ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Scheming various Architectural Designs to improve Efficiency, Scalability and Fault-Tolerant in Distributed Computing

Surya Pavan Kumar Gudla
Aditya Institute of Technology and Management, Tekkali, Andhra Pradesh, India.

N. Preeti
Aditya Institute of Technology and Management, Tekkali, Andhra Pradesh, India.

Abstract:

Distributed computing has been Holy Grail in the field of computer science that studies distributed systems. Distributed system located on networks communicates and coordinates their actions by passing messages over the components in network. Every system on network has an underlying architecture (topology) in order to achieve a common goal on sharing of resources. As distributed application involves only flow of data, requires highly sophisticated architecture with various characteristics like performance, reliability and many others. To involve exchange of information among application running across computer networks requires appropriate system topology. With a view towards, we compare and determine best topology for distributed computing with respect to its characteristics.

Keywords:
Distributed system, Topologies, Design, Architecture.

I INTRODUCTION:

Distributed computing is limited to programs with components shared among computers within a limited geographic area. Broader definitions include shared tasks as well as program components. In the enterprise, distributed computing has often meant putting various steps in business processes at the most efficient places in a network of computers. For example, in the typical distribution using the 3-tier model, user interface processing is performed in the PC at the user's location, business processing is done in a remote computer, and database access and processing is conducted in another computer that provides centralized access for many business processes. Typically, this kind of distributed computing uses the client/server communications model.

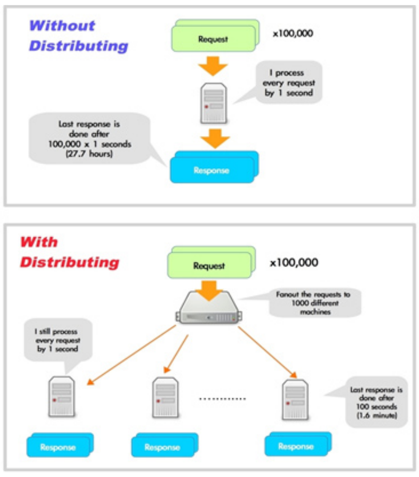
II CHARACTERISTICS:

Characteristics to be achieved in the construction of distributed systems.

Virtually all large computer-based systems are now distributed systems. Information processing is distributed over several computers rather than confined to a single machine. Distributed software Engineering is therefore very important for enterprise computing systems.
Resource sharing: Sharing of hardware and software resources.
Openness: Use of equipment and software from different vendors.
Concurrency: Concurrent processing to enhance performance.
Scalability: Increased throughput by adding new resources.
Fault tolerance: The ability to continue in operation after a fault has occurred. Ending on the system organization.

A) Properties of Distributed Systems:

Distributed Systems are made up of 100s of commodity servers. 1) No machine has complete information about the system state. 2) Machines make decisions based on local information. 3) Failure of one machine does not cause any problems. 4) There is no implicit assumption about a global clock
Examples of Distributed Systems• Amazon's e-retail store• Google• Yahoo• Facebook• Twitter• YouTube.


Without Distributing
Request
x100,000
I process every request by 1 second
Last response is done after 100,000 x 1 seconds (27.7 hours)
Response
With Distributing
Request
x100,000
Fanout the requests to 1000 different machines
I still process every request by 1 second
Last response is done after 100 seconds (1.6 minute)
Response
Response
Response

Why Need Distributed Computing

III DESIGN VS ARCHITECTURE

Design explicitly addresses functional requirements:

Architecture explicitly addresses functional and non-functional requirements such as: Reusability, Maintainability, Portability, Interoperability, Testability, Efficiency, and Fault-Tolerance and the other Quality Attributes

A) Asynchronous: Since connections and applications are intermittent, service requesters and providers may not be accessible at the same time, necessitating support for asynchronous communication.

B) Atomic: After a transaction executes, either all or none of its operations take effect.

C) Awareness: Highly distributed, decentralized, mobile, applications must be aware of their execution context in order to properly adapt to any context changes. For this reason, the middleware must be able to monitor and inform the running application of its execution context.

D) Behavior: Dependencies between events.

E) Causality: Being able to depict the causal history between events time, and response process execution time.

F) Component based Style: Software components may be written in different programming languages. They can more readily be reused and/or substituted with other components in an architecture.

G) Delivery guarantees: In resource-constrained and embedded systems, the utility of a service often directly depends on how many and at what times it has been delivered. The middleware should thus provide support for service delivery guarantees and real-time constraints.

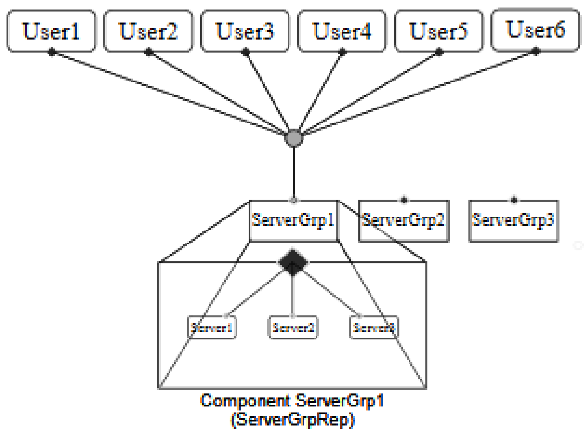
H) Security: All nodes in a (changing) network comprising an application cannot be trusted a priori. For this reason, the middleware should support secure communication between components.

IV MEASURES:

How do determine (measure) which one is better at capturing: Functional relationships and Distribution? 1) Data Topology (Distribution) & Flow? 2) Control Topology Flow? 3) Event Handling? 4) Fault-Tolerance? How do measure which one is better at addressing the issues wrt. Homogenous vs. Heterogeneous systems? 1) Expandability? 2) Interoperability? 3) Maintainability? 4) Upgradeability? 5) Responsiveness? Etc.

V ARCHITECTURES:

The centerpiece of our approach is the use of architectural models. We use a simple scheme in which an architectural model is represented as a graph of interacting components.



Nodes in the graph are termed components. They represent the principal computational elements and data stores of the system: clients, servers, databases, user interfaces, etc. Arcs are termed connectors, and represent the pathways of interaction between the components. A given connector may in general be realized in a running system by a complex base of middleware and distributed systems support.

A) Styles of designing architecture based on requirement:

If your system involves controlling continuing action, is embedded in a physical system, and is subject to unpredictable external perturbation so that preset algorithms go awry: Then consider a closed loop control architecture. If you have designed a computation but have no machine on which you can execute it: Then consider an interpreter architecture. If your task requires a high degree of flexibility, configurability, loose coupling between tasks, and reactive tasks: Then consider interacting processes. If you have reason not to bind the recipients of signals from their originators: Then consider an event architecture. If the tasks are of a hierarchical nature:

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Then consider replicated worker or heartbeat style. If the tasks are divided between producers and consumers: Then consider client/server. If it makes sense for all of the tasks to communicate with each other in a fully connected graph: Then consider a token passing style.

B) Architecture Representations:

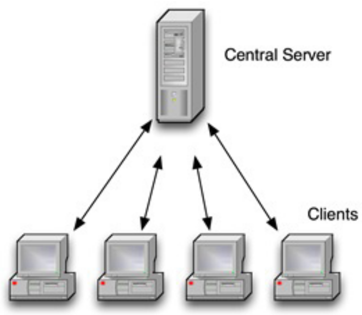
Static View vs. Dynamic View "Current software architecture research assumes a system's architecture is static, it does not evolve during execution." Architectures must be able to evolve during execution. –"First, the architectures of many existing systems change during execution, and are poorly modeled using existing techniques. Examples include systems built using OLE, OpenDOC, or CORBA, in which new components may be loaded and unloaded during execution." "Second, many systems would benefit from the dynamism afforded by a dynamic architecture. Examples include systems characterized as long running and mission critical since the delays and risks associated with shutting down these systems for upgrades may be expensive." Examples of Real World Problems (Internet?).

V ARCHITECTURES FOR DISTRIBUTED COMPUTING:

Various hardware and software architectures are used for distributed computing. At a lower level, it is necessary to interconnect multiple CPUs with some sort of network, regardless of whether that network is printed onto a circuit board or made up of loosely coupled devices and cables. At a higher level, it is necessary to interconnect processes running on those CPUs with some sort of communication system.

A) Client- Server architecture:

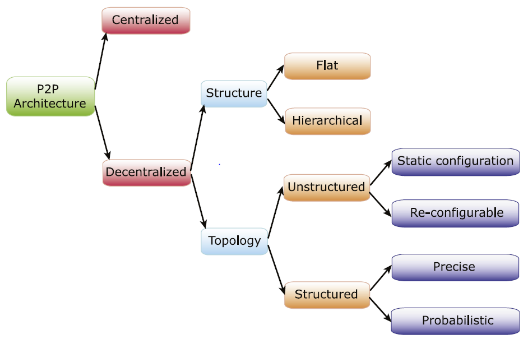
It forms the most popular system topology as like Centralized server. All functions and information is centralized on single server with many client. Both data and control flow takes place through central server.The primary advantage of centralized server is its simplicity. As all data is concentrated in single server and no question of consistency and coherence.The drawback of centralization is that all information resides at hub, when suddenly leads to single-point-of-failure and its client application connected to this hub also dies. Thus it is a bottleneck to scalability and performance. Hence centralized server is unsuitable to distributed application deployment.



Client / Server

B) Peer-to-Peer Architecture:

Peer-to-peer (P2P) computing has been hailed as a promising technology that will reconstruct the architecture of distributed computing (or even that of the Internet). This is because it can harness various resources (including computation, storage and bandwidth) at the edge of the Internet, with lower cost of ownership, and at the same time enjoy many desirable features (e.g., scalability, autonomy, etc.). The taxonomy of P2P architectures based on existing systems, on one extreme, some P2P systems are supported by centralized servers. On the other extreme, pure P2P systems are completely decentralized. We begin by looking at a taxonomy of P2P systems. This taxonomy is derived from examining existing P2P systems.

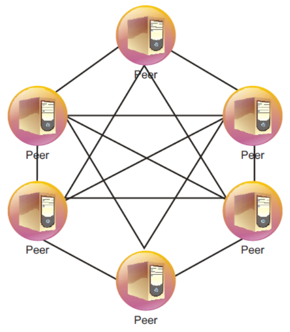


Taxonomy of P2P Architecture

There are three models of unstructured P2P computer network architecture: Pure P2P, Hybrid P2P and Centralized P2P.

1) Pure P2P systems:

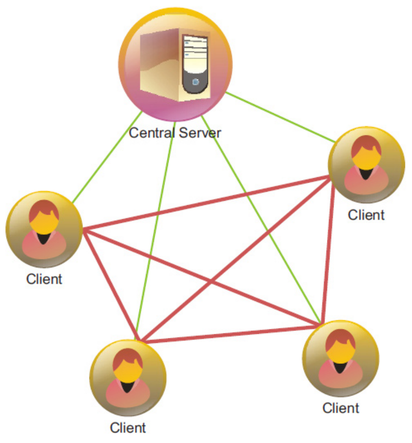
A Primary virtue of P2P systems is their scalability; any node can join a network and start exchanging data with any other node. Decentralized systems also tend to be fault tolerant, as the failure or shutdown of any particular node does not impact the rest of the system.



Peer-to-Peer System

2) Hybrid P2P:

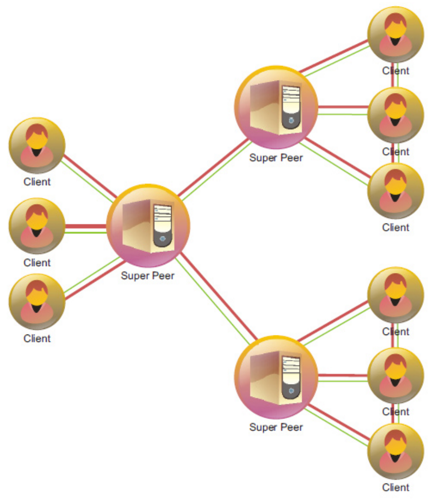
Hybrid. Peer-to-Peer networking concept, which allows the existence of central entities in its network, and the Pure Peer-to-Peer networking concepts within which Servents are the only entities allowed. This architecture alleviates the manageability problems of pure p2p systems. The control server acts as a monitoring agents for all the other peers and ensures information coherence.



Hybrid Peer-to-Peer System

C) Super P2P System:

A super-peer network operates exactly like a pure P2P network, except that every "node" is actually a super-peer, and each super-peer is connected to a set of clients. Clients are connected to a single super-peer only. We call a super-peer and its clients a cluster, where cluster size is the number of nodes in the cluster, including the super-peer itself.



Super Peer-to-Peer system

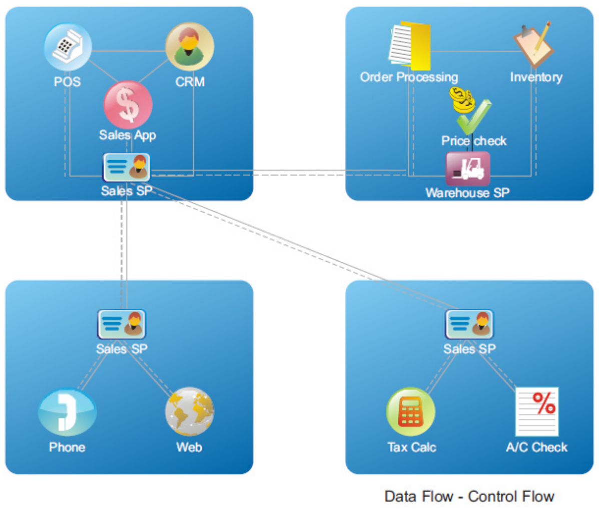
V COMPARISION OF TOPOLOGY

Topology	Manageable	Coherent	Scalable	Reliability
Centralized	Yes	Yes	No	No
De-centralized	No	No	Yes	Yes
Hybrid P2P	Yes	Yes	Yes	No
Super Peer	Yes	Yes	Yes	Yes

VI Business Case:

When a super-peer receives a query from a neighbor, it will process the query on its clients' behalf, rather than forwarding the query to its clients. In order to process the query for its clients, a super-peer keeps an index over its clients' data. This index must hold sufficient information to answer all queries. For example, if the shared data are files and queries are keyword searches over the file title, then the super-peer may keep inverted lists over the titles of files owned by its clients. If the super-peer finds any results, it will return one Response message. This Response message contains the results, and the address of each client whose collection produced a result.

In order for the super-peer to maintain this index, when a client joins the system, it will send metadata over its collection to its super-peer, and the super-peer will add this metadata to its index. When a client leaves, its super-peer will remove its metadata from the index. If a client ever updates its data (e.g., insertion, deletion or modification of an item), it will send this update to the super-peer as well. Hence, super-peer networks introduce two basic actions in addition to query: joins (with an associated leave), and updates.When a client wishes to submit a query to the network, it will send the query to its super-peer only. The super-peer will then submit the query to its neighbors as if it were its own query, and forward any Response messages it receives back to the client. Outside of the cluster, a client's query is indistinguishable from a super-peer's query.



Super Peer Topology:

The super peer architecture truly combines the virtues of centralized and decentralized systems, it can alleviates the problems associated with it. Moreover, since there are few controllers in the system, configurations is no longer a problem. The overall system is more secure since multiple controller's controls the flow of data by each of the clients. Here overall workload is distributed among multiple peers, making the system infinitely scalable.

VII CONCLUSION:

Designing of architecture and establishing interconnection with the components in the network, researchers have been made great efforts. we presented a summary of architectural issues of P2P systems, such that researchers, developers, and users are able to see clearly the potential merits of different P2P systems, identify the key architectural factors that decide the system performance, and make appropriate implementation decisions. In terms of the degree of decentralization, the architectures of P2P systems can be generally classified into three categories: centralized P2P systems, decentralized P2P systems, and hybrid P2P systems. Comparison have brought between the systems considering the characteristics. Combining the above architectures presented a super peer architecture, which combine the efficiency of the centralized client-server model with the autonomy, load balancing and robustness of distributed search. They also take advantage of heterogeneity of capabilities across peers. Because well-designed super-peer networks promise large performance improvements for P2P systems.

VIII REFERENCES:

[1] Designing a Super-Peer Network Beverly Yang Hector Garcia-Molina fbyang, hectorg@cs.stanford.edu Computer Science Department, Stanford University.
[2] Architecture of Peer-to-Peer systems. Peer-to-Peer Computing, Principles and Applications.
[3] P. Goelle, K. Keyton-Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks. In Proc. of ACM Conference on Electronic Commerce, October 2001.
[4] Super Peer Architectures for Distributed Computing, www.fiorano.com.
[5] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In Proc. of the 28th Intl. Conf. on Distributed Computing Systems, July 2002.
[6] B. Yang and H. Garcia-Molina. Comparing hybrid peer-to-peer systems. In Proc. of the 27th Intl. Conf. on Very LargeDatabases, September 2001.
[7] B. Yang and H. Garcia-Molina. Improving efficiency of peer-to-peer search. In Proc. of the 28th Intl. Conf. on Distributed Computing Systems, July 2002.

BIOGRAPHIES:

Mr. Surya Pavan Kumar Gudla
Received his MCA and MTech in CSE Engg from Aditya Institute of Technology and Management, Tekkali. He is currently working as Asst.Prof. in Department of Computer Science & Engineering, in Aditya Institute of

Technology and Management, Tekkali, Andhra Pradesh, India. He has 5 years of experience in teaching MCA and Computer Science and Engineering related subjects. His Interested areas are Computer Networks, Mobile Computing and Data Mining.

Mrs. N.Preeti

Received the MCA from Maharaja Post Graduate College, Vizianagram and MTech in CSE Engg from the Department of Computer Science and Engineering in Aditya Institute of Technology and Management, Tekkali. She is currently working as Asst.Prof. in Department of Computer Science & Engineering, in Aditya Institute of Technology and Management, Tekkali, Andhra Pradesh, India. She has 8 years of experience in teaching MCA and Computer Science and Engineering related subjects. Her Interested areas are Computer Networks, Mobile Computing and Distributed & parallel computing.