

Serving Reliable Resource Discovery in Grid Environment

**In-Cheol Jeong*, Tae-Gun Kang*,
Ho-Sang Ham***

***Mobile Distributed Computing
Research Team, ETRI**

Abstract

One of the most important functionalities in large scale distributed computing environment, such as grid systems is how to lookup resources. It would not be very big challenge under the environment of sharing resources in small scale distributed computing system or within a single organization that runs central servers in order to manage and share resources stored with predefined structure. However it would be prerequisite feature to provide efficient and fast lookup service for members to join and leave freely in large scale distributed system like grid environment. This paper describes a system that adapts Chord system which has been used as distributed resource search technique in grid environment and message queue to maintain reliability in using this technique. The proposed system is designed to integrate message queue for the purpose that resource lookup requests from an application maintain consistency with previously stored.

Introduction



- Grid has been focused as one of the most important technologies in distributed computing environment
- The most important function in large-scale grid environment is resource discovery
- Traditional lookup, i.e. LDAP, service maintains all information is normally based on central DBMS
- Benefit of Chord System in Grid Environment
 - supported peer-to-peer application in distributed computing
 - Ensure the integrity of location information when nodes are joined or left arbitrarily

APNOM 2000

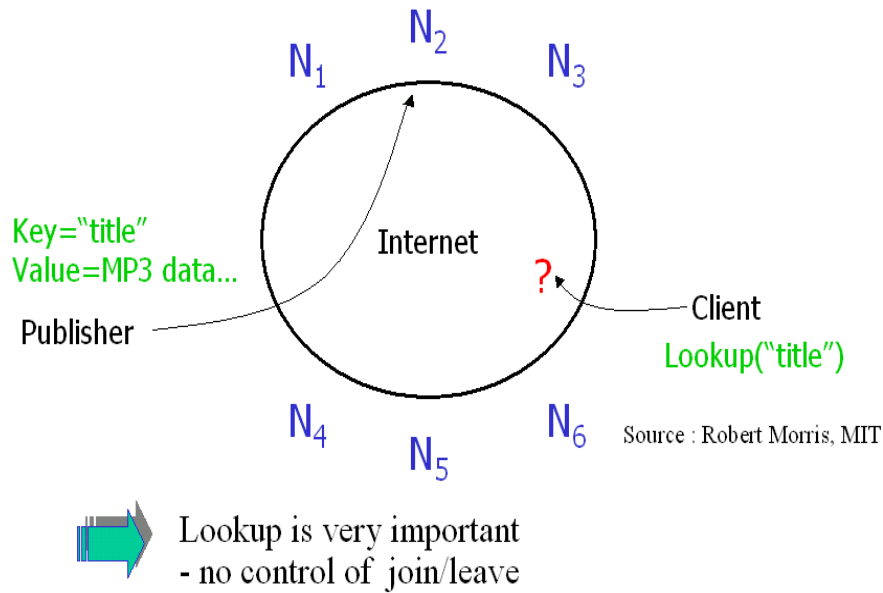
(2)

1. Introduction

As computing resources have grown and technologies have developed, distributed computing technology became to require to fully utilize a variety of those computing resources. Currently grid has been focused as one of the most important technologies in distributed computing environment, which was firstly introduced as a distributed architecture in science and engineering in mid 1990. Grid applies to many areas, from an advanced networking technology to artificial intelligence, which makes its concept not to be easily defined. However it can be regarded as a technology that enables resources of individuals or organizations to be safely shared, securely managed, and easily controlled. The most important function in large scale grid environment is resource discovery.

An efficient resource lookup service in small scale distributed network or centrally managed environment in a single organization which shares computing resources is not considered critical. However it becomes the bottom line of performance of grid service in large scale distributed network that allows arbitrary joins and leaves. Resource discovery has generally two steps. The first step is to locate a resource, and the second step is to connect to it. Locating resource would not be quite difficult in peer-to-peer network that is managed by a central server. Traditional lookup service, i.e. LDAP service, maintains all information, such as name, location, attribute in a single server that is normally based on central DBMS, which will result constant search performance. But it becomes variable when arbitrary joins and leaves are allowed [7]

Related Works(1)



APNOM 2000

(3)

1. Related Work

There are two kind of target domains that resource discovery is considered important. Firstly, a network environment that consists of a single network but variable number of nodes. Secondly, wide-area network environment. The former is recently focused on as a good environment for file sharing system as P2P (peer-to-peer) is getting popular.

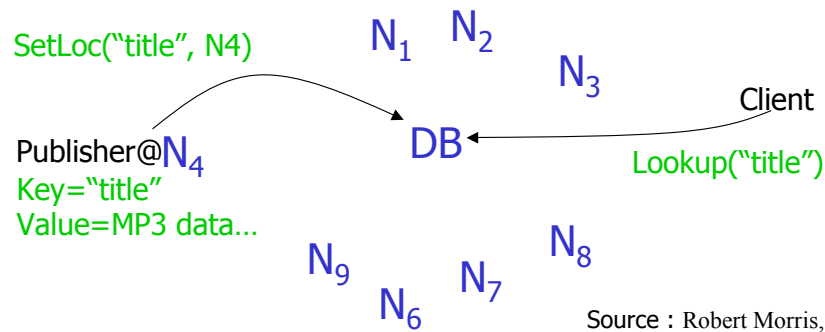
These systems recognize resources by their names. And there are a number of methods to find files. For example, aggressive flooding in Gnutella[5], request forward integrating automatic file replication in Freenet[10], an efficient name based search in CAN[13], Chord[4], Tapestry[12]. These systems have reliable and flexible architecture in order to resolve the location of data.

It is not practical to apply resource discovery based on names to computational grid because of dynamism, scale and heterogeneity. Among many discovery methods, condor and matchmaker are methods that do not use global names. Resource definition and request are processed in central server that is responsible for matching, which was first introduced in condor system, and turned out to be effective in LAN environment.

Related Works(2)



- Centralized lookup (Napster)



$O(N)$ state and a single point of failure
- should register/query to central server

APNOM 2000

(4)

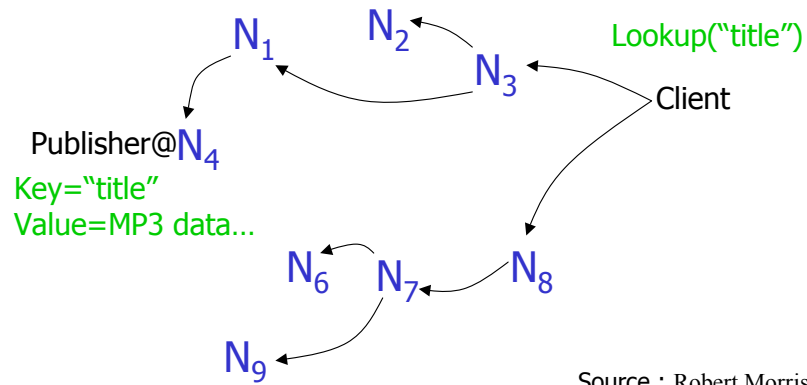
Globus has different architecture, MDS (Meta Data Service). It was originally centrally operated architecture. Later it was extended to a distributed system to cope with increasing resources and clients[7]. MDS-2 has multi information architecture to write grid data to an index server through registration protocol. The index server requests query to directory server through enquiry protocol and retrieves detailed information from information source.

In peer-to-peer file sharing systems, it is scalability that is hardest to achieve. Napster runs centrally operated directory servers which are vulnerable to single point of failure. Gnutella uses broadcast to decide search scope. Freenet manages collection of replicated documents but does not ensure searches or updates for sub level documents(?). Chord system is an appropriate discovery method for these situations.

Related Works(3)



- Flooded queries (Gnutella)



Source : Robert Morris, MIT

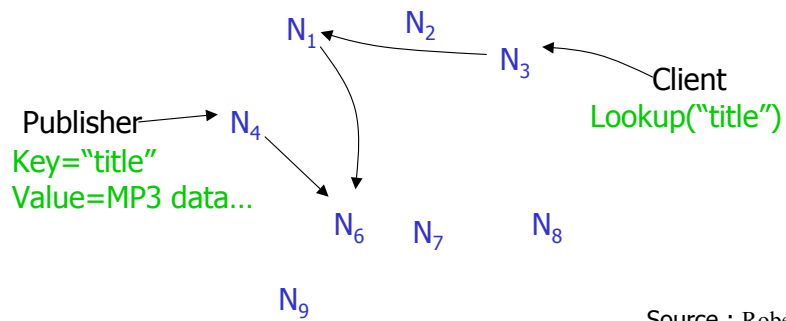


worst case $O(N)$ messages per lookup

Related Works(4)



- Routed queries(Freenet and Chord)



Source : Robert Morris, MIT

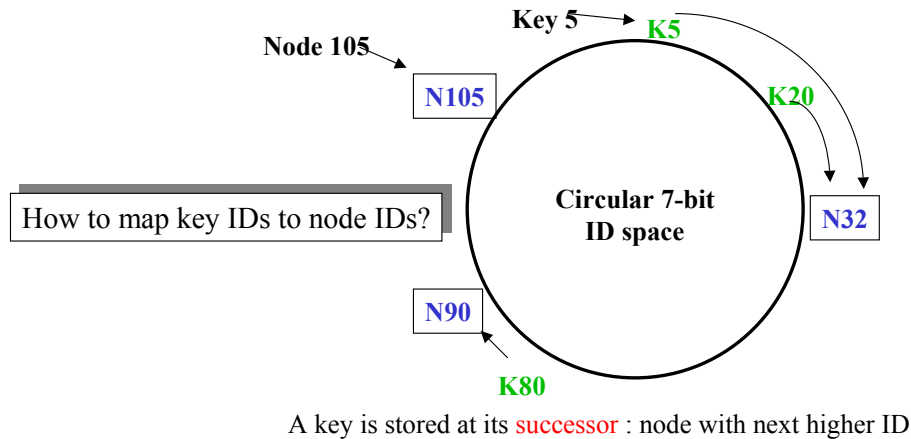
Chord : emphasizes efficiency and simplicity

- Efficient: $O(\log(N))$ messages per lookup
- Scalable: $O(\log(N))$ state per node

Related Works (Chord-1)



- Key identifier = SHA-1(key)
- Node identifier = SHA-1(IP address)
- Both are uniformly distributed and exist in the same ID space



APNOM 2000

(7)

3. Chord System Model

Chord system provides a distributed resource discovery protocol for peer-to-peer applications. It supports distributed lookup service, and insert, lookup, update by key. Key is represented by bytes array and uses random m -bit key identifier for uniqueness.

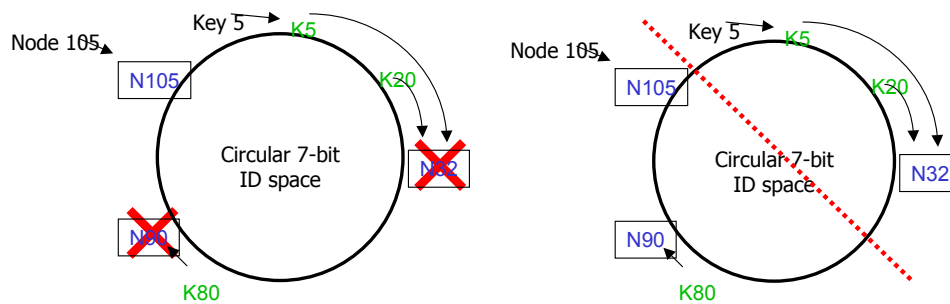
API of Chord system generally performs the followings. Key/value pair is inserted into given node when `insert(key, value)` is called. It looks for an appropriate key/value pair from neighbor nodes when `lookup(key)` is called. Chord system allows for key/value to be updated only by its originator. In Chord, delete operation does not exist, but this can be implemented easily by the update operation. The other two operations are used for joining to or leaving from Chord system.

Chord service model uses "best-effort" approach. If only one node that holds a certain key exists in Chord network, key/value pair is consistent. If Chord network is partitioned some other reason, each node communicate other nodes and is re-organized from partitioned status. So system is recovered and it only maintains partitioned binding information. After it is corrected partitioned network, this system maintains an exact location from stabilization protocol.

Related Works (Chord-2)



- (read only) data availability?
 - only one node fails we wait the recovery of that node
 - strictly sequential access → low performance
- network partition
 - partition occurs in the right before and just after the successor list



APNOM 2000

(8)

Therefore, it does not maintain tight bound. In order to maintain tight bound, it is required the consistency in key/value binding. That is, atomicity is not guaranteed when nodes are joined or left. If the network is partitioned before or after successor list when a node is in leave operation, atomicity cannot be satisfied. Therefore consistency requires to be supported during execution. In case that the network partitions and large number clients requests occurs, Chord system tends to work asynchronously which results inconsistent state of the network. Thus, it needs reliable resource discovery. Atomicity becomes more important when a system allows simultaneous write request from many clients, or a service is using agreement or totally ordered broadcast.

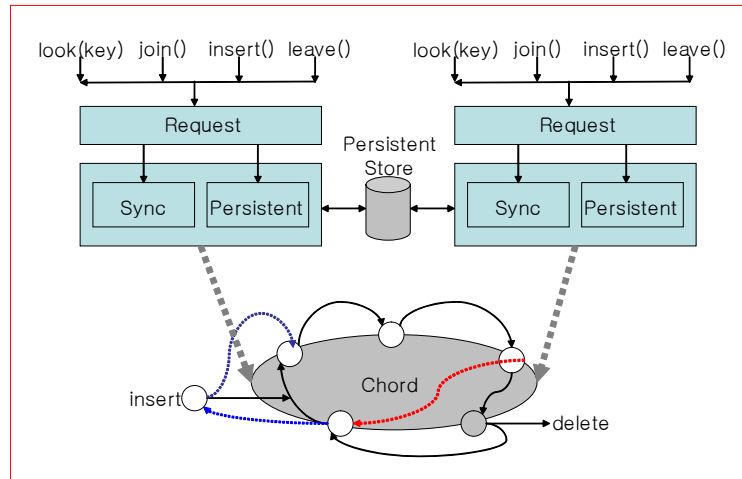
In Chord system, when a node fails, the node with a table containing n looks for successor. If its successor is found, it stores the copy of key/value into n . If network failure occurs while this operation, it needs to stop the current operation and then carries out re-stabilize process. While this process, no response from neighbor node means that the node has failed, which is then noticed to other neighbor node that it is dead. That is, inconsistency problem can occur in Chord system.

Related Works (Chord-3)



- multiple key values for GRID environment
 - available resources : CPU, disk space, printer, etc.
 - different attributes of available resources
 - amount(CPU allowed time, disk space,..)
 - start/end time of availability of resources
 -
- (read only)data availability?
 - only one node fails we wait the recovery of that node
 - strictly sequential access → low performance
- network partition
 - partition occurs in the right before and just after the successor list

Reliable Resource Discovery(1)



APNOM 2000

(10)

4. Reliable Resource Lookup

In Chord system, reliability issue means that only one copy can be accessed “serially”. It works efficiently in lookup service and shows relatively fast discovery time in joins and leaves. In addition, it carries out lazy update when join/leave/failure.

4.1. Designing an Improved System Model

Each node maintains its status. The status can hold any value of joining, active, and leaving. Each node has its own table relevant to the one that is inserted into incoming or outgoing queue of MQS. When a node sends request Insert and Delete operation, it is copied to the out coming queue. And it solved message order problem, which access constantly server information.

Reliable Resource Discovery(2)



1) Join

① Application request needs to be acknowledged and both request and data must be stored in queue atomically. ② join message tries to find target. New join or leave does not allowed until all nodes are searched. ③ mark the point on join location. ④ inform neighbor nodes of join request, and inform connection request, too. ⑤ update each finger table. Receive acknowledge from every adjacent nodes.

2) Leave

① When a node wants to leave, it changes, at first, its status into “transfer”, and then stops all pending requests. ② it transfers its data to successor atomically. Then, changes the status into “leave”. ③ after that, it sends leave message to request queue.

3) Insert

① Joined node tries to find a new successor by requesting target. ② It receives response from the successor. ③ Predecessor delivers target request to the nearest active successor. ④ the node becomes active.

4) Delete

① Joined node tries to find a new successor by requesting target. ② It receives response from the successor. ③ Predecessor delivers target request to the nearest active successor. ④ the node becomes active.

1) Join

① Application request needs to be acknowledged and both request and data must be stored in queue atomically. ② join message tries to find target. New join or leave does not allowed until all nodes are searched. ③ mark the point on join location. ④ inform neighbor nodes of join request, and inform connection request, too. ⑤ update each finger table. Receive acknowledge from every adjacent nodes.

2) Leave

① When a node wants to leave, it changes, at first, its status into “transfer”, and then stops all pending requests. ② it transfers its data to successor atomically. Then, changes the status into “leave”. ③ after that, it sends leave message to request queue.

3) Insert

① Joined node tries to find a new successor by requesting target. ② It receives response from the successor. ③ Predecessor delivers target request to the nearest active successor. ④ the node becomes active.

Conclusion



- Integrated system that can maintain the reliability of Chord System by adapting message queue
- Have to plan which evaluate the correctness of our system to see if it works properly in grid environment
- Perform a few experiments to see if the integrity is maintained under large number of simultaneous requests in large number of clients

5. Conclusion

So far, we have described the design of integrated system that can maintain the reliability of Chord system by adapting message queue. We are going to evaluate the correctness of our system to see if it works properly in grid environment. We are also planning to perform a few experiments to see if the integrity is maintained under large number of simultaneous requests in large number of clients

Reference List

[1] CLARKE, I., SANDBERG, O., WILEY, B., and HONG, T. Freenet: A distributed anonymous information storage and retrieval systems. In Workshop on Design Issues in Anonymity and Un-observability(2000)

[2] CZAJKOWSKI, K., FITZGERAND, S., FOSTER, I., and KESSELMAN, C. Grid information services for distributed resource sharing. In 10th IEEE Symposium on High Performance Distributed Computing (2001)

[3] IAMNITCHI, A., AND FOSTER, I. On fully decentralized resource discovery in grid environments. In International Workshop on Grid Computing (Denver, Colorado, November 2001), IEEE

[4] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. AND BALSKRISHNAN, H. Chord : A Scalable peer-to-peer lookup service for internet applications. In ACM SIGCOMM (2001)

[5] Gnutella protocol specification, <http://www.clip2.com/articles.html>

[6] Nancy Lynch, Dahlia Malkhi, David Ratajczak , Atomic Data Access in Distributed Hash Tables. Proceedings of the International Peer-to-Peer Symposium, March 2002.

[7] Adriana Iamnitchi, Ian Foster, Daniel C. Nurmi Peer-to-Peer Approach to Resource Discovery in Grid Environment. In High Performance Distributed Computing (Edinburgh, UK, July 2002)