How to protect data integrity on a file system

Fujita Tomonori

NTT Network Innovation Laboratories

tomof@acm.org

IWIA 2003, Darmstadt, Germany

Increased concern of security on a file system

- IP Storage protocols such as iSCSI, are on the horizon
  - Outsourcing storage saves the cost of managing growing storage volume
  - Storage is connected with networking fabrics
  - Packets are transmitted over the Internet
- New concerns
  - Can we really trust outsourcing storage vender ?
    - *Many crimes are committed by internal people*
    - *The loss by internal crime is bigger than external*
  - Networking is open
    - *Intruders get more chances to access your data*

Problems

- Prior secure file systems haven't address the followings
  - Not designed for block-level storage
    - *Most use file-level protocols such as NFS*
    - *Consistency after a system crash is more important*
  - Protection against replay attacks
    - *Protecting individual blocks, or files is not insufficient*
    - *Attackers bogus you with the combination of old block and latest block*

Proposal

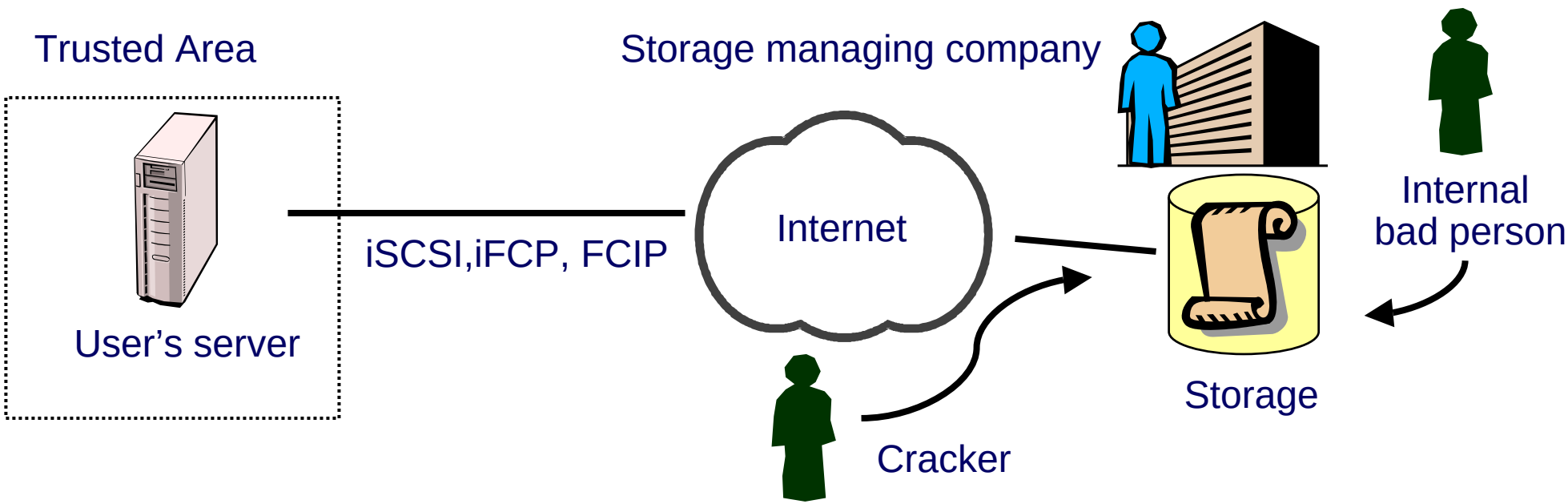
- Arbre: a file system providing strong data integrity
  - designed to run on untrusted remote block-level storage
  - Integrate hash tree to the file system structure
- All operations are atomic
  - You never see any inconsistency
- The entire file system integrity at some point
  - Not each block, or each file individually

Outline

- Thread model
- Three issues about data integrity on a file system
  - Meta data is more important than file data
  - The integrity of each block or each file is not enough
  - Consistency techniques are essential
- Solution
- Design
- Performance

Threat Model

- Assumptions
  - User's server and OS are trusted
  - Storage is untrusted
  - Connected with a block level protocol such as iSCSI, iFCP, FCIP
- Goals
  - Compromised data are detectable



The integrity of meta data

- Two type of data on a file system
  - File data : actual user data, i.e. user read or write it from/to files
  - Meta data : the structure of a file system, i.e. the file system uses it.
    - *Block bitmap : manage free blocks*
    - *Inode bitmap : manage free inodes*
- The integrity of meta data is more important
  - Corrupt meta data crash OS, and all applications.
  - OS may unintentionally destroy file data due to corrupt meta data
    - *Corrupt block bitmap lies that a block in use is free*
    - *Corrupt inode bitmap lies that an inode in use is free*
  - NFS-like file systems don't bother with this problem
    - *OS runs on an outsourcing company server*
    - *OS running on customer's host has no concept of low level information like meta data, inode bitmap or super block*

Integrity of the entire file system

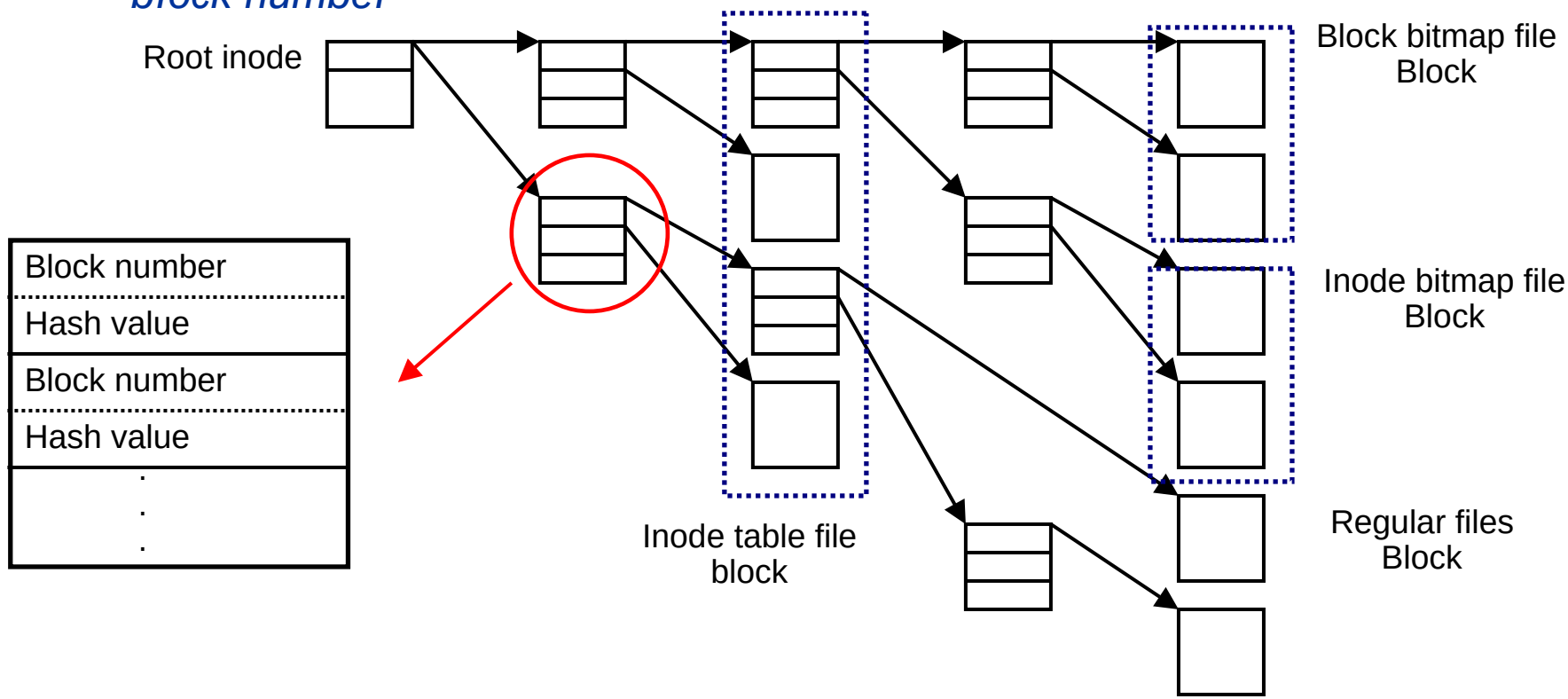
- The integrity of individual blocks or files are not enough
  - With these approaches, you cannot detect replay attacks
    - *Attackers replace a latest set of a block and its valid hash with old one*
    - *The file system can detect it because it is old, but valid*
    - *The file system consists of some latest blocks and old blocks !*
- The integrity of the entire file system is better
  - Save the all hashes of blocks constituting the file system at some point
  - A attacker has to replace all sets of blocks and hashes

Consistency after a system crash

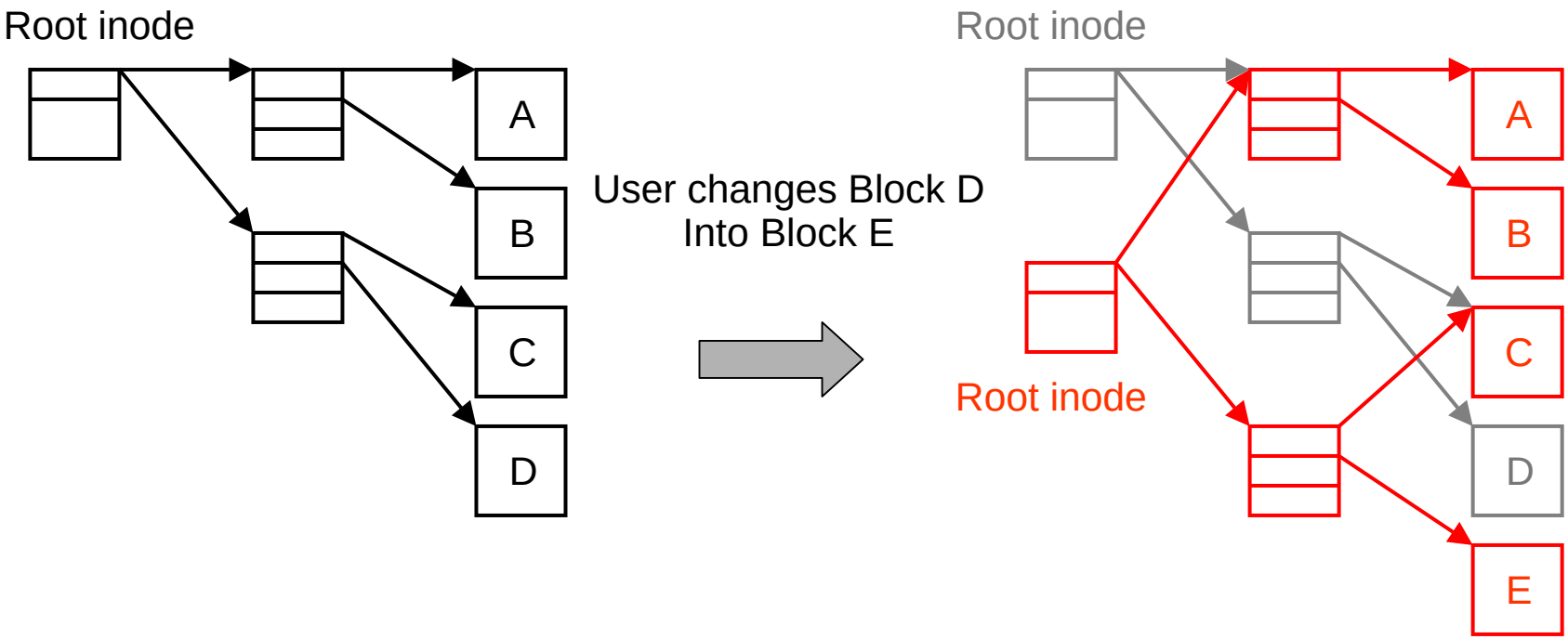
- Ensure consistency after an unexpected event
  - Data consistency
    - *Metadata*
    - *File data*
  - Hash consistency
    - *The hash value of data*
- Only Metadata consistency is not enough
  - With the metadata journaling approach, you may see inconsistency
    - *A operation modifies a file and its inode, only one of them may be written to storage*
  - You cannot know what happened
    - *Whether the system crash caused the inconsistency*
    - *Or whether malicious replay attacks caused it*

Arbre design

- Integrate a hash tree to the file system structure
  - File system is structured as a single tree
  - The root block encapsulate the entire file system
  - Store disk block number and its hash together
    - *Read a block and verify it by using the hash stored alongside the block number*



Consistency : Phase Algorithm



Red blocks represent the latest file system

- Write leaf blocks first, the root block last
  - All modifications are atomic
  - No-overwrite algorithm, and reclaim unused blocks later

Phase Algorithm characteristics

- Advantage
  - Write is fast
    - *You can delay block allocations*
    - *You can write physically continuous blocks like a log structured file system*
- Disadvantage
  - Layout optimizations are difficult
    - *Fragment arises easily*
  - Read is slow
    - *Reading requires many disk seeks like a log structured file system*

Phase algorithm's bottlenecks

- Two clear bottlenecks
  - Write more data
    - *Each modification results in modification all the way up the root of a tree*
  - Tree transition requires many synchronous writes
- Two solutions
  - You can cluster some modifications and push them into a single branching tree
  - You copy data and continue to modify it instead of waiting the completion of block writes

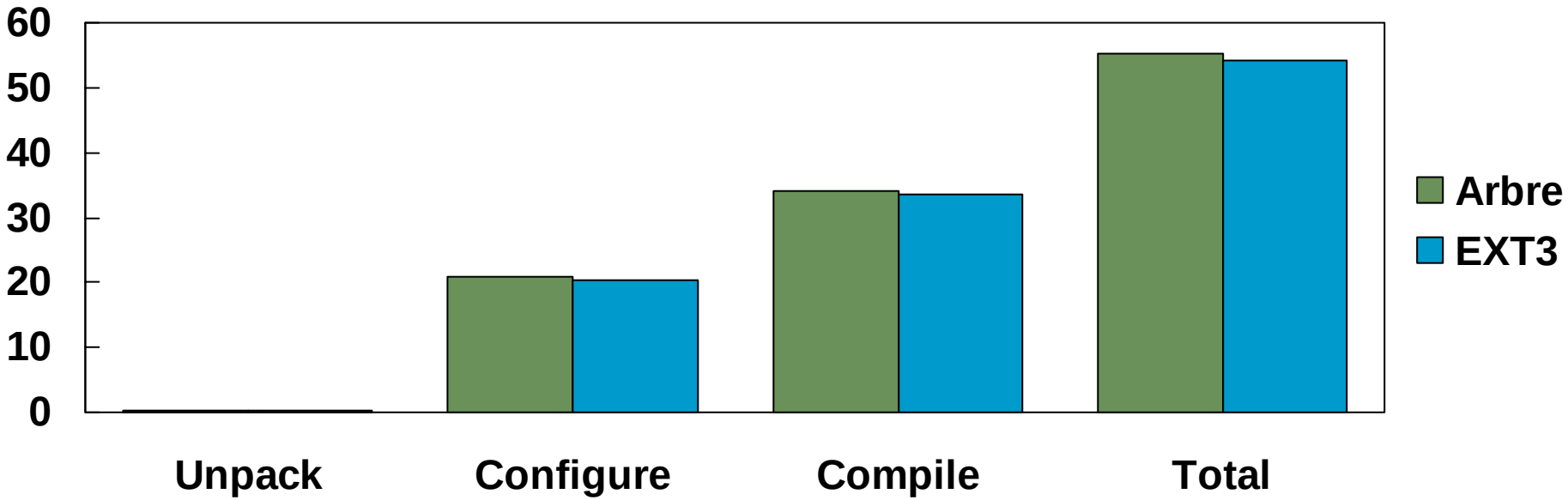
Implimentation

- Asynchronous writes
  - Modified metadata is not written when a system call returns
    - *This semantics is identical to file systems logging data asynchronous*
- Asynchronous hashing
  - Arbre avoids needless hash re-computations

Experiment environment

- Platform
  - Now, Arbre runs on Linux Kernel 2.5.x
  - The former implementation on 2.4.x has some performance problems
- Host
  - 2GHz Xeon running Linux 2.5.63 with 1GB of main memory
- Storage
  - Local disk : 18GB 15000RPM SCSI disk
- File systems compared against:
  - EXT3 writeback mode : logs metadata operations asynchronous
    - *It provides the consistency of metadata*
    - *Arbre provides the stronger consistency : metadata and file data.*

Performance : SSH benchmark



- SSH benchmark
  - Unpack the source code, configure and compile it
- The small performance loss
  - less than 2% compared with EXT3(writeback mode)

Summary

- Arbre for untrusted remote block-level storage
  - Combination of a hash tree and a tree-structured file system
    - *Provides the consistency of file data and metadata*
    - *Provides the integrity of the entire file system*
    - *Provides acceptable performance*