PATRY based P2P application: PAST

Andrea Marin

Università Ca' Foscari di Venezia
Dipartimento di Informatica
Corso di Sistemi Distribuiti

2009

Presentation outline

Presentation based on the original paper: P. Druschel and A. Rowstron. *PAST: A large-scale, persistent peer-to-peer storage utility.*

What is PAST?

Definition (PAST)

PAST is an Internet-based, peer to peer global storage utility. It provides strong persistence, high availability, scalability and security.

- Composed of several nodes
- Each node routes requests, may add or retrieve a file
- Each node may optionally contribute storage to the system
- Inserted files are replicated
- Cache mechanisms are provided to enhance performances

Advantages of PAST storage system

- Nodes are different in terms of location, ownership, administration, jurisdiction, ... $\Longrightarrow$ strong persistence and high availability
- No need of transportation of data
- No need of backups
- No need of application level mirroring of resources to meet performance constraints
- Storage and bandwidth can be shared among a group of nodes
  - A group of nodes can store a resource which exceeds the capacity of each individual node

General characteristics

- Stored objects are immutable
- Each file has its own *quasi-unique* FileId
- A file can be shred by the owner by distributing somehow its FileId
- Delete operation is not supported
- File owner may *reclaim* the storage associated with a file
    - This does not guarantee that the file is no longer available
- The routing scheme is Pastry

Assumption on the node

- Each node has a public/private key pair
- A node is identified by a 128 bit NodeId
- NodeId is a hash of the Node's public key

File certificates

- Each file stored in a PAST network has a 160 bit FileId (only the first 128 are used for the routing)
- FileId is computed as a hash of
  - the file textual name
  - owner's public key
  - random salt
- Each file has a certificate:
  - It is generated before the file is inserted
  - It contains:
    1. FileId
    2. replication factor
    3. the salt
    4. insertion date
    5. hash of file content

What do we obtain?

- A file is available if one of the k nodes with a replica is active
- Resources are stored in nodes with different characteristics with high probability
- Load balance is preserved

Security

Assumptions for the security analysis:

- Breaking the public-key cryptographic system or the hash function is infeasible
- We assume that the number of malicious nodes is small compared to the total number of nodes
- If smartcards are used, then the attacker cannot control this system

In the following we assume that smart card are used.

- Each node has a smartcard with a public/private key
- Each user has a smartcard with a public/private key

Public keys are signed with the private key of the issuer of the smartcard

NodeId generation

- NodeId is a hash of public key of the node smartcard
- All the 128 bit sequences have the same probability of being chosen as NodeId
- Numerically close NodeIds are not necessary geographically close nodes
- A NodeId may be checked by other nodes

File certificate generation

- The user smartcard is used to generate a file certificate which contains:
  - Hash of the file content (computed by the node)
  - FileId (computed by the smartcard)
  - Replication factor k
  - The salt
  - Signature of the smartcard
- The storing node verifies:
  - is the user authorized to store the file?
  - has the file being stored corrupted?
  - is the FileID authentic? (DoS attak?)
- A certificate is delivered when a file is required

Store receipts generation

- When a node stores a copy of a file it produces a *store receipt*
- The store receipt is sent back to the client
- The client verifies that k copies of the files are stored

Reclaim certificate and receipt

- The reclaim request is associated with a reclaim certificate
- The reclaim certificate contains the FileId and is signed by user's smartcard
- The storage node receiving a reclaim request...
  - checks if the signature is correct (only the owner can reclaim)
  - if ok, the smartcart of the storage node generates a receipt wich contains the reclaim certificate and the amount of storage reclaimed

Smartcard and quotas

- The user's smartcard maintains storage quotas
- When a file certificare is issued an amount corresponding to the file size multiplied by the replication factor is debited against the quota
- When a node presents a reclaim receipt received by a storage node then the user's quota is updated
- Node's smartcard maintain the storage quota offered by the node

Conclusions

- Simulations have proved that PAST can use around 95% of storage
- Usage of smartcards allows for high security and payments
- PAST can work even without a system of quota and smartcards
- Reliability is given by PASTRY routing