**Lab 3 Writing Assignment**

[**Scale in Distributed Systems** by **B. Clifford Neuman**](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.5514&rep=rep1&type=pdf)

The paper looks at the aspects of how scaling affects distributed systems. In the modern day, system is set to scalable if it could handle adding of multiple resources and users without suffering a visible loss in performance, or increase in complexity of administration. Scalability could be seen in 3 aspects:

1. Size Scalability : Number of users and resources part of the system
2. Geographical Scalability : maximum distance between nodes in the system
3. Administrative scalability : Number of administrative domains controlling the pieces of system

**Definitions**

* A **distributed system** is a collection of computers, connected in a network and working together as one to implement minimum set of services.
* A service/resource is set to call **replicated** when it has several identical instances occurring on different nodes in the system.
* A service is set to call **distributed** when it is provided by several nodes in the network. Each node capable enough to handle a subset of the requests for the service. A distribution function maps is used to handle these requests and forward it to the node.
* **Caching** could be seen as temporary form of replication. It is used to save and reuse of query results on nodes for the faster execution of the requests.

**Problems and Solutions in Distributed System:**

1. **Reliability**

As the number of components increases in system, chances of all being working together at any point of time decreases. Also, it would become less likely that nodes will be able to communicate among each other. This could be resolved by increasing the autonomy of nodes in the system (i.e. making set of nodes work independently without any worry about failure of other sets of node), and adding replication.

1. **System Load**

With increase in the amount of nodes, data and services, the system query load also gets increased. This issue could be resolved using replication, distribution and caching. Distribution and Replication spreads the request to multiple servers. Caching helps in reducing the repeated requests.

1. **Administration**

Increase in the number of nodes also creates complexity in administrating of users, services and system.

Keeping the common information centrally could help in such situations as maintain multiple copies of user administrative data will increase issues and bring overhead of maintenance.

1. **Heterogeneity**

Scaling of the nodes in the system comes with challenges of heterogeneity i.e hardware and software o different nodes could be different. To handle this situation we need approach of coherence. This approach expects nodes to support a common interface.

* All nodes should have common instruction sets.
* All nodes should have similar execution abstraction(i.e. two nodes could easily compile the same program)
* All nodes should support common set of protocols.

**Distributed system components affected by Scaling:**

* Naming and directory services
* Authentication
* Authorization
* Accounting
* Communication
* Remote Resources

Scaling of all these components could be improved by using replication, distribution and caching techniques.

**Goals and Mechanism of Scalability:**

Goal of Scalability is

* How it could provide high reliability and availability of services and data at any moment of time.
* How could it reduce the latency per user request and how quickly it could perform the operation without any noticeable loss of performance.

Different Mechanism:

* Replication
  + Replication is an important resource for building scalable system. It uses naming, authentication and file services to reduce the load on selective servers. It also improves the reliability and availability of the services.
  + Placement of replicas also matters as it tries to resolve availability of the service or for reducing the network delays for accessing the system.
  + Use loose consistency as it not mandatory for all the application to be absolute consistent. It also guarantees that all replicas are eventually will contain identical data.
* Distribution
  + It facilitates the information maintained by distributed services to spread across multiple servers.
  + Request distributed evenly to servers in proportions to their power. Not having such distribution makes one server idle and other being overloaded at same moment.
  + Exploit locality gives us an edge in distributed systems. Network traffic and latency could be reduced if data and services are stored to servers that are close to the location of frequent usage.
  + In hierarchical system, avoid moving to upper level of hierarchy of root for information. Maintaining cached copies on lower level hierarchy reduces latency and increase distribution.
* Caching
  + **Cache frequently accessed data** reduces the load on server and network.
  + **Consider access patterns when caching** for example amount of data accessed together, read to write ratio, likelihood of conflicts and number of simultaneous users.
  + **Cache timeouts** is a way to set time-to-live with cached data an upper bound could be placed on the time required for changes.
  + **Caching at multiple levels** helps in reducing the number of requests to next level.
  + **Look first locally** for copies of data before checking with central servers, reduce the load time.
  + **Data cached extensively must be changed less frequently**
* Avoid global broadcast as it does not scale well. It requires multiple system to process the response even though, they don’t need to.
* Shed load but not too much: perform computation where it suits better
* Support multiple access mechanisms for the services and data
* Keep users in mind

**System Evaluation for Scaling:**

Evaluating Distributed Systems is a tedious task. One system may scale administratively better while other scale numerically. Sets of evaluation questions are as follows,

* Based on how the system is use
  + How the queries grows with the system.
  + In case queries grow, what percentage of queries needs to be handled by central servers.
  + Ideal number of central server in the system, and
  + Issues which could arise with their(central server) replication
* Concerning with Data
  + With Increase in data size, how its maintained in each node in the system
  + Does it Increase the query time with data size
  + Which data update process is used and how it scales
  + When will be the Cache data invalidation and how it increases the request queries.
* Administration Component of Scale
  + Does the system require a centralized system admin?
  + Is it practical in the environment in which the system is used?

**Evaluation Point View:**

I believe the paper has done evaluation of Scaling Distributed system quiet fairly. It has added multiple examples to demonstrate both the problems and their solution. Few places in the paper, discussion went into subsystem discussion too deep rather than evaluating the system as a whole. It has also shown how scale affects the large system, and how each problem that we come across could be more readily is solved by repeated use of replication, distribution and caching.

**Please Note**: The paper doesn’t talk about two questions mentioned in the lab-3(Earlier Proposed Solutions and Aspects of Problem that are not addressed). I discussed the same with professor as well, so would not be to answer them. This paper is more of bringing up common understanding among everyone rather than creating and supporting one new aspect in distributed system.