Optimizing the Storage of Massive Electronic Pedigrees in HDFS

Authors: Yin Zhang, Weili Han, Wei Wang, Chang Lei
Presented by: Yin Zhang
Oct 25th 2012
Electronic pedigree system

>> trustworthily tracking of the processes

>> Small-sized but huge volume of electronic pedigrees

Optimizing the storing and accessing of massive small XML files in HDFS

Abstract
Abstract

- reduce the metadata occupation at NameNode
- improve the efficiency of accessing small XML files

- Feasibility,
- Effectiveness,
- Efficiency.
Reduce memory consumption of NameNodes by 50%
Improve performance of storing by 91%
Accelerate accessing by 88% in Hadoop
What is a pedigree?

Electronic pedigree

Background: electronic pedigree
 ✓ Of a nested architecture
 ✓ Tens of KB to hundreds of KB
 ✓ Different types
 ✓ Attribute ‘Lot number’
 ✓ Attribute ‘Item serial number’
 ✓ Attribute ‘pedigreeID’

Background: electronic pedigree
Characteristics of electronic pedigree

✓ serialized attributes
✓ One single-writer, multiple-reader Model
✓ Correlated electronic pedigrees
✓ freshness date of goods
Electronic Pedigree Storage Server

Manage massive electronic pedigrees
Access electronic pedigrees
Receive electronic pedigrees in two ways

Background: EPSS
Hadoop Distributed File System

- open-source software framework
- One single-writer, multiple-reader model

Performance bottlenecks for massive small files
① High cost for metadata management
② High memory cost for files
③ High time cost for contacting

Optimizing the Storage of Massive Electronic Pedigrees in HDFS

Disadvantages of HDFS in EPSS
Merge correlated electronic pedigrees into a bigger file
>> decrease file size in HDFS
>> relieve NameNode’s memory

Four strategies to merge

Internal index file

File merging
A global mapping table >> records between each small XML file and its merged file

<table>
<thead>
<tr>
<th>E-pedigree file name(var)</th>
<th>Start(Long)</th>
<th>End(Long)</th>
<th>Merged file name(var)</th>
</tr>
</thead>
</table>

An indexing table >> records between attributes and small XML files

e.g. “pedigreeID=001” → “pedigree001” → “bigfile1”

File mapping
File reading
File writing
Prefetching

- Reduce response time for user
- Prefetch XML files in the same merged file
- Consistent between cache and HDFS
- Influenced by merging strategy
Frequency of accessing electronic pedigrees will turn down with time passing by, especially after the freshness date of goods ends.
Experiment
Evaluation
Evaluation
Evaluation

**Reading Time (second)**

- **Origin HDFS**
  - 200 pedigrees from 27 Groups: 3168 seconds
  - 200 pedigrees from 10 Groups: 2653 seconds
  - 200 pedigrees from 2 Groups: 1830 seconds

- **the proposed approach**
  - 200 pedigrees from 27 Groups: 378 seconds
  - 200 pedigrees from 10 Groups: 206 seconds
  - 200 pedigrees from 2 Groups: 18 seconds
① Helpful for the files of XML type
② Dynamic grouping
③ File correlation and attribute serialization
④ Global indexing table and global mapping table
⑤ Prefetching technology
⑥ File remerging technology

Difference from related work
Conclusion and future work

✓ Optimizing the Storage of Massive Electronic Pedigrees in HDFS

✓ Avoiding privacy problem of storing electronic pedigrees
Q&A