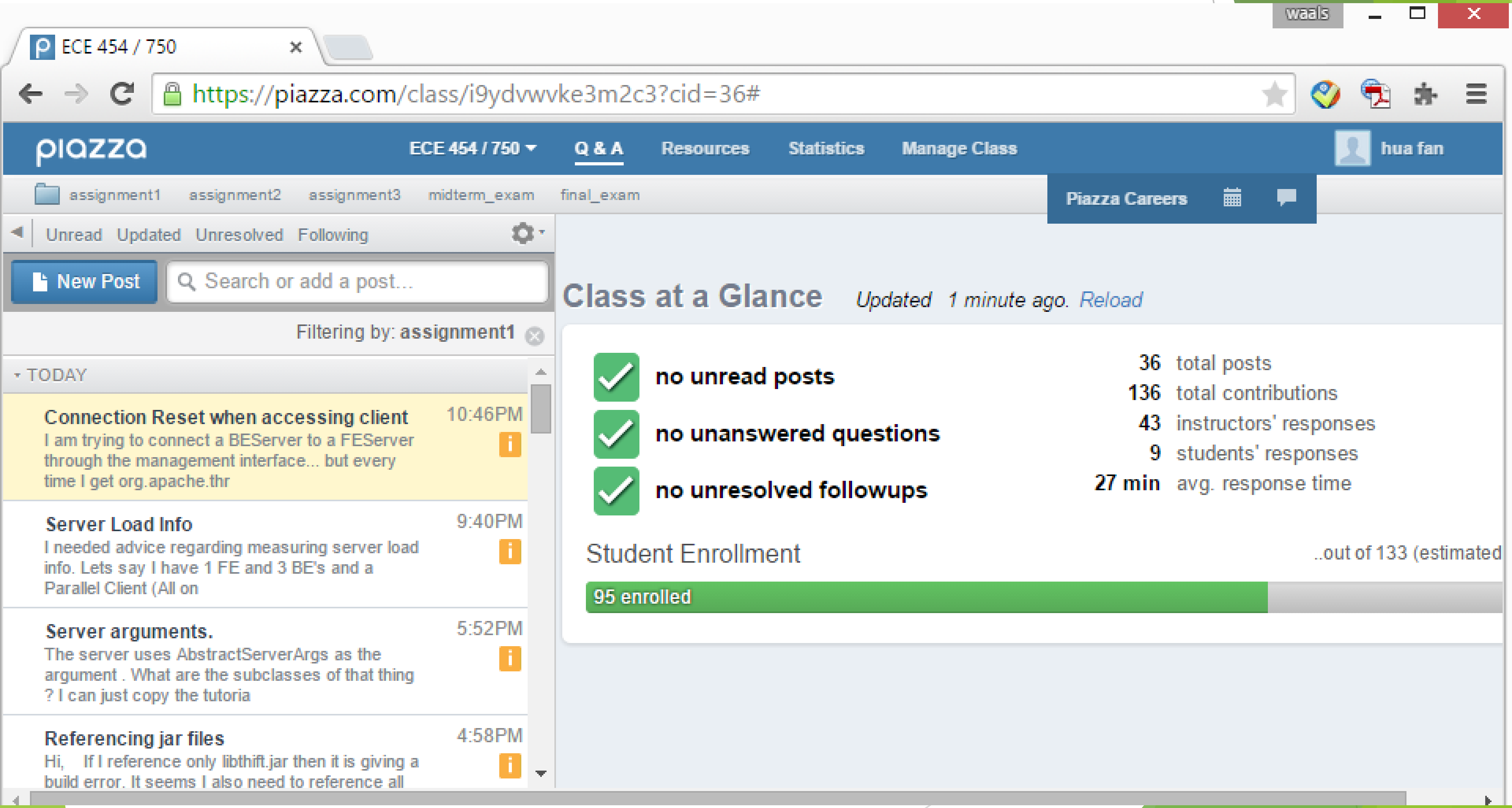
- Start a FE as seed on ecelinux1 taking 6 cores
- Start a BE on ecelinux2 taking 1 cores
- Start another BE on ecelinux2 taking 3 cores
- Start another BE on ecelinux5 taking 2 cores(Intel i7)
- Client send requests
- Request performance counters on the servers
- Expecting: BE with 3 cores receives roughly 3 times as much load as the BE with 1 core; BE on ecelinux5 with two cores receives roughly twice as much load as the BE with 1 core on ecelinux2, due to better hardware.
- Note: To reduce load on the ecelinux servers please keep your experiments short. Do not run the system at full throttle for more than about 10s at a time. This will help avoid interference between performance experiments executed concurrently by different groups.

Sample: fault tolerance

- Start a FE as seed on ecelinux1 taking 4 cores
- Start a BE on ecelinux2 taking 2 cores
- Start another BE on ecelinux5 taking 2 cores
- Client send fixed number of requests to FE allowing both ecelinux2&5 have enough workload
- kill one of the BEs 5 seconds after start-up(requests have not finished then.)
- Request performance counters on the servers
- Expecting: client doesn't receive any exceptions and eventually the numRequestsCompleted at the FE equals the number of requests issued by the client.

Use Piazza

- piazza.com/uwaterloo.ca/summer2015/ece454750



Filtering by: **assignment1**

TODAY

Connection Reset when accessing client 10:46PM
I am trying to connect a BEServer to a FEServer through the management interface... but every time I get org.apache.thr

Server Load Info 9:40PM
I needed advice regarding measuring server load info. Lets say I have 1 FE and 3 BE's and a Parallel Client (All on

Server arguments. 5:52PM
The server uses AbstractServerArgs as the argument . What are the subclasses of that thing ? I can just copy the tutoria

Referencing jar files 4:58PM
Hi, If I reference only libthift.jar then it is giving a build error. It seems I also need to reference all

Class at a Glance *Updated 1 minute ago. Reload*

✓ no unread posts

✓ no unanswered questions

✓ no unresolved followups

36	total posts
136	total contributions
43	instructors' responses
9	students' responses
27 min	avg. response time

Student Enrollment ..out of 133 (estimated

95 enrolled