Enabling Data Integrity Protection in Regenerating-Coding-Based Cloud Storage: Theory and Implementation

ABSTRACT:

To protect outsourced data in cloud storage against corruptions, adding fault tolerance to cloud storage, along with efficient data integrity checking and recovery procedures, becomes critical. Regenerating codes provide fault tolerance by striping data across multiple servers, while using less repair traffic than traditional erasure codes during failure recovery. Therefore, we study the problem of remotely checking the integrity of regenerating-coded data against corruptions under a real-life cloud storage setting. We design and implement a practical data integrity protection (DIP) scheme for a specific regenerating code, while preserving its intrinsic properties of fault tolerance and repair-traffic saving. Our DIP scheme is designed under a mobile Byzantine adversarial model, and enables a client to feasibly verify the integrity of random subsets of outsourced data against general or malicious corruptions. It works under the simple assumption of thin-cloud storage and allows different parameters to be fine-tuned for a performance-security trade-off. We implement and evaluate the overhead of our DIP scheme in a real cloud storage testbed under different parameter choices. We further analyze the security strengths of our DIP scheme via mathematical models. We demonstrate that remote integrity checking can be feasibly integrated into regenerating codes in practical deployment.

EXISTING SYSTEM:

- The problem of checking the integrity of static data, which is typical in long-term archival storage system is considered. This problem is first considered under a single-server scenario by Juels and Kaliski and Ateniese et al, giving rise to the similar notions POR and PDP, respectively.
- The existing system design extends the single-server compact POR scheme for regenerating-coded storage.
HAIL system, which stores data via erasure coding. HAIL operates on a per-file basis and it is nontrivial to directly apply HAIL to regenerating codes.

PROBLEM DEFINITION:

- A major limitation in the existing system is that they are designed for a single-server setting.
- Efficient data integrity checking has been proposed for different redundancy schemes, such as replication erasure coding and regenerating coding.
- A large storage overhead, as the amount of data stored increases with a more flexible checking granularity
- The storage servers have encoding capabilities for generating encoded data.

PROPOSED SYSTEM:

- The aim of our proposed system is to protect outsourced data in cloud storage against corruptions, adding fault tolerance to cloud storage, along with efficient data integrity checking and recovery procedures, becomes critical.
- Regenerating codes provide fault tolerance by striping data across multiple servers, while using less repair traffic than traditional erasure codes during failure recovery. We design and implement a practical data integrity protection (DIP) scheme for a specific regenerating code, while preserving its intrinsic properties of fault tolerance and repair-traffic saving.
- Our DIP scheme is designed under a mobile Byzantine adversarial model, and enables a client to feasibly verify the integrity of random subsets of outsourced data against general or malicious corruptions. It works under the simple assumption of thin-cloud storage and allows different parameters to be fine-tuned for a performance-security trade-off.
- We further analyze the security strengths of our DIP scheme via mathematical models. We demonstrate that remote integrity checking can be feasibly integrated into regenerating codes in practical deployment.
ADVANTAGES OF PROPOSED SYSTEM:

- By striping redundant data across multiple servers, the original files can still be recovered from a subset of servers even if some servers are down or compromised.
- A thin-cloud setting is used where servers only need to support standard read/write functionalities for portability and simplicity.
- Different parameters can be adjusted for the performance-security trade-off.

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 44 Mb.
- Monitor : 15 VGA Colour.
- Ram : 512 Mb.

SOFTWARE REQUIREMENTS:

- Operating system : Windows XP/7.
- Coding Language : net, C#.net
- Tool : Visual Studio 2010
- Database : SQL SERVER 2008

REFERENCE: