Large Scale Computing

Oscar Teeninga

lab3

1. Za pomocą gLite proszę uruchomić zadanie wykonujące polecenie hostname (przykładowy jdl znajduje się w podręczniku PLGrid).

   Niestety nie byłem w stanie wygenerować certyfikatu. Otrzymuję błąd:

```
[zeus][plgteeninga@login01 lab3]$ voms-proxy-init --voms vo.plgrid.pl
Enter GRID pass phrase for this identity:
Contacting voms.cyf-kr.edu.pl:15004 [/C=PL/O=GRID/O=Cyfronet/CN=voms.cyf-kr.edu.pl] "vo.plgrid.pl"...
Remote VOMS server contacted succesfully.

VOMS server voms.cyf-kr.edu.pl:15004 returned the following errors:
vo.plgrid.pl: User unknown to this VO.
None of the contacted servers for vo.plgrid.pl were capable of returning a valid AC for the user.
User's request for VOMS attributes could not be fulfilled.
```

   Stworzyłem plik job.jdl, który niestety nie mogłem uruchomić, ponieważ certyfikat nie został wygenerowany.

```
[zeus][plgteeninga@login01 lab3]$ cat job.jdl
Executable = "/bin/hostname";
Arguments = "-f";
StdOutput = "job.out";
StdError = "job.err";
OutputSandbox = {"job.out", "job.err"};
```

2. Proszę przygotować zwykły skypt powłoki bash, ktory wysyła skończoną ilość pingów do hosta home.agh.edu.pl, sprawdzić czy działa lokalnie (np. na serwerze jabba). Następnie przekopiować skrypt na zeus.cyfronet.pl i użyć gLite do uruchomienia go w infrastrukturze PLGrid. Nazwa skryptu (./pinger.sh) powinna być wpisana w polu *Executable* pliku jdl.

   Stworzyłem skrypt pinger.sh, który wysyła 100 pingów na home.agh.edu.pl. Niestety nie mogłem go sprawdzić w pracy poniewaz glite nie działał przez brak certyfikatu.

```
[zeus][plgteeninga@login01 lab3]$ cat pinger.sh
ping -c 100 home.agh.edu.pl
[zeus][plgteeninga@login01 lab3]$ cat pinger.jdl
Executable = "./pinger.sh";
Arguments = "";
StdOutput = "pinger.out";
StdError = "pinger.err";
OutputSandbox = {"pinger.out", "pinger.err"};
```

3. Proszę wykonać skrypt z poprzedniego punktu przy pomocy Rimrocka dostępnego na zeusie (dowolnie wybranym sposobem). Klient REST powinien zostać urchomiony z zewnetrzenej maszyny poza zeusem (np. jabba). Do uwierzytelniania należy przekopiować plik proxy wygenerowany na zeusie do lokalnego katalogu na maszynie, na której wywołujemy klienta.

   Na zeusie nie byłem w stanie otrzymać dostępu, otrzymuje kod 403 i Access Denied.

```
[zeus][plgteeninga@login01 .globus]$ proxy="`cat usercred.p12 | base64 | tr -d '\n'`"
[zeus][plgteeninga@login01 .globus]$ curl -k -X POST --data '{"host": "zeus.cyfronet.pl", "command": "pwd"}' \
> --header "Content-Type:application/json" --header "PROXY:$proxy" https://submit.plgrid.pl/api/process
{"timestamp":1635231080198,"status":403,"error":"Forbidden","message":"Access Denied","path":"/api/process"}[zeus][plgte
eninga@login01 .globus]$
```

   Analogicznie na prywatnej maszynie

```
oscarteeninga@Oscars-MacBook-Pro Large-Scale-Computing % proxy="`cat usercred.p12 | base64 | tr -d '\n'`"
oscarteeninga@Oscars-MacBook-Pro Large-Scale-Computing % curl -k -X POST --data '{"host": "zeus.cyfronet.pl", "command":
 "pwd"}' \
--header "Content-Type:application/json" --header "PROXY:$proxy" https://submit.plgrid.pl/api/process
{"timestamp":1635230497202,"status":403,"error":"Forbidden","message":"Access Denied","path":"/api/process"}%
```

4. Kaskada zadań. Proszę napisać klienta uzywającego Rimrocka uruchomiającego kilka (4-5) zadań w pętli, a następnie sprawdzającego i wyświetlającego ich status co określony przedział czasu. Język - dowolny. Typ zadań - dowolny.

   Napisałem prosty program w pythonie, który tworzy 5 requestów do Rimrocka, a następnie sprawdza w pętli co zostało zwrócone. Niestety nie sposób to wytestować bez działającego proxy.

```python
import requests
import time
import base64

def get_proxy():
    creds_file = open("usercred.p12", "r").read()
    return base64.b64encode(creds_file)

def get_data(cmd):
    return '{"host": "zeus.cyfronet.pl", "command": "{cmd}"}'

def get_header():
    return """{
        Content-Type:application/json,
        PROXY:{proxy}
    }"""

url = "https://submit.plgrid.pl/api/process"
responses = {}
commands = ["pwd", "ls", "ps", "man", "echo"]
proxy = get_proxy()


for cmd in commands:
    responses[cmd] = requests.post(url, headers=get_header(), data=get_data(cmd))

for _ in range(100):
    time.sleep(10)
    for cmd in commands:
        print(responses[cmd].status_code)
```

5. Proszę przy pomocy PLG-Data wyświetlić obrazki wygenerowane na poprzednim laboratorium.

   Z jakiegoś powodu u mnie povray nie działał poprawnie, nie byłem w stanie zdiagnozować co poszło nie tak, natomiast wyświetlanie z poziomu PLG-Data działa poprawnie.