The scalability of software systems refers to the upper limits of its capacity for data processing, data storage, and data access. The objective for software system scalability is to 1) maximize the upper limits of this capacity and 2) provide an incremental means to increase capacity from very small to very large scales. Systems should be able to start as a single computer to minimize initial costs, and then be able grow to whatever size is needed without any changes to the software architecture or code of the system. The upper limits of scalability of a system should be well beyond the needs of most business so that software systems will never need to be redesigned and rebuilt with a different architecture, platform, language, etc. to meet future scalability requirements.

Requirements

1. The system should have a wide range of scalability, from very small to very large.

What: *The system should be able to run well on a single computer, and also scale up and out to a large enterprise level, with only changes to endpoint configurations at the different scales.*

Why: *Running an entire system on a single computer is needed for software development, to quickly debug any problems, and for small scale production environments. The upper limits of scalability should be beyond the needs of almost all software systems so that it will never have to be rewritten to use a different architecture, tools, etc. as it grows in size.*

Testing: *Testing small scale deployments can be done using any development computer. Testing large scale deployments will generally have to be extrapolated from the measurements of smaller scale deployments.*

2. The design should provide horizontal scalability to increase the processing power of the system.

What: *The processing power of a system should allow for incremental horizontal scalability (scale out).*

Why: *The scalability of the processing power of the system is handled by the load balanced server farms which provide incremental horizontal scalability, and also fault-tolerance when extra nodes are added to a server farm.*

Testing: *Testing the scalability of a system using stress and/or load tests is typically very difficult and expensive. A more practical approach is to measure the resource utilization of a system during its actual use, and extrapolate a system's scalability from those measurements.*

3. The design should provide vertical scalability to increase the storage capacity of the system.

What: *Data storage in general does not scale horizontally like processing power, and therefore storage nodes (database servers) will need the ability to scale vertically to levels beyond the needs of the business.*

Why: *The solution to handle database scalability is to over-engineer the database servers themselves, with massive amounts of TB and IOPS that a system can grow into over time. Partitioning of databases onto their own separate hardware using natural divisions is another way to increase the scale of data storage short of database sharding.*

Testing: *Monitoring the storage and resource utilization of database servers is the best way to test the scalability of the data storage in a system.*

4. (Optional) The design should provide horizontal scalability to increase the storage capacity of the system.

What: *Horizontal scalability of data storage involves some type of data shard mechanism (both vertical and horizontal data partitioning) for relational databases, or the use of other types of data storage such as key/value pairs, etc.*

Why: *Horizontal scalability via shards for relational data is very desirable for large scale systems. The estimated total volume of data over time will determine if horizontal scalability is needed, or if growth can be handled with vertical scalability alone. Any shard mechanism must be compatible with DGP's omnidirectional data replication..*

Testing: *Depends on the tool and mechanisms used.*

5. All critical subsystems (such as security for example) must themselves be able to scale to keep up with the growth of the system as a whole, and perform well at all relevant scales.

What: *Scalability for a modular N-tier system is not monolithic. Each tier and subsystem will have its own scalability issues.*

Why: *The overall scalability of a system will be determined by the tier or subsystem with the lowest scalability.*

Testing: *A small scale environment can be used to test scalability limitations of each subsystem, and those measurements can then be extrapolated to estimate the scalability of the system as a whole.*