Design Patterns

MSc Computer Science

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Enterprise Web Performance

Performance

- Response Time
- Responsiveness
- Latency
- Throughput
- Load
- Load Sensitivity
- Efficiency
- Capacity
- Scalability

Response Time

- Amount of time it takes for the system to process a request from the outside.
- This may be a UI action, such as pressing a button, or a server API call.

Responsiveness

- How quickly the system acknowledges a request as opposed to processing it.

- Important if users may become frustrated if a system has low responsiveness, even if its response time is good.

- If the system waits during the whole request, then your responsiveness and response time are the same.

- If systems indicates receipt of the request before completion, then responsiveness is improved.

    - e.g. providing a progress bar during a file copy improves the responsiveness of your user interface, even though it doesn't improve response time.

Latency

- Minimum time required to get any form of response, even if the work to be done is nonexistent.
- Asking a program to do nothing, but respond to the user that it has done doing nothing, should be almost instantaneous.
- However, this is only the case if program runs locally (on device).
- If the program runs on a remotely, a few seconds required to make the request and process response.
- Latency beyond control - so attempt to minimize remote calls.

Throughput

- How many tasks/transactions can be done in a given amount of time.
  - e.g. copying of a file, throughput might be measured in bytes per second.
- For enterprise applications a typical measure is transactions per second (tps),
  - but the problem is that this depends on the complexity of your transaction.

Performance

- Performance may be measured as
  - throughput,
  - response time
  - responsiveness
- Difficult to talk about performance when a technique improves throughput but decreases response time
- From a user's perspective responsiveness may be more important than response time, so improving responsiveness at a cost of response time or throughput will increase performance

Load

- How much stress a system is under, which might be measured in how many users are currently connected.
- The load is usually a context for some other measurement, such as a response time.
- Thus, you may say that the response time for some request is 0.5 seconds with 10 users and 2 seconds with 20 users.

Load Sensitivity

- An expression of how the response time varies with the load. Example:
  - System A has a response time of 0.5 seconds for 10 through 20 users
  - System B has a response time of 0.2 seconds for 10 users that rises to 2 seconds for 20 users.
- In this case system A has a lower load sensitivity than system B.

… also expressed as system B degrades more than system A.

Efficiency

- Performance divided by resources.
  - eg A system that gets 30 tps on two CPUs is more efficient than a system that gets 40 tps on four identical CPUs.

Capacity

- An indication of maximum effective throughput or load.
- This might be an absolute maximum or a point at which the performance dips below an acceptable threshold.

Scalability

- A measure of how adding resources (usually hardware) affects performance.
- A scalable system is one that allows you to add hardware and get a commensurate performance improvement, such as doubling how many servers you have to double your throughput.
- Vertical scalability, or scaling up, means adding more power to a single server, such as more memory.
- Horizontal scalability, or scaling out, means adding more servers.

Comparison: Swordfish V Camel - 1 Server

- On 1 server Swordfish's capacity is 20 tps while Camel's capacity is 40 tps.
- Which has better performance? Which is more scalable?

- We can't answer the scalability question from this data, and we can only say that Camel is more efficient on a single server.

Comparison: Swordfish V Camel - 2 Servers

- On 2 servers Swordfish's capacity is 25 tps while Camel's capacity is 50 tps.
- Which has better performance? Which is more scalable?

- Camel's capacity is still better, but Swordfish looks like it may scale out better.
- If we continue adding servers we may discover that Swordfish gets 15 tps per extra server and Camel gets 10.
- Given this data we can say that Swordfish has better horizontal scalability, even though Camel is more efficient for less than five servers.

Strategies - Scalability vs Efficiency

- It may make sense to build for hardware scalability rather than capacity or even efficiency.
- Scalability gives you the option of better performance if you need it and may be easier to do (just add hardware). Software efficiency improvements may be complex and expensive to implement
- Adding new hardware is often cheaper than making software run on less powerful systems. Similarly, adding more servers is often cheaper than adding more programmers—providing that a system is scalable.