운영체제의 실제 과제3 CLSH 보고서

20191638 이민재

목차

---

1. CLSH 설계

CLSH를 구현하기 위해 ssh를 이용하고 IPC중 pipe를 이용하여 메인 프로세스에서 SSH로 연결중인 프로세스와 pipe를 통해 데이터를 주고 받는 방식으로 구현했다. CLSH와 SSH사이에 stdin, stdout, stderr를 주고받는 pipe 3개를 두고 CLSH와 SSH가 통신할 수 있도록 설계했다.

CLSH의 동작을 구현하기 위해 원격의 다수의 쉘에 명령어를 실행하고 결과를 수집하여 출력하는 기본 구현을 했다. 사용 예시(1), 사용 예시(2)처럼 직접 호스트 이름을 입력하는 방식과 호스트 이름을 파일에 입력하는 방식을 모두 구현했다.

그리고 기본 구현 외에 옵션1, 옵션4, 옵션5, 옵션6을 선택하여 CLSH를 설계했다. 첫번째 옵션인 -hostfile 옵션을 생략할 경우의 동작하도록 구현했다. 네번째 옵션인 Interactive Mode를 구현하여 내부 쉘에서 host 연결을 유지하며 원격에 명령을 수행할 수 있도록 구현했다. 다섯번째 옵션인 CLSH 메인 프로세스에서 각 SSH 연결을 인터럽트 할 수 있도록 구현하고, 여섯번째 옵션으로 SSH 연결 중 일부 연결이 끊어지는 경우 SIGCHLD 시그널을 활용해 메인 프로세스가 종료될 수 있도록 구현했다.

2. CLSH 구현

2.1. 기본 구현

clsh -h node1,node2,node3,node4 cat /proc/loadavg 라는 명령어를 CLSH 메인 프로세스에 stdin으로 입력하면 각 노드에서 cat /proc/loadavg를 실행한 결과를 CLSH의 stdout으로 출력하도록 구현했다. 그리고 clsh -hostfile ./hostfile cat /proc/loadavg 라는 명령어를 CLSH 메인 프로세스에 입력할 경우 ./hostfile에서 cat /proc/loadavg 명령어를 시행할 호스트 이름을 읽은 후 해당 호스트에 명령을 실행한 결과를 stdout으로 출력하도록 구현했다.

이러한 기본 구현을 하기 위해 우선 CLSH 메인 프로세스를 실행할 경우 원격 연결을 실행할 노드의 개수 만큼 fork, exec을 하여 exec으로 ssh 명령어를 두어 자식 프로세스마다 각 노드에 ssh 연결이 되도록 구현했다. 이때 ssh로 exec 할 때는 추가적으로 비밀번호를 입력하는 과정을 생략하기 위해서 sshpass라는 명령어를 추가적으로 활용하여 exec 되는 동시에 해당 호스트에 비밀번호가 입력되며 연결될 수 있도록 구현했다. sshpass를 활용한 명령어의 형태는 다음과 같다. “sshpass -p {비밀번호} ssh -tt node1 -l ubuntu” 비밀번호에는 ubuntu를 입력했다.

프로세스 전체 구조는 메인 프로세스가 부모가 되고 모든 ssh 연결 프로세스는 자식 프로세스가 되는 구조이다. 메인 프로세스와 ssh 연결 프로세스, 즉 부모 프로세스와 자식 프로세스가 통신하는 방식으로 pipe를 활용했다. stdin, stdout, stderr를 주고받아야 하기 때문에 양방향 파이프를 3개만들어서 구현했다. 우선 부모 프로세스에서 stdin으로 명령어를 입력 받으면 해당 명령어에 따라 자식 프로세스에 pipe를 통해 특정 명령을 전달하고 자식 프로세스에서 특정 명령을 수행한 결과를 다시 pipe를 통해 부모 프로세스에 전달해서 부모 프로세스는 전달받은 내용을 전처리하여 stdout으로 내보내는 구조이다. 이 때 자식 프로세스의 경우 stdin과 stdout을 부모 프로세스와 연결된 pipe로 주기 위해서 dup2 함수를 이용했다.

2.2. —hostfile 옵션 생략

--hostfile 옵션을 생략하는 경우 해야할 동작은 첫번째는 CLSH_HOSTS 환경 변수에서 호스트의 이름을 읽어오고, 만약 CLSH_HOSTS 환경변수에서 호스트 이름을 정상적으로 읽어올 수 없다면 CLSH_HOSTFILE 환경 변수에서 파일이름을 읽은 후 해당 파일에 존재하는 호스트의 이름을 읽고, 이번에도 정상적으로 처리되지 않는다면 현재 디렉토리에 존재하는 .hostfile 파일에서 호스트 이름을 읽고, 만약 실패한다면 에러 처리를 한다.

환경 변수로 CLSH_HOSTS, CLSH_HOSTFILE을 설정하기 위해서 docker-compose.yml에 main service에 environment로 CLSH_HOSTS=node1:node2:node3:node4를 설정하고, CLSH_HOSTFILE=clusterfile를 설정했다. 그리고 메인 프로세스에서 해당 환경 변수에 해당하는 값을 읽기 위해서 getenv함수를 활용하여 환경 변수 값을 읽고 각 조건에 따라 호스트의 파싱해서

읽도록 구현했다.

2.3. Interactive Mode

CLSH 메인 프로세스에 -i 옵션이 명령어의 마지막에 적힌 경우 지정된 호스트에 지속적으로 원격 쉘 명령으로 stdin 입력을 할 수 있도록 구현했다. 명령이 없는 경우 내부 쉘에서 입력하면 원격 노드들에서 명령을 실행하도록 구현했다.

2.4. 종료

메인 프로세스가 종료시 모든 자식 프로세스를 종료하기 위해서 CLSH 프로세스에서 quit 명령을 입력받는 경우 모든 자식 프로세스에 SIGTERM 명령을 보내서 자식 프로세스를 종료시키고, 메인 프로세스에서는 wait함수를 이용하여 모든 자식 프로세스가 종료된 후 메인 프로세스도 종료되도록 구현했다. 메인 프로세스가 SIGTERM을 수신하는 경우에도 이와 같이 구현했다.

그리고 메인 프로세스가 SIGQUIT을 수신하는 경우 모든 자식 프로세스를 가능한 즉시 중지시키기 위해서 모든 자식 프로세스에 SIGKILL 명령을 보내 즉시 종료시키고 메인 프로세스에서는 wait 함수를 이용해 모든 자식 프로세스가 종료된 후 메인 프로세스가 종료되도록 구현했다.

2.5. 예외

CLSH 메인 프로세스에서 각 SSH 연결 중 일부 연결이 끊어지는 경우를 알기 위해 자식 프로세스가 종료하는 경우 메인 프로세스에 보내는 SIGCHLD 시그널을 받는 경우에 에러 메시지를 출력하고 모든 자식 프로세스를 종료하고 메인 프로세스도 종료하도록 구현했다. SIGCHLD를 자식 프로세스가 종료하는 경우만 받도록 하기 위해서 sa_flags = SA_NOCLDSTOP으로 설정했다.

SIGCHLD 핸들러가 원하는 예외 상황 뿐만 아니라 2.3. 종료에 해당하는 상황에서도 메인 프로세스가 자식 프로세스에 SIGTERM이나 SIGKILL 명령어를 보낼 경우 자식 프로세스가 종료되고 이를 다시 메인 프로세스에 SIGCHLD를 보내는 경우가 발생하여 메인 프로세스가 종료되는 상황인지 여부를 판단하는 bool 값을 전역적으로 두어 메인 프로세스가 먼저 종료되는 상황에서는 SIGCHLD 핸들러가 별도 동작을 하지 않고 바로 return을 하도록 구현했다.

3. CLSH 결과

3.1. 설치하기

소스코드.zip 파일을 압축 해제하면 Dockerfile, docker-compose.yml, 그리고 source 폴더 안에 clsh.c 파일과 프로그램을 실행하기 위한 여러가지 파일(.hostfile, clusterfile, cluster.txt, hostfile)이 존재한다.

intel cpu를 사용하는 경우에는 Dockerfile 8번 라인과 19번 라인에 존재하는 dumb-init_1.2.5_arm64.deb를 dumb-init_1.2.5_amd64.deb로 변경해주어야 올바르게 실행된다. arm cpu를 사용하는 경우에는 그대로 두면 된다.

docker-compose.yml이 존재하는 경로에서 쉘에 docker compose up -d —build 명령어를 치면 도커 컨테이너들이 실행된다. docker compose exec -it main /bin/bash 명령어를 통해 CLSH 프로그램을 실행할 메인 컨테이너에 접속할 수 있다. 그 후 cd source 로 source 디렉토리로 이동한 후 gcc -o clsh clsh.c 명령어를 친 후 ./clsh를 다시 쉘에 입력하면 CLSH 프로그램이 실행된다.

이 때 맨 처음 ./clsh 명령어를 치는 경우에 바로 모든 자식 프로세스가 종료되게 된다. 그 이유는 clsh 프로그램에서 ssh 연결을 수행할 때 sshpass 명령어를 이용해서 비밀번호 추가 입력 없이 ssh 연결을 바로 수행하도록 구현했는데, sshpass로 exec하는 경우 이전에 연결해본 적이 있는 host에 연결을 진행할 때만 정상적으로 ssh 연결이 진행된다. 때문에 맨 처음 ./clsh 명령어로 CLSH 프로그램을 실행하기 전에 메인 프로세스에서 각 노드에 ssh node1 -l ubuntu로 ssh연결을 한 번씩 수행 한 후 ./clsh 명령어를 입력해야 CLSH 프로그램이 올바르게 동작한다.

3.2. 빠르게 시작하기

도커 컴포즈 파일로 도커 컨테이너들을 실행시키고, 메인 컨테이너에 접속한다.
docker compose up -d —build
docker compose exec -it main /bin/bash

sshpass 연결을 위해 각 노드에 한번씩 ssh 연결을 미리 진행한다.
ssh node1 -l ubuntu
ssh node2 -l ubuntu
ssh node3 -l ubuntu
ssh node4 -l ubuntu

source 디렉토리로 이동하여 clsh.c 소스코드로 clsh 프로그램을 컴파일하고 실행한다.

cd source

gcc -o clsh clsh.c

./clsh

기본 명령어를 실행하면 원격 쉘에서 cat /proc/loadavg를 실행한 결과를 출력한다.

clsh -h node1,node2,node3,node4 cat /proc/loadavg

clsh -hostfile ./hostfile cat /proc/loadavg

3.3. 사용방법

우선 clsh 프로그램을 실행하고 모든 노드에 정상적으로 연결되면 다음과 같이 출력된다.

```
root@main:/source# ./clsh
모든 노드에 ssh 연결 중입니다.
node2:Last login: Sun Dec 18 20:27:56 2022 from 172.19.0.5
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@node2:~$
node3:Last login: Sun Dec 18 20:28:33 2022 from 172.19.0.5
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@node3:~$
node4:Last login: Sun Dec 18 20:28:52 2022 from 172.19.0.5
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@node4:~$
node1:Last login: Sun Dec 18 20:27:31 2022 from 172.19.0.5
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@node1:~$
모든 노드에 연결되었습니다.
```

기본 구현 명령어 실행 결과

clsh -h node1,node2,node3,node4 cat /proc/loadavg

```
clsh -h node1,node2,node3,node4 cat /proc/loadavg
node2:0.00 0.00 0.00 4/590 58
node3:0.00 0.00 0.00 1/587 65
node4:0.00 0.00 0.00 8/590 61
node1:0.00 0.00 0.00 5/590 75
```

clsh -hostfile ./hostfile cat /proc/loadavg (./hostfile에는 node1과 node3이 기록되어 있다)

```
[clsh --hostfile ./hostfile cat /proc/loadavg
node3:0.04 0.01 0.00 7/587 66
node1:0.04 0.01 0.00 5/588 76
```

옵션 1. —hostfile 옵션을 생략할 경우
docker-compose.yml파일에서 main의 환경변수 CLSH_HOSTS를
node1:node2:node3:node4 로 둔 상태이다.

```
[clsh cat /proc/loadavg
Note: use CLSH_HOSTS environment
node2:0.05 0.03 0.00 1/587 59
node3:0.05 0.03 0.00 8/588 67
node4:0.05 0.03 0.00 6/588 62
node1:0.05 0.03 0.00 2/588 77
```

옵션 4. Interactive mode 구현

```
[clsh --hostfile ./hostfile -i
Enter 'quit' to leave this interactive mode
Working with nodes: node1, node3
[clsh>uname
--------------------------
node3:Linux
node1:Linux
--------------------------
[clsh>cat /proc/loadavg
--------------------------
node3:0.06 0.04 0.00 10/590 104
node1:0.06 0.04 0.00 10/590 113
--------------------------
clsh>
```

옵션 5. 종료 SIGTERM이나 SIGQUIT 시그널을 메인 프로세스에 보내면
자식 프로세스를 모두 종료하고 메인 프로세스가 종료된다.
쉘에서 kill 명령어를 입력해 메인 프로세스에 SIGTERM과 SIGQUIT을 보냈다.

```
Received Signal:: Terminated
자식 프로세스(207)가 종료되었습니다.
자식 프로세스(209)가 종료되었습니다.
자식 프로세스(210)가 종료되었습니다.
자식 프로세스(208)가 종료되었습니다.
모든 프로세스가 종료되었습니다.
```

```
Received Signal:: Quit
자식 프로세스(221)가 종료되었습니다.
자식 프로세스(218)가 종료되었습니다.
자식 프로세스(219)가 종료되었습니다.
자식 프로세스(220)가 종료되었습니다.
모든 프로세스가 종료되었습니다.
```

옵션 6. SSH 연결 중 일부 연결이 끊어지면 에러를 출력하고 메인 프로세스도 종료된다.
쉘에서 kill 명령어를 입력해 자식 프로세스에 SIGQUIT을 보냈다.

```
자식 프로세스(234)가 종료되었습니다.
ERROR: node2 connection lost
자식 프로세스(236)가 종료되었습니다.
자식 프로세스(233)가 종료되었습니다.
자식 프로세스(235)가 종료되었습니다.
모든 프로세스가 종료되었습니다.
```

추가 예시
출력이 길지 않은 경우 다음과 같이 원격 명령을 수행한 결과를 알 수 있다.
Interactive mode로 node1과 node3에 접속하고 ls / 명령어를 수행한 결과와
cd etc, ls 명령어를 수행한 결과이다.

```
[clsh --hostfile ./hostfile -i
Enter 'quit' to leave this interactive mode
Working with nodes: node1, node3
[clsh>ls /
-------------------------
node3:bin   build  etc   lib    mnt  proc  run   srv  tmp  var
boot  dev    home  media  opt  root  sbin  sys  usr
node1:bin   build  etc   lib    mnt  proc  run   srv  tmp  var
boot  dev    home  media  opt  root  sbin  sys  usr
-------------------------
clsh>
```

```
clsh>cd etc
--------------------------
node3:node1:--------------------------
clsh>ls
--------------------------
node3:adduser.conf            host.conf           nsswitch.conf  shadow-
alternatives            hostname            opt            shells
apt                     hosts               os-release     skel
bash.bashrc             hosts.allow         pam.conf       ssh
bindresvport.blacklist  hosts.deny          pam.d          ssl
binfmt.d                init.d              passwd         subgid
ca-certificates         inputrc             passwd-        subgid-
ca-certificates.conf    issue               perl           subuid
cron.d                  issue.net           profile        subuid-
cron.daily              kernel              profile.d      sudo.conf
dbus-1                  ld.so.cache         protocols      sudo_logsrvd.conf
node1:adduser.conf            host.conf           nsswitch.conf  shadow-
alternatives            hostname            opt            shells
apt                     hosts               os-release     skel
bash.bashrc             hosts.allow         pam.conf       ssh
bindresvport.blacklist  hosts.deny          pam.d          ssl
binfmt.d                init.d              passwd         subgid
ca-certificates         inputrc             passwd-        subgid-
ca-certificates.conf    issue               perl           subuid
cron.d                  issue.net           profile        subuid-
cron.daily              kernel              profile.d      sudo.conf
dbus-1                  ld.so.cache         protocols      sudo_logsrvd.conf
--------------------------
clsh>
```