Kenny R N Wilson

Non-Functional Requirements

Reliability

A reliable system is one in which faults do not lead to failure. We say the system is fault-tolerant or resilient. Faults can be classified as hardware faults, human errors, and software errors.

Fault Types

Hardware Faults

Hard disk crashes, memory faults and power outages are all good examples of hardware faults. We can arrange our disks in raid configuration, duplicate power supplies etc to minimise the chance of a single machine going down, however modern systems are moving more and more to architectures that are tolerant of whole machines going down.

To calculate availability, see the following.

https://static.usenix.org/event/osdi10/tech/full_papers/Ford.pdf

Software Faults and Human Error

The only way to deal with software faults is to perform careful testing. Approaches to minimising human error include building well designed interfaces and providing non-production sand box environments that users can practice on. I will not talk about software faults or human error in this document.

Scalability

The scalability of a system measures how easily we can add hardware to improve performance. Vertical scalability or scaling up involves adding hardware to a single server. Horizontal scalability or scaling out involves adding more servers. We measure scalability with the goal of answering two questions: what happens to performance when we keep system resources constant and increase load; and if we increase load how much we need to increase system resources to keep the level of performance the same.

To talk about scalability, we need to be able to measure how the performance of a system is affected by increases in load. To describe load, we need to choose a relevant load parameter. Some typical load parameters are as follows.

- Requests per second
- Ratio of reads to writes.
- Hit rate on a cache.
- Concurrent users.

Once we have decided on the relevant load parameter, we need to look at what happens to performance when the load parameter increases. Typical measures of performance include.

- Throughput – how much work per unit of time e.g., bytes per second.
- Response Time – elapsed time from sending request to receiving a response.
- Latency – minimum time to get a response.

Once we have determined our load parameter and performance measure, we can start making statements such as "our system has a response time of 0.25 seconds when we have 200 concurrent users, and this rises to 1.0 seconds with 2000 concurrent users. We can use the following descriptive statistics.

- Load Sensitivity – How does performance degrade as load increases.
- Efficiency – Performance / Resources
- Capacity – How much we can increase load and keep acceptable performance.

Statistical Considerations

Response time averages can hide outliers. Page faults, garbage collection and dropped packets are all causes of increases in response time. Typically we should prefer to use percentiles rather than averages.