Human's Cloud

A community cloud served by a P2P overlay network on top of the web platform

David Dias, david.dias@computer.org

Lisbon Tech, University of Lisbon

Abstract. Grid computing has been around from the 90ś No one true way of easy sharing resources Voluntary computing only used for Research, not accessible for application developers MOAR

Keywords: Cloud Computing, Peer-to-peer, Voluntary Computing, Cycle Sharing, Decentralized Distributed Systems, Web Platform

1 Introduction

1.1 Lorem ipsum

Excepteur sint

2 Objectives

2.1 Lorem ipsum

Excepteur sint

3 Related Work

The purpose of this section is to show the state of the art of the research topic, namely: Volunteer Computing, Cloud Computing, P2P Networks and the Web Platform

3.1 Cloud computing and Open Source Cloud Platforms

3.2 Volunteered resource sharing

3.2.1 Hybrid and Community Clouds

3.2.2 Cycle and Storage Sharing, using Volunteer Computing Systems

3.2.3 Peer-to-Peer Networks Architectures - Efficient resource discovery mechanism are fundamental for a distributed platform success, such as grid computing, cycle sharing or web application infrastructures[12], although the centralized model, keeping data bounded inside a data center offers the ability to have a stable and scalable way for resource discovery, this does not happen in a P2P network, where peers churn rate can vary greatly, there is no way to start new machines on demand for high periods of activity, the machines present are heterogeneous and so is their Internet connectivity, creating an unstable and unreliable environment. To overcome this challenges, several researches have been made in order to optimize how data is organized across all the nodes, improving the performance, stability and the availability of resources. The following paragraphs will describe the current state of the art P2P organizations, typically categorized in P2P literature as Unstructured or Structured[10], illustrated in Figure 1.

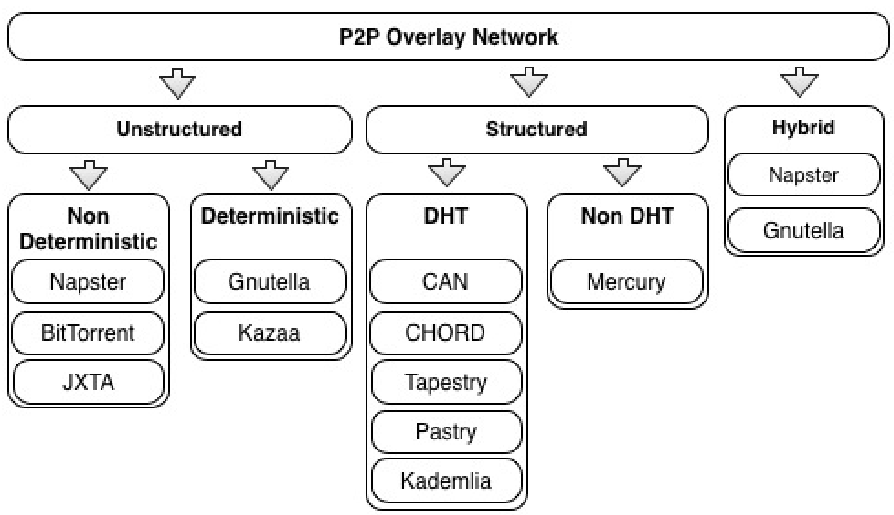

Fig. 1. Different types of P2P Overlay networks organizations

***Unstructured* -** We call 'Unstructured' to a P2P system that doesn't require or define any constraint for the placement of data, these include Napster, Kazaa and Gnutella, famous for it's file sharing capabilities, where nodes can share their local files directly, without storing the file in any specific Node. There is however a 'caveat' in the Unstructured networks, by not having an inherent way of indexing the data present in the network, performing a lookup results of the cost of asking several nodes the whereabouts of a specific file or chunk of the file, creating a huge performance impact with an increasing number of nodes.

In order to overcome this, Unstructured P2P networks offer several degrees of decentralization, one example is the evolution from Gnutella 0.4[5] to Gnutella 0.6 [16][13], which added the concept of super nodes, entities responsible for storing the lookup tables for the files in parts of the network they are responsible for, increasing the performance, but adding centralized, single points of failure. [12] classifies Unstructured networks into two types: deterministic and non-deterministic, defining that in a deterministic system, we can calculate before hand the number of hops needed to perform a lookup, knowing the predefined bounds, this includes systems such as Napster and BitTorrent[3], in which the file transfers are decentralized, the object lookup remains centralized, keeping the data for the lookup tables stored in one place, which can be gathered by one of two ways : (i) peers inform directly the index server the files they have; or (ii) the index server performs a crawling in the network, just like a common web search engine, this gives this network a complexity of O(1) to perform a search, however systems like Gnutella 0.6, which added the super node concept, remain non deterministic because it's required to execute a query flood across all the super nodes to perform the search.

Structured with Distributed Hash Tables - Structured P2P networks have an implicit way of allocating nodes for files and replicas storage, without the need of having any specie of centralized system for indexing, this is done by taking the properties of a cryptographic hash function [1][7][11], such as SHA-1[4], which applies a transformation to any set of data with a uniform distribution of possibilities, creating an index with O(log(n)) peers, where the hash of the file represents the key and gives a reference to the position of the file in the network. DHT's such as Chord[15], Pastry[14] and Tapestry[18], use a similar strategy, mapping the nodes present in the network inside an hash ring, where each node becomes responsible for a segment of the hash ring, leveraging the responsibility to forward messages across the ring to his 'fingers'(nodes that it knows the whereabouts). Kademlia[9] organizes it's nodes in a balanced binary tree, using XOR as a metric to perform the searches, while CAN[6] introduced and a several dimension indexing system, in which a new node joining the network, will split the space with another node that has the most to leverage. Evaluating the DHT Structured P2P networks raises identifiable issues/challenges, that result as the trade-off of not having an centralized infrastructure, responsible for railing new nodes or storing the meta-data, these are: (i) generation of unique node-ids is not easy achievable, we need always to verify that the node-id generated doesn't exist, in order to avoid collisions; (ii) the routing table is partitioned across the nodes, increasing the lookup time as it scales. Table 1, showcases a comparison of the studied DHT algorithms.

Structured without Non-Distributed Hash Tables - Mercury[2], a structured P2P network that uses a non DHT model, was design to enable range queries over several attributes that data can be dimensioned on, which is desired on searches over keywords in several documents of text. Mercury design

P2P system	Overlay Structure	Lookup Protocol	Networking parameter	Routing table size	Ruting complexity	Join/leave overhead
Chord	1 dimension, Hash ring	Matching key and NodeID	n= number of nodes in the network	O(log(n))	O(log(n))	O(log(n)^2)
Pastry	Plaxton style mesh structure	Matching key and prefix in NodeID	nn̄umber of nodes in the network, bb̄ase of identifier	O(log$_b$ (n))	O(b log $_b$ (n)+b)	O(log(n))
CAN	d-dimensional ID Space	Key value pair map to a point P in the D-dimensional space	n= number of nodes in the network, d=number of dimensions	O(2d)	O(d n$^{1/2}$)	O(2d)
Tapestry	Plaxton style mesh structure	Matching suffix in NodeID	n=number of nodes in the network, b=base of the identifier	O(log$_b$(n))	O(b log $_b$ (n)+b)	O(log(n))
Kademlia	2	3	4	5	6	7

Table 1. Summary of complexity of structured P2P systems

offers an explicit load balancing without the use of cryptographic hash functions, organizing the data in a circular way, named 'attribute hubs'.

3.2.4 Fault Tolerance, Assurance and Trust Volunteer resource sharing means that we no longer have our computational infrastructure in a confined and very well monitored place, introducing new challenges that we have to address [8] to maintain the system running with the minimum service quality, this issues can be: scalability, fault tolerance and security[17] of the data and that the system doesn't get compromised. This part of the document serves to describe the techniques implemented in previous non centralized systems to address this issues.

Fault Tolerance

Assurance and Trust

3.3 Resource sharing using the Web as platform

3.3.X What has been happening

3.3.X Previous attempts

4 Architecture

4.1 Node Level

4.2 Client API

4.3 Storage

4.4 Reputation Mechanism

4.5 Job Scheduling

5 Evaluation

5.1 Lorem ipsum

Excepteur sint

6 Conclusions

6.1 Lorem ipsum

Excepteur sint

References

1. S Bakhtiari and J Pieprzyk. Cryptographic Hash Functions : A Survey 1 Introduction. pages 1–26.
2. Ashwin R Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury : Supporting Scalable Multi-Attribute Range Queries. pages 353–366.
3. Bram Cohen. The BitTorrent Protocol Specification, 2009.
4. Cisco D. Eastlake, 3rd Motorola; P. Jones Systems. RFC 3174 US Secure Hash Algorithm 1 (SHA1), 2001.
5. Protocol Definition. The Gnutella Protocol Specification v0 . 4. *Solutions*, pages 1–8, 2003.
6. Mark Handley and Richard Karp. A Scalable Content-Addressable Network.
7. David Kargerl, Tom Leightonl, and Daniel Lewinl. Consistent Hashing and Random Trees : Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web *. pages 654–663.
8. Georgia Koloniari and Evaggelia Pitoura. Peer-to-Peer Management of XML Data : Issues and Research Challenges. 34(2):6–17, 2005.
9. Petar Maymounkov and David Mazières. Kademlia: A Peer-to-peer Information System Based on the XOR Metric.
10. Dejan S Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, Zhichen Xu, and J I M Pruyne. Peer-to-Peer Computing. Technical report, 2003.
11. Bart Preneel. The State of Cryptographic Hash Functions. pages 158–182, 1999.
12. Rajiv Ranjan, Aaron Harwood, and Rajkumar Buyya. A study on peer-to-peer based discovery of grid resource information. *..., Australia, Technical Report GRIDS ...*, pages 1–36, 2006.

13. M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. *Proceedings First International Conference on Peer-to-Peer Computing*, pages 99–100, 2002.
14. Antony Rowstron and Peter Druschel. Pastry : Scalable , Decentralized Object Location , and Routing for Large-Scale Peer-to-Peer Systems. pages 329–350, 2001.
15. Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan Ý. Chord : A Scalable Peer-to-peer Lookup Service for Internet. pages 149–160, 2001.
16. R. Manfredi T. Klingberg. RFC - Gnutella 0.6 Protocol Specification, 2002.
17. Dan S Wallach. A Survey of Peer-to-Peer Security Issues.
18. Ben Y Zhao, John Kubiatowicz, and Anthony D Joseph. Tapestry : An Infrastructure for Fault-tolerant Wide-area Location and Routing Tapestry : An Infrastructure for Fault-tolerant Wide-area Location and Routing. (April), 2001.