Distributed system requirements

Remco Hobo

December 1, 2004

In this report I will give a description about some keywords that have to do with Distributed Systems:

- Scalability
- Openness
- Heterogeneity
- Resource Sharing
- Fault-Tolerance
- Access transparency
- Location transparency
- Migration transparency
- Replication transparency
- Concurrency transparency
- Scalability transparency
- Performance transparency
- Failure transparency

Scalability

A distributed system can exist of a huge number of computers, or just a few. Scalability means the system is easily scalable. The system should remain stable when the number of computers (nodes) changes. A system is scalable when

- The number of nodes can vary without affecting the stability.
- The number of nodes can vary without the performance dropping below pre-defind thresholds
- Nodes can be added and removed without too much human intervention

Openness

Openness means the level of open standards with which the distributed system is built. A good distributed system is well-documented and is adaptable. If a programmer wants to thange or add something to the system, this should be easy to do. For this, creating simple interfaces and good resource sharing is paramount. Using open standards can make the system very adaptable and flexible.

Heterogeneity

Heterogeneity means the system has to be able to run on a wide range of hardware and software platforms. This means it has to run on a wide range of:

- Networks
- Computer hardware
- Operating systems
- Programming languages

Normally heterogeneity is accomplished by the use of middle ware like CORBA of Java RMI

Fault-Tolerance

A proper fault-tolerating distributed system can handle errors from nodes. If a node goes down, if a node becomes unstable, the system has to cope with that and remain stable. The system also has to function properly when the network becomes congested, becomes high-latency or even goes down. Mission-critical systems still have to function if all connectivity to all other nodes is disrupted. Some techniques are:

- Detect failures. Some failures can be detected by checking the data's CRC. Also, when for instance a mathematical calculation has been performed, the system can do a rudimentary check if the result can be correct.
- Retransmitting. If a message has been lost along the network, the message can be retransmitted.
- Buffering. Some critical data can be stored, so that if a network failure occurs, the node can continue to function.
- Redundancy. Sometimes fault-tolerance is of such priority it may impact performance. In such instances operations an be redundant, more nodes may do the same thing or network paths are layed out redundant.

Transparecy

Transparency means that the whole distributed system is seen as a whole instead of a lot of different components. Forms of transperency are:

- Acces transparency means data can be accessed locally or remote using the same operations. The system takes care of differences in data types. A linux user can access an NTFS file without knowing it isn't a normal linux file.
- Location transparency means that resources are accessible from anywhere in the distributed system. Users can access their data from a local node or from abroad with the same ease. The user isn‘t aware where the data actually comes from. A user in America can access a file in the Netherlands without noticing it.
- Migration transparency means resources can be moved within the distributed system without being lost. If a user‘s mailbox is stored somewhere else in the distributed system, the references will be modified as well, so the user won‘t notice the data has migrated.

- Replication transparency means replicating data in the distributed system without affecting the operation parameters of the system. If a user file is stored in the Netherlands and in France, if the user modifies the file in France, the file in the Netherlands should also be modified without user intervention.

- Cucurrency transparency. Multiple processes have to be able to use the same resources without affecting the operation parameters of the system. If five users use the same word editor, it should work the same as if only one person used it.

- Scalability transparency means the distributed system has to be scalable. The number of nodes, network paths etc. should be able to change without affecting the systems operations. If a distributed system that calculates primes, gets more nodes, the system should be able to incorporate them into the system easily. Also when nodes are taken offline the system has to be able to redistribute their work to other nodes.

- Performance transparency. The system should perform the same under a little load as under heavy load. If a process is generating heavy load because it is calculating a difficult equation, it has to run at less priority then a word editor where some person is working with. This way, the user‘s experience is the same.

- Failure transparency. Some components of the distributed system should be able to fail without affecting the distributed systems operation parameters. If a router fails, a backup router can take over so network integrity is restored.

References

[1] Distributed Systems Concepts and Design, Addison Wesley, ISBN 0-201-61918-0