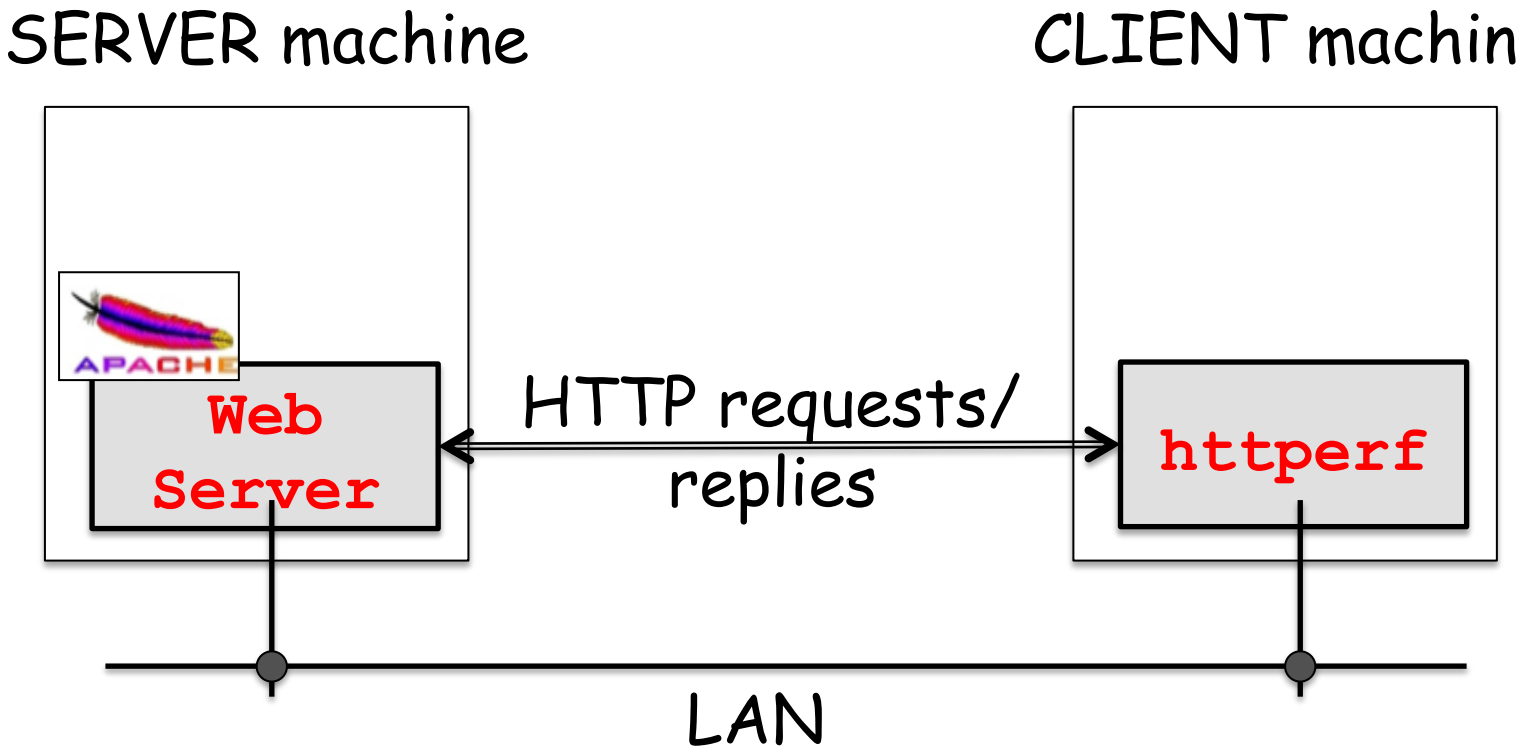
Performance Analysis

Case study and experimental setup

Introduction

- Measuring and understanding performance indicators of a server installation (e.g., small cloud system)
  - private and public network switches
  - authentication server
  - computation nodes
  - storage node
  - logging/monitoring node
- Objectives
  - system evaluation;
  - architectural bottlenecks;
  - factors influencing performance;
  - improvement;
  - …

Experimental setup



- Linux machines; either a physical or virtualized environment.

Software Download

- Apache Web Server:
  - ver. 2.0.64 is used in this study
  - source: `httpd-2.0.64.tar.gz`

  http://httpd.apache.org/download.cgi

- Httperf tool
  - ver 0.9.0
  - source: `httperf-0.9.0.tar.gz`

  ftp://ftp.hpl.hp.com/pub/httperf/

Installing Apache Web Server

- Copy **httpd-2.0.64.tar.gz** on the server machine and extract the source code

```
[fdpfi@localhost ~]$ ls
httpd-2.0.64.tar.gz     …      …

[fdpfi@localhost ~]$ gzip -dvf
httpd-2.0.64.tar.gz
httpd-2.0.64.tar.gz:     78.7% -- replaced with
httpd-2.0.64.tar

[fdpfi@localhost ~]$ tar xvf httpd-2.0.64.tar
…
httpd-2.0.64/buildconf
httpd-2.0.64/acconfig.h
httpd-2.0.64/.gdbinit
```

Installing Apache Web Server

- Move to the installation directory, configure and compile the code
  - **prefix** indicates the folder where the web server will be installed
  - the **make** command will start the compilation

```
[fdpfi@localhost ~]$ cd httpd-2.0.64

[fdpfi@localhost httpd-2.0.64]$  mkdir /home/fdpfi/
impianti-ws

[fdpfi@localhost httpd-2.0.64]$ ./configure --prefix=/
home/fdpfi/impianti-ws

[fdpfi@localhost httpd-2.0.64]$ make
```

Installing Apache Web Server

- The web server is installed in the folder specified by prefix with the make install command

```
[fdpfi@localhost httpd-2.0.64]$ make install
…
…
mkdir /home/fdpfi/impianti-ws/man
mkdir /home/fdpfi/impianti-ws/man/man1
mkdir /home/fdpfi/impianti-ws/man/man8
mkdir /home/fdpfi/impianti-ws/manual
Installing build system files
make[1]: Leaving directory `/home/fdpfi/
httpd-2.0.64'
```

Starting/Stopping the server

- Move to the installation directory (e.g., impianti-ws) and run the following command:
  - apachectl start: starts the web server
  - apachectl stop: stops the web server
- An error might be raised if the server is started with port 80, such as in the example

```
[fdpfi@localhost impianti-ws]$ ./bin/apachectl start
(13)Permission denied: make_sock: could not bind to
address [::]:80
no listening sockets available, shutting down
Unable to open logs
```

Configuration File

- The httpd.conf file (available at the conf folder in the installation folder) allows specifying several configuration parameters of the web server
  - the "Listen" parameter is the port used by the server (it can be changed into 8080 instead of 80 to solve the startup issue discussed before)
  - the conf file features important tuning parameters, such as timeouts, max number of requests/conn, …

```
[fdpfi@localhost impianti-ws]$ vi conf/httpd.conf

[fdpfi@localhost impianti-ws]$ ./bin/apachectl start
```

Testing the installation

http://192.168.185.143:8080/

Installing httperf

- Copy the **`httperf-0.9.0.tar.gz`** on the client machine and extract the source code

```
[fdpfi@localhost ~]$ ls
httperf-0.9.0.tar.gz    …      …

[fdpfi@localhost ~]$ gzip -dvf httperf-0.9.0.tar.gz
httperf-0.9.0.tar.gz:    78.0% -- replaced with
httperf-0.9.0.tar

[fdpfi@localhost ~]$ tar xvf httperf-0.9.0.tar
…
httperf-0.9.0/config.h.in
httperf-0.9.0/AUTHORS
httperf-0.9.0/ChangeLog
```

Installing httperf

- Move to your installation directory and run the following commands

```
[fdpfi@localhost ~]$ cd httperf-0.9.0

[fdpfi@localhost httperf-0.9.0]$ mkdir build

[fdpfi@localhost httperf-0.9.0]$ cd build

[fdpfi@localhost build]$ /home/fdpfi/httperf-0.9.0/
configure

[fdpfi@localhost build]$ make

[root@localhost build]# make install  (NB: it is run as
"root")
```

Running httperf

- httperf can be now used to generate a workload and to test the web server
  - a simplified output is shown in the example

```
[fdpfi@localhost httperf-0.9.0]$ httperf  --hog
--server 192.168.185.143 --port 8080  --uri /index.html.it --rate 1
--num-conn 1 --num-call 1 --timeout 10
…
Total: connections 1 requests 1 replies 1 test-duration 0.002 s
…
Reply rate [replies/s]: min 0.0 avg 0.0 max 0.0 stddev 0.0 (0 samples)
Reply time [ms]: response 2.0 transfer 0.0
Reply size [B]: header 251.0 content 1788.0 footer 0.0 (total 2039.0)
Reply status: 1xx=0 2xx=1 3xx=0 4xx=0 5xx=0
…
Errors: total 0 client-timo 0 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

Request-oriented workload parameters

```
[fdpfi@localhost httperf-0.9.0]$ httperf  --hog --server
192.168.185.143 --port 8080  --uri /index.html.it --rate 1 --
num-conn 1 --num-call 1 --timeout 10
```

- --hog: allows httperf to request as many TCP ports it needs
- --sever: IP or hostname of the server
- --port: port of the server
- --rate: rate at which connections are created (conns/second)
- --num-conn: total number of connections
- --num-call: number of calls per connection
- --timeout: the time (in seconds) httperf will wait for a server response

Example

```
[fdpfi@localhost httperf-0.9.0]$ httperf  --hog --server 192.168.185.143 --port
8080  --uri /index.html.it --rate 10 --num-conn 100 --num-call 10 --timeout 10

Total: connections 100 requests 1000 replies 1000 test-duration 9.943 s

Connection rate: 10.1 conn/s (99.4 ms/conn, <=2 concurrent connections)
Connection time [ms]: min 8.2 avg 43.3 max 144.4 median 36.5 stddev 23.7
...
Request rate: 100.6 req/s (9.9 ms/req)
Request size [B]: 81.0

Reply rate [replies/s]: min 100.0 avg 100.0 max 100.0 stddev 0.0 (1 samples)
Reply time [ms]: response 3.8 transfer 0.1
Reply size [B]: header 251.0 content 1788.0 footer 0.0 (total 2039.0)
Reply status: 1xx=0 2xx=1000 3xx=0 4xx=0 5xx=0

Errors: total 0 client-timo 0 socket-timo 0 connrefused 0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

Interpreting the output

- Workload session summary

```
Total: connections 100 requests 1000 replies 1000 test-
duration 9.943 s
```

- Connection and request rate statistics

```
Connection rate: 10.1 conn/s (99.4 ms/conn, <=2
concurrent connections)
Connection time [ms]: min 8.2 avg 43.3 max 144.4
median 36.5 stddev 23.7
…
Request rate: 100.6 req/s (9.9 ms/req)
```

Performance indicators

- Reply rate and reply time:
  - a reply rate sample is collected every 5 second!
  - make sure tests are long enough by increasing the --num-conn parameter (the example test lasted around 9.9 seconds)
  - reply time: average time for the server to respond to a request

```
Reply rate [replies/s]: min 100.0 avg 100.0 max 100.0 stddev
0.0 (1 samples)
Reply time [ms]: response 3.8 transfer 0.1
Reply size [B]: header 251.0 content 1788.0 footer 0.0 (total
2039.0)
Reply status: 1xx=0 2xx=1000 3xx=0 4xx=0 5xx=0
```

Error statistics

```
Errors: total 0 client-timo 0 socket-timo 0 connrefused
0 connreset 0
Errors: fd-unavail 0 addrunavail 0 ftab-full 0 other 0
```

- Errors occurred during the test. E.g.:
  - client-timo: number of times a connection failed due to the client timeout
  - connrefused: number of times a TCP connection failed with a "connection refused by server error"
  - fd-unavail: httperf run out of file descriptors (client is overloaded)

A quickstart httperf reference documentation is available at http://www.comlore.com/redist_files/httperf-quickstart-guide.pdf

Example 1

- Analysis of the server reply time for a given request rate (e.g., 10000 requests/sec)

```
httperf  --hog --server 192.168.185.143 --port 8080
--uri /index.html.it --rate 20 --num-conn 400 --num-call 500
```

- Collected sample (reply time); N=10

  {9.6, 8.8, 8.4, 9.0, 9.4, 8.8, 9.8, 10.1, 10.1, 10.4}

$\overline{x}$ = 9.44 ms (mean) => C.O.V. = 0.67/9.44 = 0.07

s = 0.67 ms (std dev)

Example 1

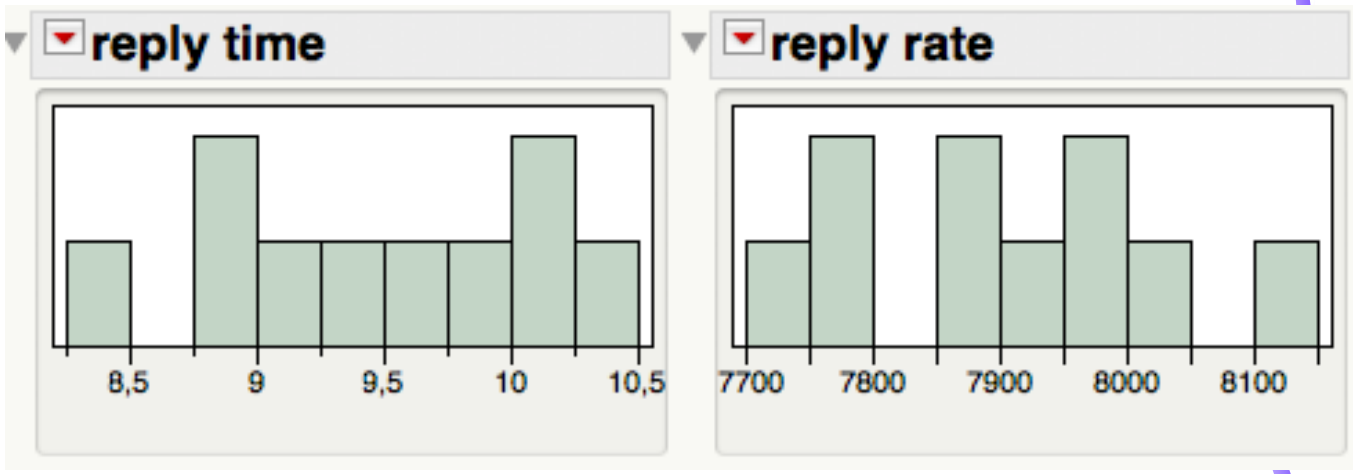
- Collected sample (reply rate); N=10

  {7868.5, 8039.6, 8111.3, … , 7726,4}

$\bar{x}$ = 7908.7 rep/s => C.O.V. = 121.9/7908,7 = 0.02

s = 121.9 rep/s

- Single parameter histograms

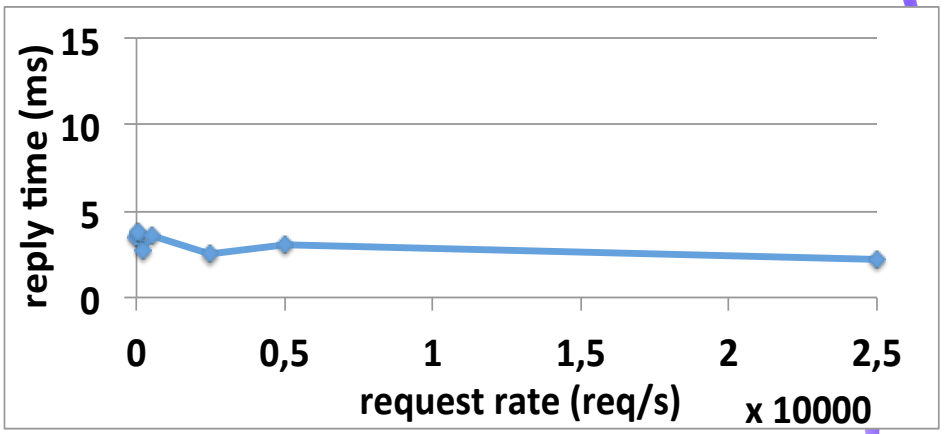


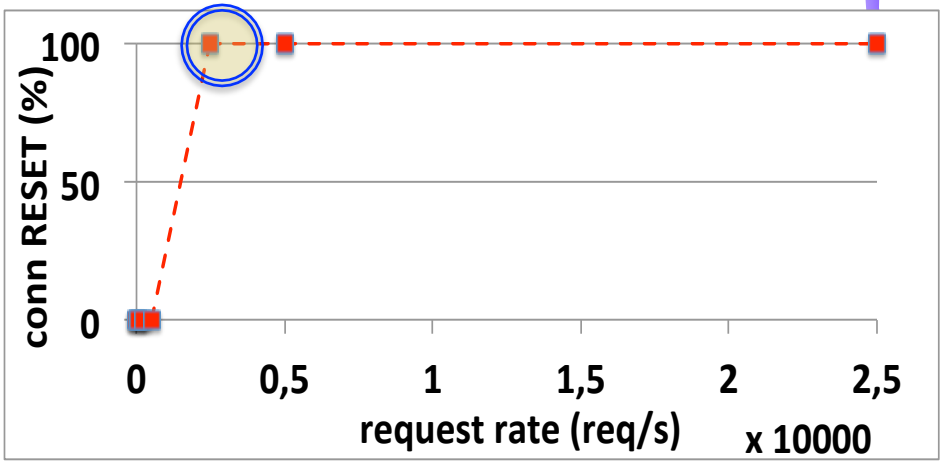
What about the data distribution? Are the two parameters correlated?

Example 2

- Analysis of the server reply time with respect to increasing request rates (rate=5, num-conn 50, num-call in [1…5000])

- All the connections show a RESET failure when the rate is 2,500 req/sec



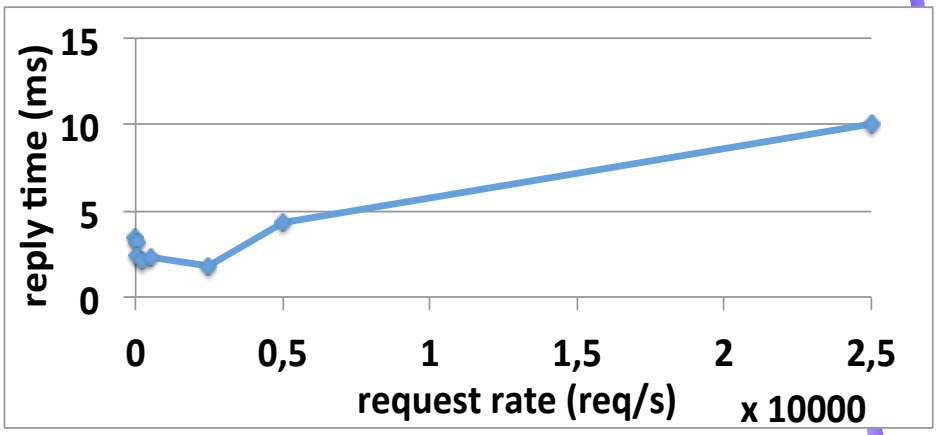


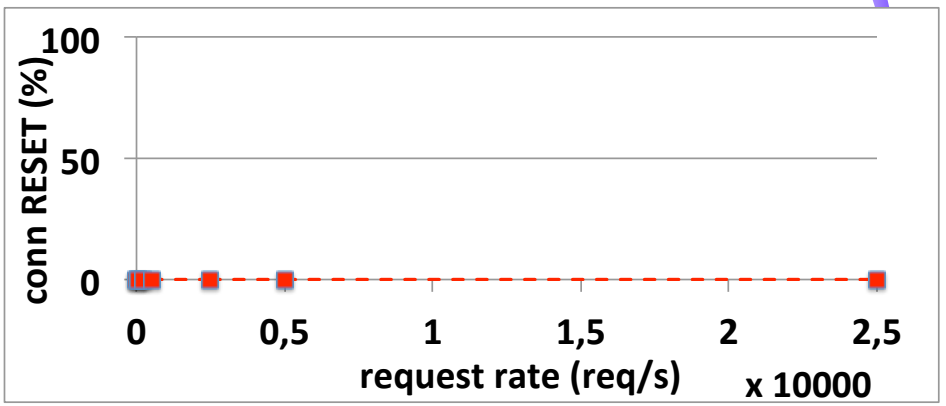
A look into httpd.conf revealed that the server was configured to accept a MAX number of request/conn

Example 2

- The same test is repeated by properly tuning the server capacity by means of httpd.conf

- Analysis shows that the reply time increases as the request rate increases; connections do not exhibit resets

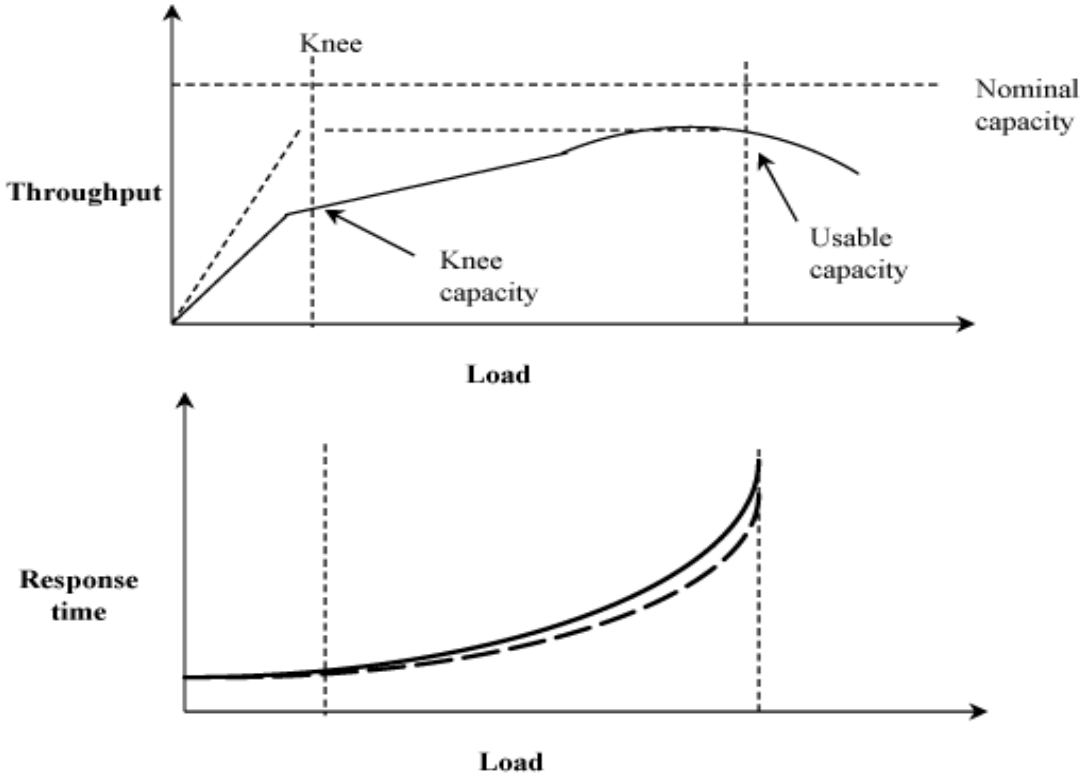




Exercise

- Given a value of the httperf rate, analyze throughput and response time with respect to increasing values of the load applied to the web server.

- Collect several observations for each value of the load condition



Exercise

- Estimate:
  - usable capacity,
  - knee capacity,
  - power.

- Repeat the exercise with different values of the httperf rate parameter
  - 1, 10, 20, 30, 40, 50