Unix System Programming
Project Report

June 2nd, 2019

201414840 Jung YoonSung
201315298 Jung SungMin

CONTENTS

- Role
- Overview
- Experiments
- Evaluation
- Conclusion

Role

- 정윤성
  - Log 파일 처리
  - Cgi 요청 처리
  - 서버 최적화
- 정성민
  - 서버 스켈레톤 코드 작성
  - 서버 최적화

Overview

load on memory

- Load file on memory
  - 서버가 돌아가기 전 example 폴더 안에 image, html 파일을 buffer에 적재
  - 요청이 들어오면 전송

- No load file on memory
  - 요청이 들어올 때 마다 작업(read(), write(), etc..)을 수행

Experiments

비교군

- 처음 만들었던 서버(Original server) – 서버 작동하는 중(while문)에 계속 변수 선언
- Load on memory
- No load on memory

환경

- 개인 서버
- 프로세스 최소로 진행

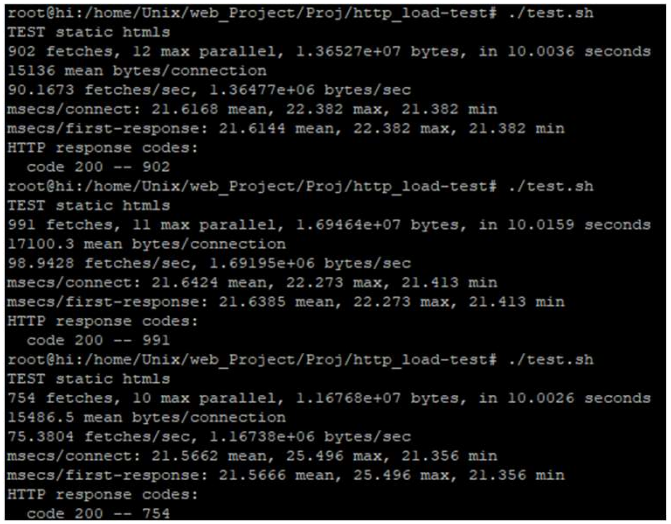
Experiments

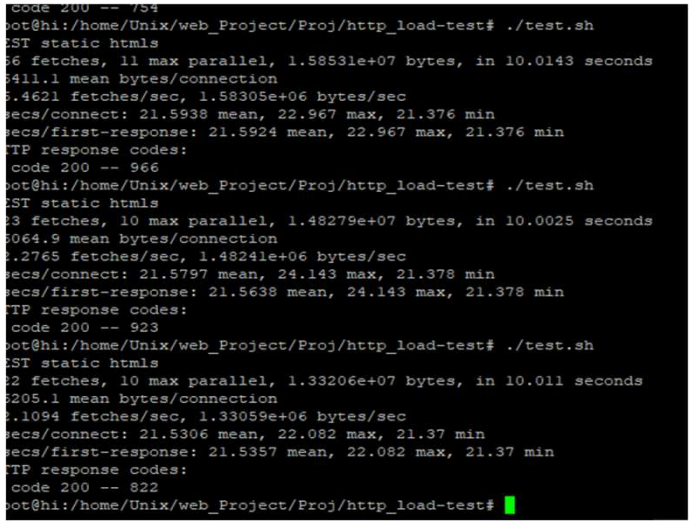
Original server

- 변수 선언들이 while문 안에 있음
- 경로, http 헤더 등의 문자열 복사 시에 strcpy() 함수 이용

Experiments

Original server





- 평균 시간(msecs/first-response) : 21.584

Experiments

Load on memory

- 서비스 해야할 것들을 buffer(memory)에 load
  - Path 와 data를 미리 서버가 가지고 있음
  - Fork()가 실행되기 전, 모든 file data를 buffer에 적재
  - 요청이 들어오면 해당 요청에 맞는 buffer를 write()함수를 이용하여 전송

Experiments

Load on memory

```
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
992 fetches, 13 max parallel, 1.65614e+07 bytes, in 10.0046 seconds
16695 mean bytes/connection
99.1543 fetches/sec, 1.65538e+06 bytes/sec
msecs/connect: 21.6176 mean, 22.716 max, 21.288 min
msecs/first-response: 21.6192 mean, 22.716 max, 21.288 min
HTTP response codes:
  code 200 -- 992
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
989 fetches, 12 max parallel, 1.65512e+07 bytes, in 10.0175 seconds
16735.3 mean bytes/connection
98.7274 fetches/sec, 1.65223e+06 bytes/sec
msecs/connect: 21.6523 mean, 22.753 max, 21.383 min
msecs/first-response: 21.648 mean, 22.753 max, 21.383 min
HTTP response codes:
  code 200 -- 989
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
938 fetches, 9 max parallel, 1.38419e+07 bytes, in 10.0175 seconds
14756.8 mean bytes/connection
93.6358 fetches/sec, 1.38177e+06 bytes/sec
msecs/connect: 21.5957 mean, 22.686 max, 21.178 min
msecs/first-response: 21.591 mean, 22.686 max, 21.267 min
HTTP response codes:
  code 200 -- 938
```

```
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
990 fetches, 14 max parallel, 1.54839e+07 bytes, in 10.0136 seconds
15640.3 mean bytes/connection
98.8656 fetches/sec, 1.54629e+06 bytes/sec
msecs/connect: 21.6482 mean, 22.288 max, 21.333 min
msecs/first-response: 21.6401 mean, 22.288 max, 21.333 min
HTTP response codes:
  code 200 -- 990
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
992 fetches, 11 max parallel, 1.61606e+07 bytes, in 10.0122 seconds
16290.9 mean bytes/connection
99.0791 fetches/sec, 1.61409e+06 bytes/sec
msecs/connect: 21.5906 mean, 22.286 max, 21.361 min
msecs/first-response: 21.5967 mean, 22.286 max, 21.361 min
HTTP response codes:
  code 200 -- 992
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
944 fetches, 10 max parallel, 1.42926e+07 bytes, in 10.016 seconds
15140.5 mean bytes/connection
94.249 fetches/sec, 1.42697e+06 bytes/sec
msecs/connect: 21.5935 mean, 22.945 max, 21.412 min
msecs/first-response: 21.588 mean, 22.945 max, 21.412 min
HTTP response codes:
  code 200 -- 944
```

- 평균 시간(msecs/first-response) : 21.613
- 요청이 들어오기 전, 모든 data를 buffer에 저장을 하는데 시간이 더 느려지는 지 의문점이 있음

Experiments

No load on memory

- 서비스 해야할 것들을 buffer에 load 하지 않음
  - 요청이 들어오면 요청에 해당하는 서비스를 확인(문자열 검사, read() 함수)하고 해당 data를 전송(write() 함수)

- data 복사할 때 memcpy()을 사용 – 메모리 공간을 복사

Experiments

Load on memory

```
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
993 fetches, 11 max parallel, 1.40069e+07 bytes, in 10.0073 seconds
14105.6 mean bytes/connection
99.2278 fetches/sec, 1.39967e+06 bytes/sec
msecs/connect: 21.5708 mean, 22.285 max, 21.259 min
msecs/first-response: 21.5719 mean, 22.285 max, 21.259 min
HTTP response codes:
  code 200 -- 993
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
709 fetches, 10 max parallel, 1.22745e+07 bytes, in 10.0072 seconds
17312.4 mean bytes/connection
70.8487 fetches/sec, 1.22656e+06 bytes/sec
msecs/connect: 21.5257 mean, 22.482 max, 21.29 min
msecs/first-response: 21.5305 mean, 22.482 max, 21.29 min
HTTP response codes:
  code 200 -- 709
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
656 fetches, 8 max parallel, 1.05218e+07 bytes, in 10.0019 seconds
16039.4 mean bytes/connection
65.5874 fetches/sec, 1.05198e+06 bytes/sec
msecs/connect: 21.611 mean, 23.288 max, 21.358 min
msecs/first-response: 21.6041 mean, 23.288 max, 21.358 min
HTTP response codes:
  code 200 -- 656
```

```
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
737 fetches, 10 max parallel, 1.23135e+07 bytes, in 10.017 seconds
16707.6 mean bytes/connection
73.5751 fetches/sec, 1.22926e+06 bytes/sec
msecs/connect: 21.5542 mean, 24.204 max, 21.258 min
msecs/first-response: 21.5593 mean, 24.204 max, 21.258 min
HTTP response codes:
  code 200 -- 737
root@hi:/home/Unix/web_Project/Proj/http_load-test# ./test.sh
TEST static htmls
658 fetches, 7 max parallel, 1.0234e+07 bytes, in 10.0063 seconds
15553.2 mean bytes/connection
65.7588 fetches/sec, 1.02276e+06 bytes/sec
msecs/connect: 21.5225 mean, 22.242 max, 21.267 min
msecs/first-response: 21.5234 mean, 22.242 max, 21.267 min
HTTP response codes:
  code 200 -- 658
```

- 평균 시간(msecs/first-response) : 21.557

Evaluation

누적 실행시간 비교

1st no load on memory

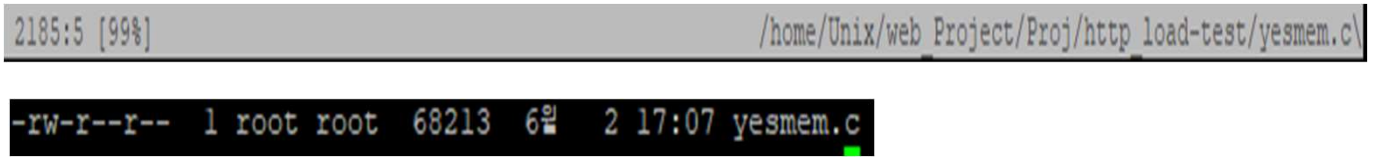
2nd original server

3rd load on memory

Evaluation

속도 향상을 위해 해본 노력

- 모든 정보(32개의 file)를 각 배열에 load한 뒤, if문에 따라 해당 배열을 전송하는 방식으로, 파일을 읽는 작업을 1회만 수행한 뒤, 모든 작업을 메모리에서 다루게 변경(2000줄의 hard coding)



- 문자열 복사를 메모리 복사로 바꿈 {strcpy() -> memcpy() }
- ~~Html 파일 수정(html file마다 용량을 줄일 수 있음)~~
- 고수준 파일 입출력 -> 저수준 파일 입출력
- Listen() 함수에서 처리할 수 있는 client수를 조정

Evaluation

속도 향상을 위해 해본 노력

- Cgi 파일 서비스 처리시 포인터를 사용하여 처리
- Strcpy(), strcat() -> sprint() : 2번의 문자열 복사 작업을 1번으로 줄임
- If .cgi > Not found > file data 검색 순서가 가장 속도가 빠름
- Fork() -> vfork() 변경 해봤으나 서비스 기능이 제한됨
- TCP -> UDP protocol 변경 시도 해봤으나 서비스 불가

Conclusion

Conclusion

- CPU의 병목현상을 제어하기 힘들어 동시에 들어오는 여러 요청을 효율적으로 처리하는데 어려움.
- 요청이 들어오기 전, 모든 data를 buffer에 저장을 하고 문자열을 검사한 뒤 조건에 부합하는 buffer 전송하는 것이 더 느림
- Log file을 기록하는 것을 비활성화하여도 속도의 차이가 없음을 보아 log 기록의 속도는 빠름
- Taskset 명령어를 사용하여 CPU 활용을 제한하여도 속도의 차이는 없음
- Vfork()를 실행하면 첫번째 실행만 유효함