Case study and experimental setup

Objectives

- Workload Characterization
- Capacity Test and Experimental Design and Analysis
- Performance Degradation Analysis

Capacity Test

Capacity Test

- Capacity test: characterize performance in the limit
- More generally, performance analysis refers to characterizing performance under several working conditions

Capacity Test

- Performance can be characterized by several indicators
- In the web server example, typical indicators are:
  - Response time
  - Response rate (throughput)
  - Latency
  - Number of errors (i.e, requests failed)
  - …

Example 1

- Performance indicators can be characterized by the same techniques used for WL characterization

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

- Collected sample (response time); N=10

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

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

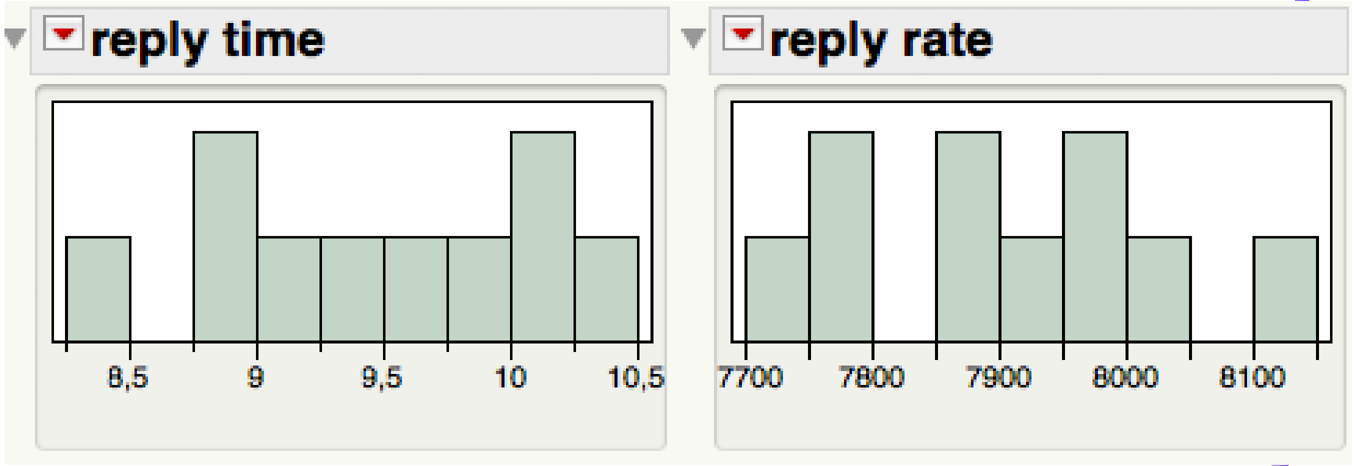
s = 0.67 ms (std dev)

Example 1 (cont.)

- Collected sample (response 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?

Capacity Test

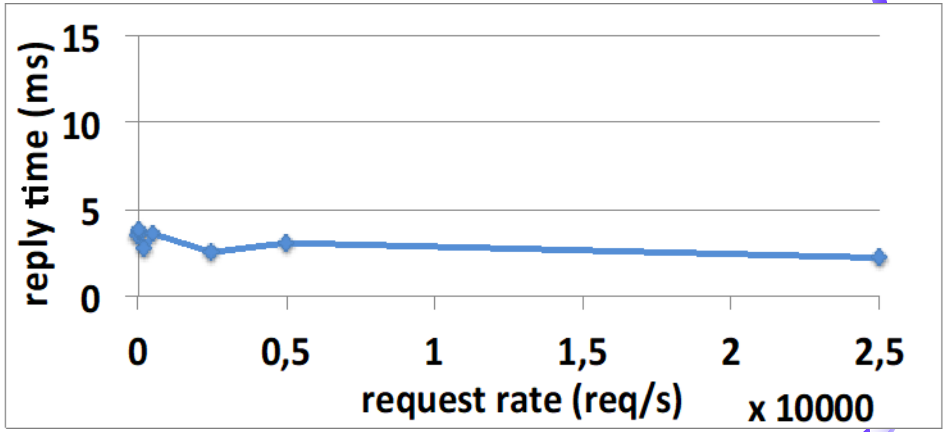
- Capacity Test
  - Objective: find the limits of the system under study
  - Useful to: capacity planning, management, and performance tuning
  - Can be used as input to other experimental analyses

  - Requires a preliminary WL characterization

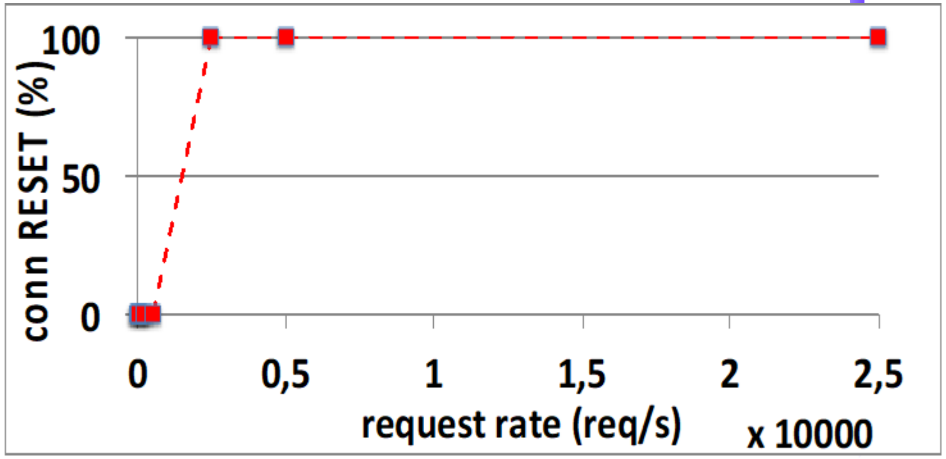
Example 2

- Analysis of the server response time with respect to increasing request rates

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



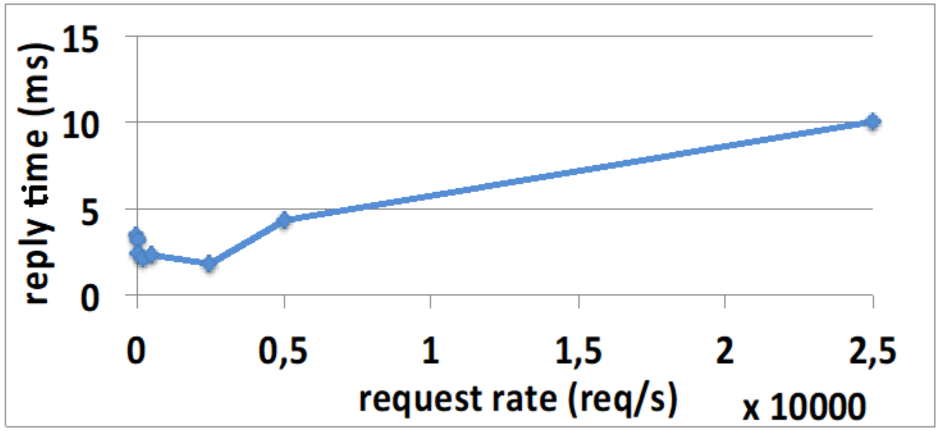
A look into httpd.conf revealed that the server was configured to accept a MAX number of request/conn

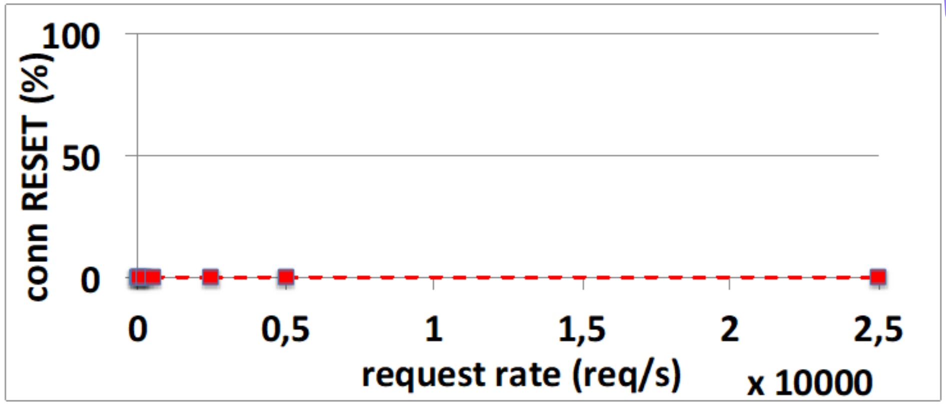


Example 2 (cont.)

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

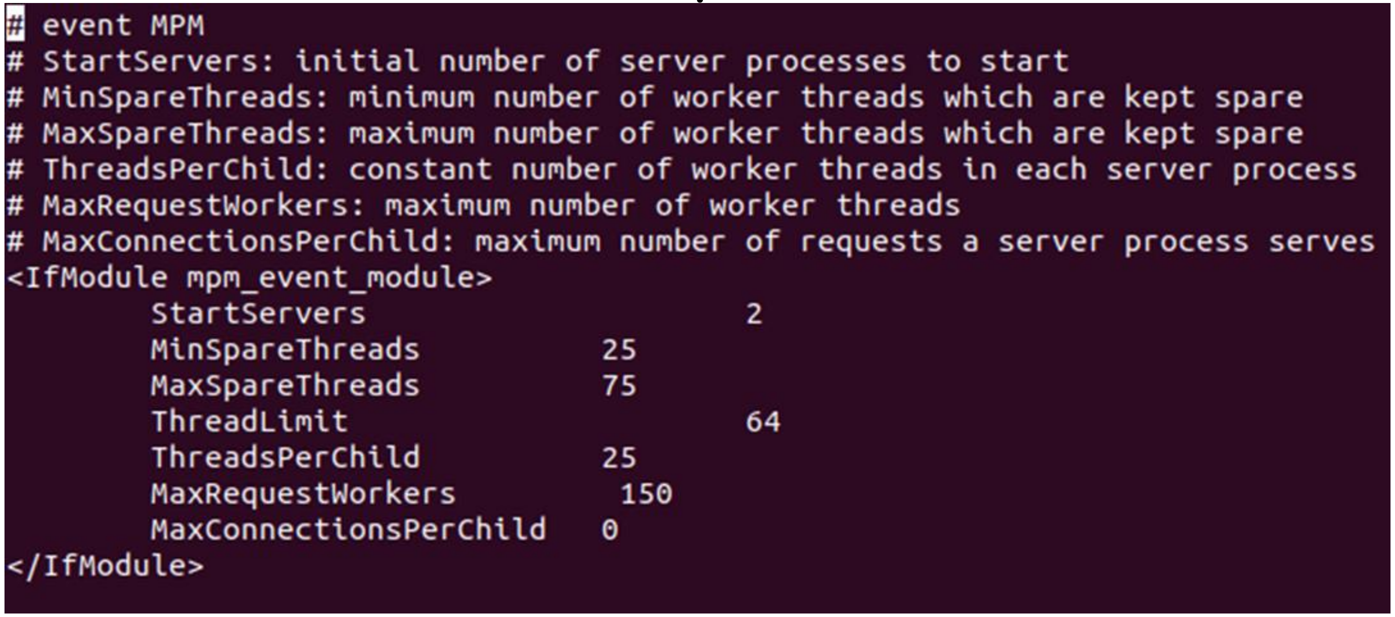
- Analysis shows that the response time increases as the request rate increases; connections do not exhibit resets





Example 2 (cont.)

- In the last version of Apache Web Server, you can set the thread limit by editing the file mpm_event.conf in the directory /etc/apache2/conf-enabled/
  - MaxRequestWorkers directive sets the limit on the number of simultaneous requests that will be served



```
# event MPM
# StartServers: initial number of server processes to start
# MinSpareThreads: minimum number of worker threads which are kept spare
# MaxSpareThreads: maximum number of worker threads which are kept spare
# ThreadsPerChild: constant number of worker threads in each server process
# MaxRequestWorkers: maximum number of worker threads
# MaxConnectionsPerChild: maximum number of requests a server process serves
<IfModule mpm_event_module>
        StartServers                    2
        MinSpareThreads          25
        MaxSpareThreads          75
        ThreadLimit                     64
        ThreadsPerChild          25
        MaxRequestWorkers         150
        MaxConnectionsPerChild   0
</IfModule>
```

Capacity Test: procedure

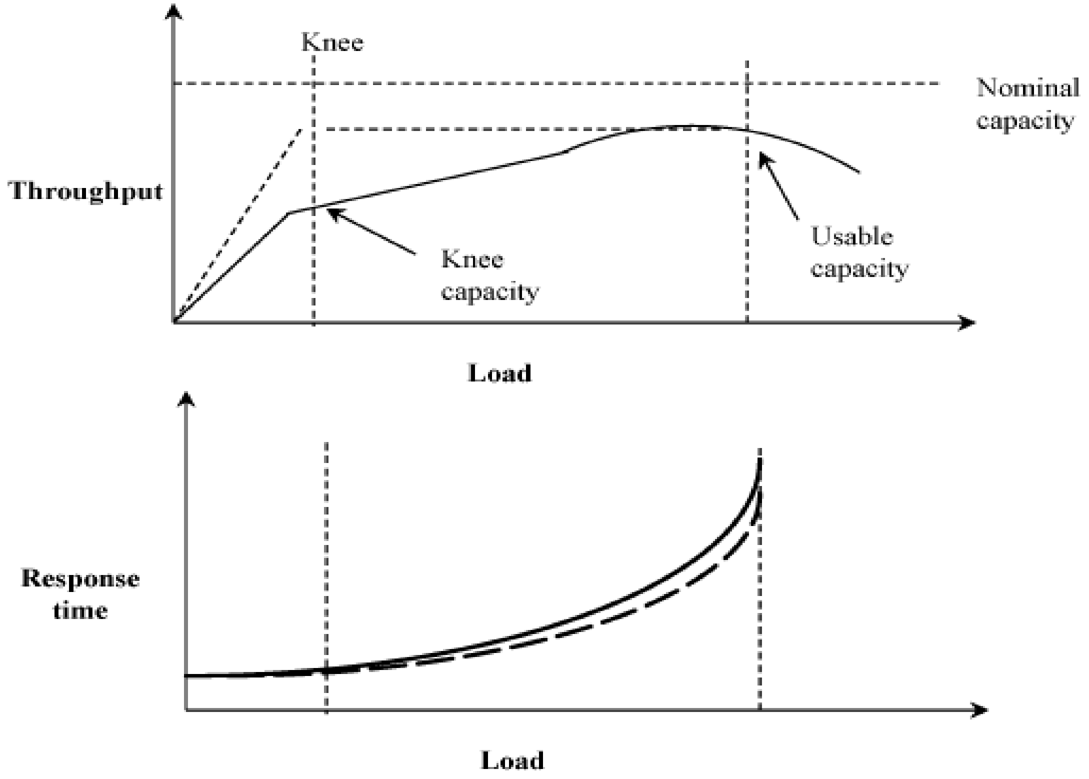
1. Determining a metric to measure the throughput of the system
2. Soliciting the system with an increasing load, in terms of intensity, until a knee in the throughput curve is reached.

Capacity Test: procedure

3. The knee indicates the limit of the system's capacity, since beyond such limit the system is no longer able to serve requests properly
   - As the request rate increases beyond that limit, its throughput does not increase anymore, as it would be expected, but it remains the same or even decreases
4. Since different limits can be reached with different request types, capacity tests can be performed per each request type.
5. Then, either the average of the observed limits, or, to be conservative, the minimum of such limits, is chosen as the system's capacity

Exercise: capacity test and perf. characterization

- 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
- With JMeter, If you want to simulate a given request rate, set it via a Constant Throughput Timer



Exercise: capacity test

- With Jmeter, set the request rate, and test the server for a given time, and repeat for increasing request rates
    - Perform the test and determine the usable capacity and the knee capacity with the random controller considering all the pages (i.e., our “request type”)

Exercise: capacity test

- Repeat the entire test with single request types (at least two), and then:
- Choose as (usable and knee) capacity values the average (over request types) usable and knee capacity
- Choose as (usable and knee) capacity values the worst (over request types) usable and knee capacity
- Compare the result with the random controller case

Experimental Design and Analysis

Objective

- With the previously collected info, design an experiment to study the impact of request factors on the response time
- Use the Design of Experiment technique

Design

- Response Variables
  - Average response time
- Factors
  - Intensity (request rate), Page Type
- Resource Constraint
  - Suppose to be constrained on at most 8 treatments

Design

- Levels
  - Since we analyze only two factors, we can choose several options, e.g.:
    - Group together *intensity* in 2 levels, Low and High, and page type in 4 levels (e.g., 4 different pages)
    - Group together *intensity* in 4 levels, Low, Low-Medium, High-Medium, and High, and page types in 2 types (e.g., with high page size, and low page size, or static and dynamic page)
    - Etc..
  - Intensity levels can be determined in terms of percentage w.r.t. the usable capacity (averaged over page types)
- Repetitions
  - Repeat a treatment N times, with N >= 10 (assume that each repetition lasts for 1 minute, and take the average response time)

Analysis

- Run ANOVA with repetition
  - Objective: Assess which factor (neglecting interactions) is statistically significant, if any.
  - Steps:
    - Verify normality of residuals
    - Verify homoschedasticity
    - Choose the type of analysis (parametric vs non-parametric) and thus the corresponding test (F-test, Kruskal-Wallis, Welch)

Analysis of Residuals: guidelines

- Conduct first a Visual Analysis
- Then, run the Shapiro-Wilk (n<2000) or KSL test (n>2000) for checking normality and Levene's test for homosch.
- Since a parametric ANOVA is considered a robust test against the normality assumption, "small" violations of normality are still accepted. Therefore:
  - If the visual test indicates normality and Shapiro-Wilk (S-W) confirms the normality (*p-value>0.05*), go for a parametric ANOVA
  - If the visual test indicates normality and S-W does not confirm the normality, but with *0.01< p-value < 0.05*, go again for a parametric ANOVA (i.e., rely more on visual test than on a 1/0 test like S-W)
  - If the visual test indicates normality and S-W does not confirm the normality with *p-value < 0.01*, go for a non-parametric ANOVA
  - If the visual test indicates non-normality, go for non-parametric ANOVA

Choice of the Analysis

- In the case of normal and homoschedastic data use the classical ANOVA F-test
- If data are normal but not homoschedastic use the Welch's ANOVA test
- If data are not normal (regardless the homoschedasticity) use the Wilcoxon/Kruskal-Wallis test

– *We skip the final post hoc analysis, assuming it's not of interest in our case*