LH^*_{RS}: A Highly Available Scalable Distributed Data Storage System

Witold Litwin **Rim Moussa**
{Witold.Litwin, Rim.Moussa}@dauphine.fr
Université Paris Dauphine, France

Thomas J.E. Schwarz
TjSchwarz@scu.edu
Santa Clara University, USA

Basic Capabilities

- **Stores application data on any number of storage servers at a local net**
  - **Scales up dynamically & transparently to the application**
  - **Uses *scalable distributed linear hash* partitioning (LH^*_{LH} scheme)**
  - **Appends new servers by splits of existing ones 0,1,2…**
    - **in the *linear hash* order 0, 0,1, 0,1,2,3, 0…2^i – 1, 0…**
- **Provides *k* – availability**
  - **All data remain available for the application despite unavailability (failure) of any *k* servers**
  - ***k* = 0,1,2,3... on demand or *k* may scale with the file to preserve the reliability level**
- **Data in distributed RAM**
  - **Data access & recovery speed orders of magnitude faster than to disk storage**
- **Close to minimal storage overhead for any *k***
- **Intended for large scalable files, DBMSs, P2Ps, Grids...**

Performance

(Wintel P4 1.8GHz, 1Gbs Ethernet)

***Data bucket load factor* : 70 %**

Parity overhead* : *k / m

***m* is file parameter, *m* = 4,8,16…**

larger *m* increases the recovery cost

Key search time

- **Individual : 0.2419 ms**
- **Bulk : 0.0563 ms**

File creation rate

- **0.33 MB/sec for *k* = 0,**
- **0.25 MB/sec for *k* = 1,**
- **0.23 MB/sec for *k* = 2**

***Record insert time* (100 B)**

- **Individual : 0.29 ms for *k* = 0,**
  0.33 ms for *k* = 1,
  0.36 ms for *k* = 2
- **Bulk : 0.04 ms**

Record recovery time

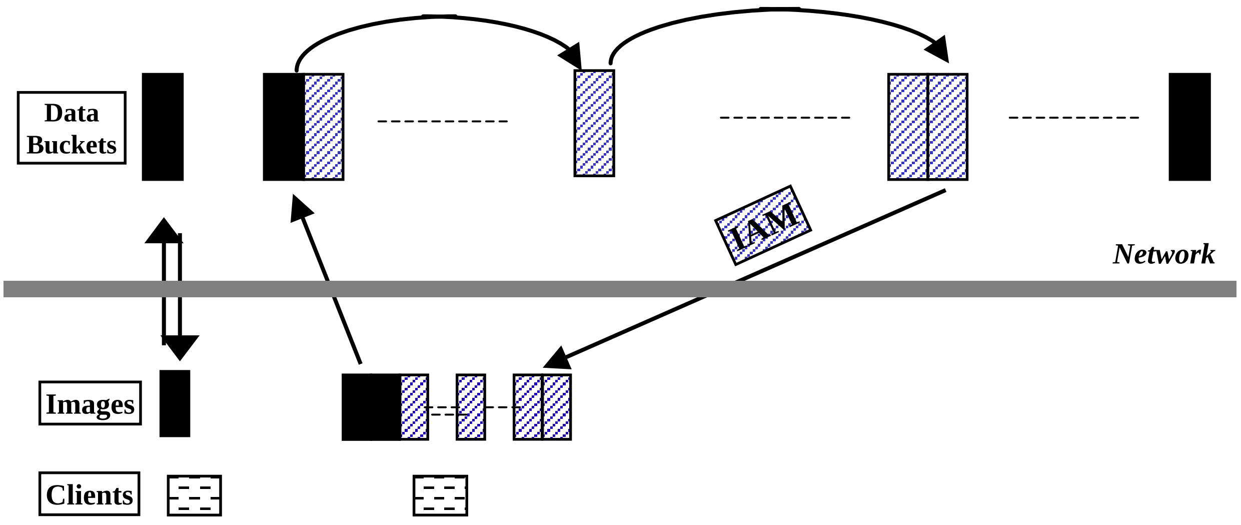
- **About 1.3 ms**

***Bucket recovery rate* (*m* = 4)**

- **5.89 MB/sec from 1-unavailability,**
- **7.43 MB/sec from 2-unavailability,**
- **8.21 MB/sec from 3-unavailability**

LH^*_{RS}: A Highly Available Scalable Distributed Data Storage System

Data Addressing & File Expansion



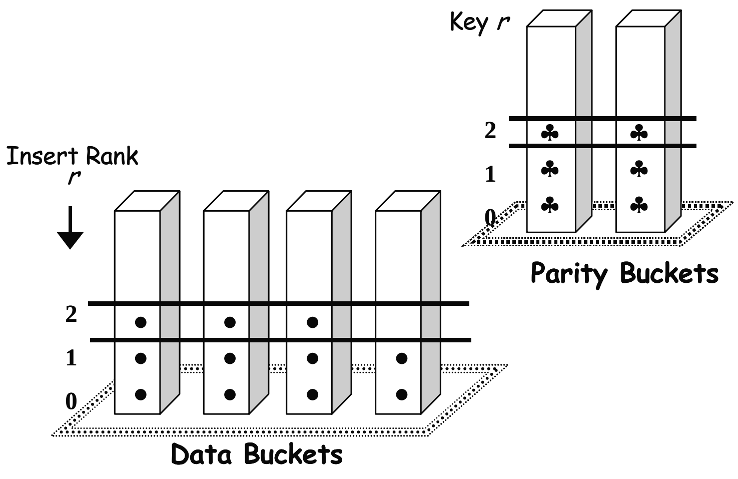
Data Record = (Key, Non-Key Data)

- ***Linear hashing*** **(LH) of the key defines the *correct* data bucket and server**
  - **Where the record should be**
  - **According to the *file state* meta-data**
    - **Dynamically evolving at the *coordinator* server with the file size**
- **An application interfaces the LH^*_{RS} *client* software**
  - **Located at its computer**
- **Each client maintains its *image* of the file state**
  - **Possibly different from the actual state**
    - **The coordinator does not post file state updates to clients**
- **To address a record, the client applies LH to its image**
  - **The query may arrive to an *incorrect* bucket**
  - **An incorrect bucket forwards the query to another bucket**
- **Any forwarding reaches the correct bucket in at most <u>two</u> hops**
  - **<u>Property of LH* unique at present</u>**
  - **Usually there is <u>no</u> <u>hops</u>**
- **A server getting a forwarded message sends back to the client the *Image Adjustment Message* (IAM)**
- **The client updates its image**
- **The same addressing error cannot repeat.**

LH*$_{RS}$: A Highly Available Scalable Distributed Data Storage System

High Availability

Group Structure



● : Key Data Field

♣ : Rank [Key List] Parity Field

Parity Matrix

$$\begin{array}{cccc}
0001 & 0001 & 0001 & \dots \\
0001 & eb9b & 2284 & \dots \\
0001 & 2284 & 9é74 & \dots \\
0001 & 9e44 & d7f1 & \dots \\
\dots & \dots & \dots & \dots
\end{array}$$

Logarithmic Parity Matrix

$$\begin{array}{cccc}
0000 & 0000 & 0000 & \dots \\
0000 & 5ab5 & e267 & \dots \\
0000 & e267 & 0dce & \dots \\
0000 & 784d & 2b66 & \dots \\
\dots & \dots & \dots & \dots
\end{array}$$

- ***Bucket*** **groups of *m* data buckets each : 0,1…*m* – 1 ; *m*…2*m* – 1 ; 2*m*…**
- ***Record*** **groups of up to *m* data record each**
  - **Records with the same rank *r* = 1,2… in a bucket group**
- ***k*** **parity buckets (records) per group**
- **Novel & fastest known generalized Reed Salomon code for parity encoding/decoding**
  - **Galois Field GF (2^{16})**
  - **Parity matrix with 1st column of 1’s and first line of 1’s**
    - **XOR only calculus for *k* = 1**
      - **As in popular RAID systems (limited to *k* = 1 usually, *k* = 2 at most)**
    - **XOR only calculus for 1st bucket (record) of the group for every *k***
  - **Use of the *logarithmic* parity matrices for encoding and decoding**

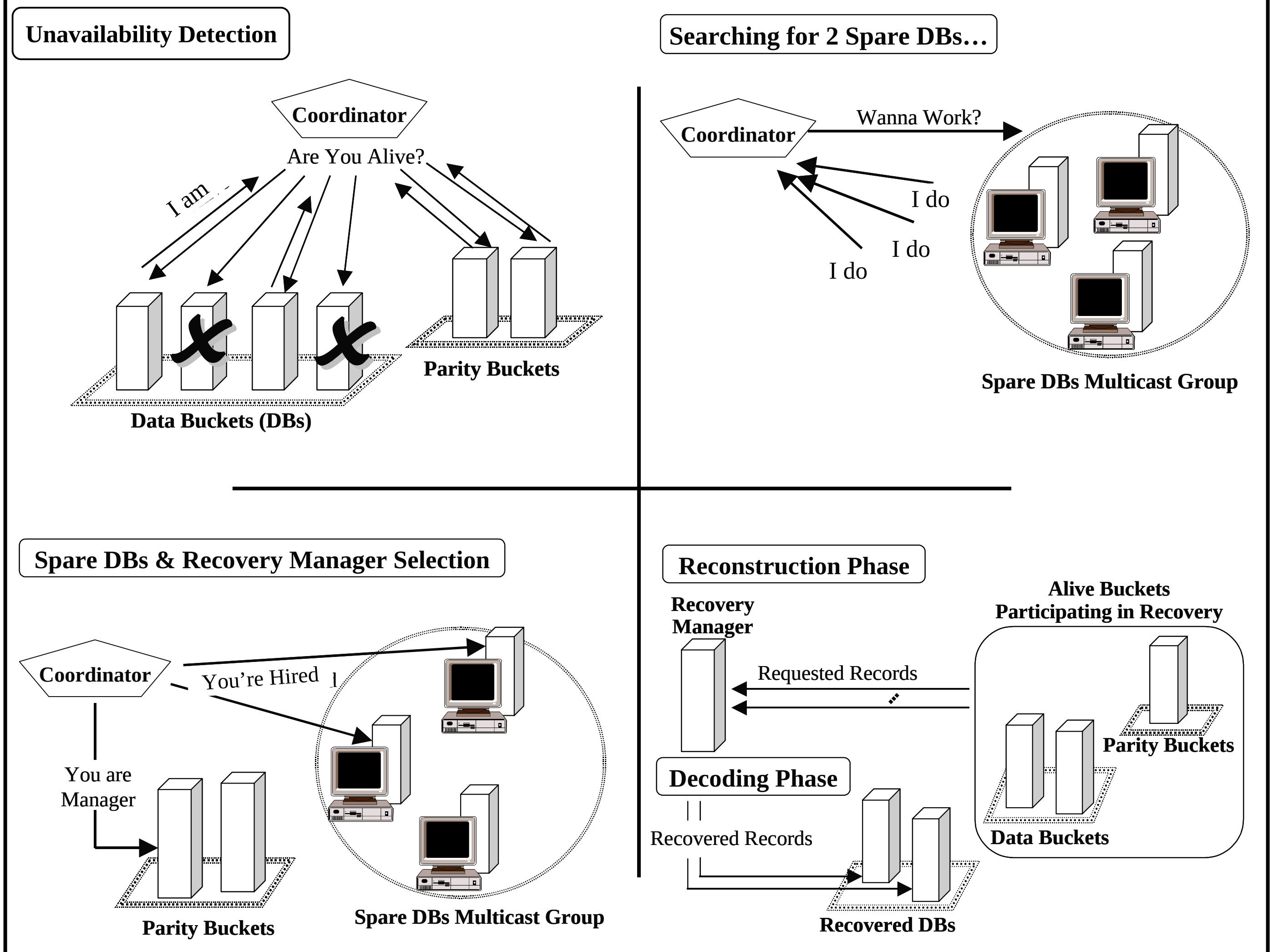
System Architecture

- **Multithreading**
- **TCP/IP in Passive Mode for Large Transfers**
- **UDP for Individual Queries and Control Messages**
  - **with Flow Control**
- **Multicast for Probing New Servers (Spares)**

VLDB 2004 Toronto

LH*$_{RS}$: A Highly Available Scalable Distributed Data Storage System

Bucket Recovery Scenario



Demonstration Outline

- File Creation
  - Client and Server Setup
  - Record Insert
  - File Expansion (bucket splits)
- Key Search
- Record Update
- High Availability Level Increase
- *k* Data Bucket Recovery; *k* = 1, 2, 3
- Record Recovery

Partial Support

MS Research, CEE-ICONS, SCU, IBM Research