変える力を、ともに生み出す。 NTT DATAグループ

World Telecom Congress 2012
Workshop on "Cloud Computing in the Telecom Environment, Bridging the Gap"

A Filesystem Layer Data Replication Method for Cloud Computing

Masanori Itoh, Kei-ichi Yuyama, Kenjirou Yamanaka
March 4, 2012
NTT DATA CORPORATION

Agenda

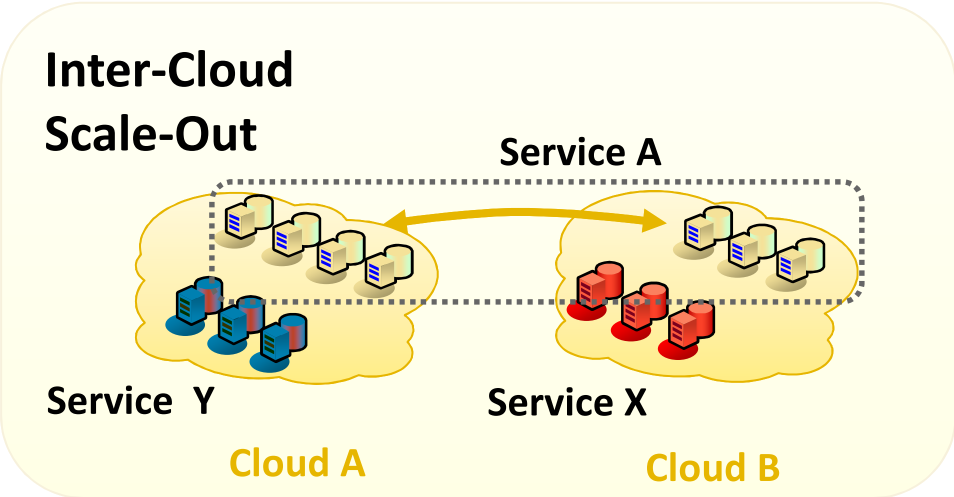
1

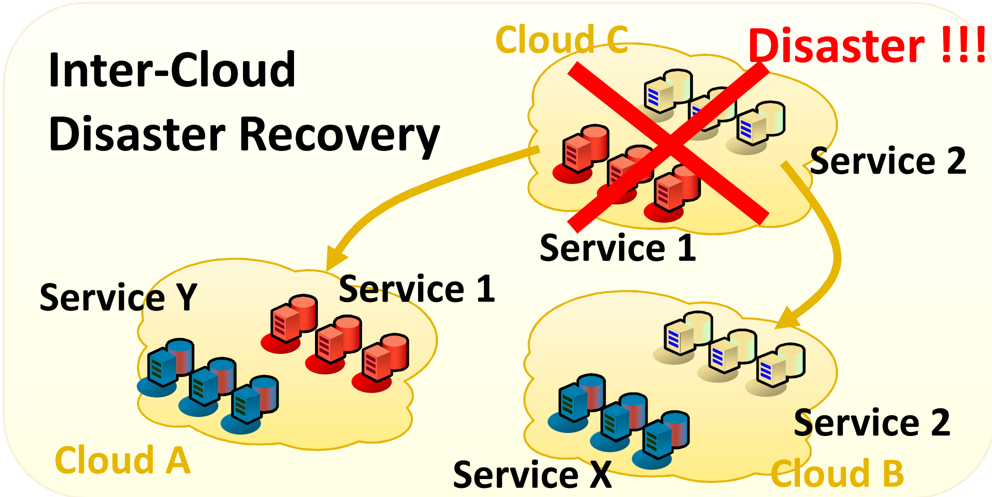
Background, Motivation and Goal

1. Background

- Social Background
  - Eastern Japan Disaster ('**東日本大震災**') on March 11, 2011
    - Strong Needs for Disaster Recovery
  - Un-predictable Computing / Networking Resource Demand
    - e.g., Systems for Checking People's Safety

- Technical Background
  - Demand for Inter-Cloud Federation Technology
    - Aggregating Resources of Multiple Cloud Systems
      - ✓ Inter-Cloud Scale-Out
      - ✓ Inter-Cloud Disaster Recovery





- Motivation
  - A National R&D Project Shooting for Inter-Cloud Scale-Out and Disaster Recovery in terms of Resource Control
    - The Project Achieved Enabling Computing / Networking Resource Federation among Heterogeneous Multiple Cloud Systems
      - Standardization Effort : GICTF (http://gitctf.jp)
  - But, we needed to address **the Tenant Data Replication Issue** in a Suitable Way for Inter-Cloud Computing Environment.

1. Goal

- Goal
  - An Efficient Mechanism Enabling Tenant Data Replication (e.g., Database, various Log Files, etc.) with Reasonable Trade Offs under Inter-Cloud Computing Environment
    - Need to Keep Replica(s) of Data as Up-to-Date as Possible
    - Immediate, Synchronous, … Replication Mechanism

1. Goal

- Requirements

1. **Performance**
    1. <u>Better Than Existing Solutions</u>
    2. Sufficient Replication Throughput even for Geologically Distributed Environment (e.g., Tokyo - Osaka)
2. **Minimum Impact to Wide Variety of (New/Existing) Systems**
    1. Minimum Software (esp. Application) Modifications
    2. Reasonable Operation Impact
3. **Cost Efficiency**
    1. No Expensive Special Hardware

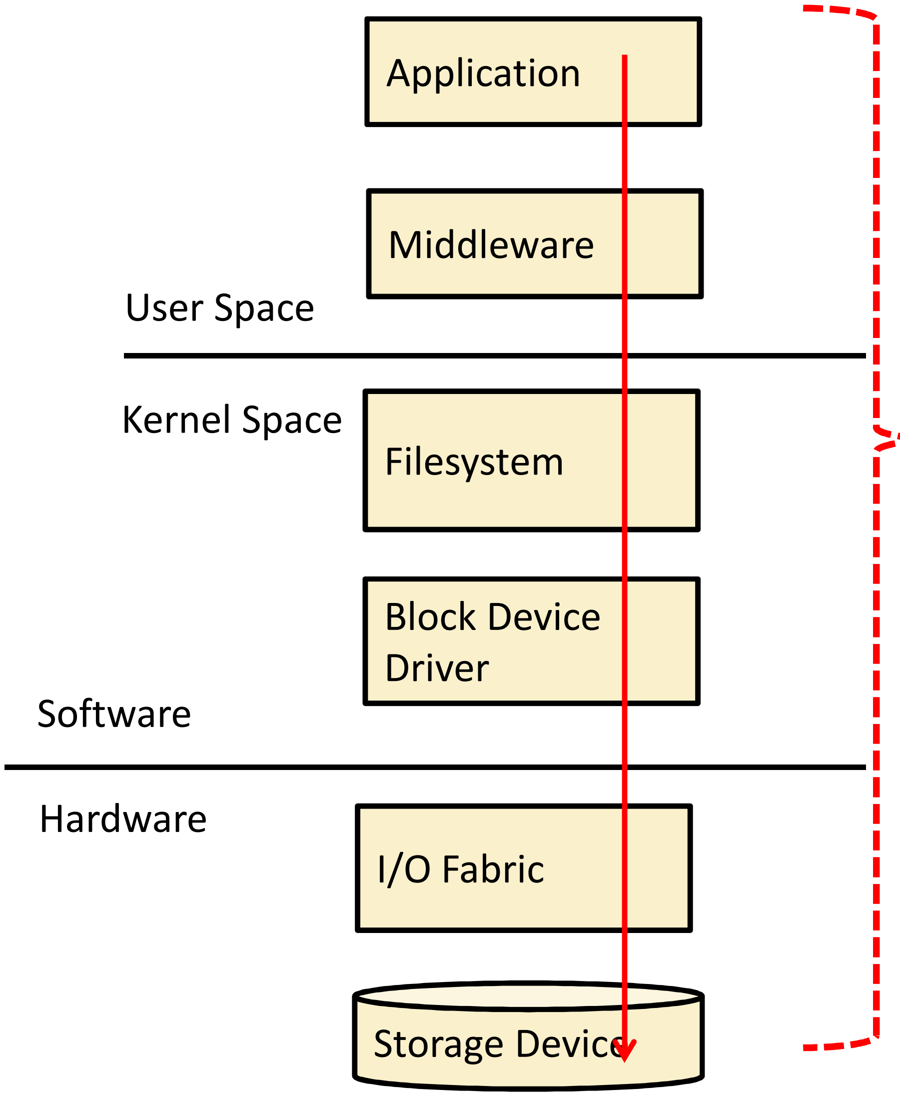
2

Overall Considerations and Related Works

2. Possible Layers

- Possible Layers　(Existing Solutions)
  - Application Layer
    - User Application Dependent Implementations
  - Middleware Layer
    - MySQL Cluster, PostgreSQL Streaming Replication, etc.
  - Block Device Layer
    - drbd
  - Hardware Layer
    - EMC SRDF, etc.

- **Overview of Possible Layers**



Example 1: Application Layer
Implemented by Application Developers

Example 2: Middleware Layer
MySQL Cluster,
PostgreSQL Streaming Replication, etc.

Need to Trap Data Issued by Applications Somewhere and Transfer to Geologically Distant Place(s)

Example 3: Block Device Layer
drbd (distributed redundant block device)

Example 4: Hardware Layer
EMC SRDF, etc.

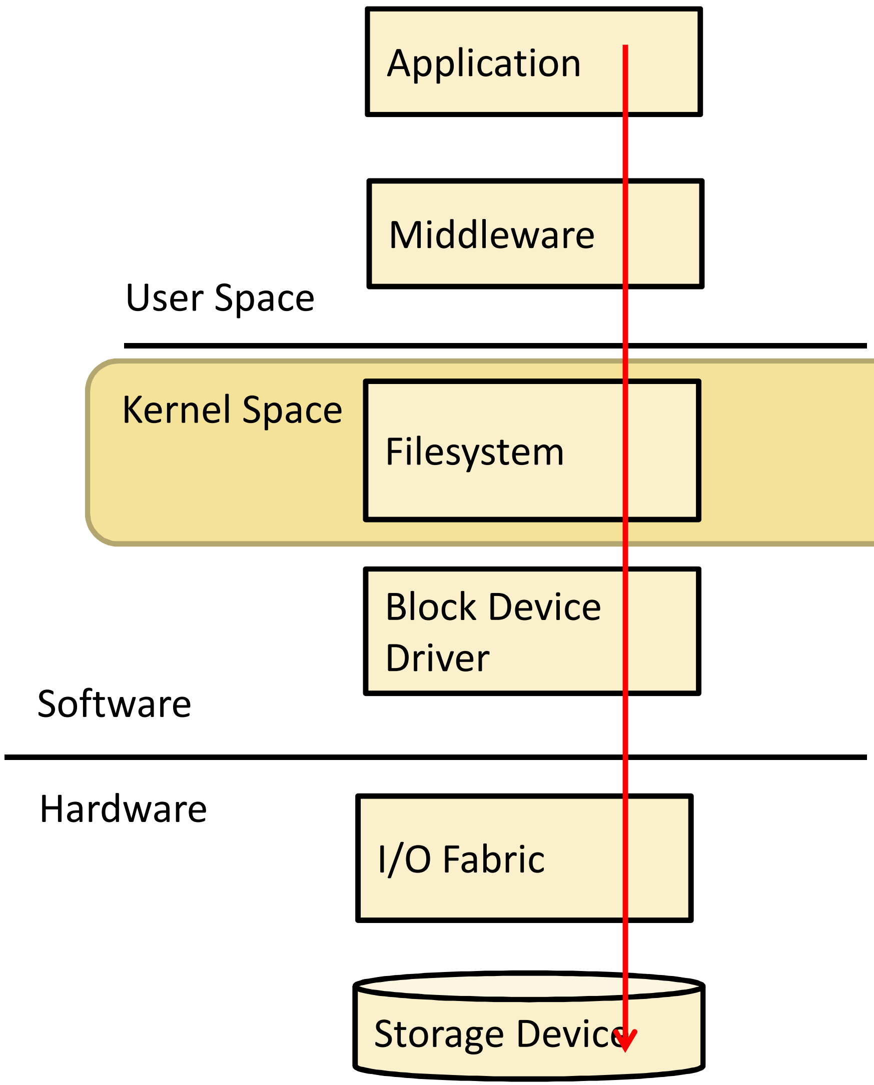
2. Related Works – Pros/Cons Analysis

変える力を、ともに生み出す。 NTT DATAグループ

- **Example 1/2 : Application/Middleware Layer : (e.g., MySQL Cluster)**
  - Pros Easy to Keep Consistency, Reasonable Performance
  - Cons Application/Middleware Dependent
- **Example 3 : Block Device Layer : (e.g., drbd)**
  - Pros Application/Middleware/File System Neutral <- Good !
  - Cons Poor Performance, Small Room to Optimize
- **Example 4 : Hardware Layer**
  - Pros Software Neutral
  - Cons Hardware Dependent, (Very Much) Expensive, Poor Performance, Very Small Room to Optimize

2. Related Works – Yet Another Layer

- **Overview of Possible Layers**



Example 1: Application Layer
Implemented by Application Devlopers

Example 2: Middleware Layer
MySQL Cluster,
PostgreSQL Streaming Replication, etc.

Example 5 : Filesystem Layer
'zfs send/recv' ?
Here is a room to do something…

Example 3: Block Device Layer
drbd (distributed redundant block device)

Example 4: Hardware Layer
EMC SRDF, etc.

2. Related Works – Pros/Cons Analysis

変える力を、ともに生み出す。 NTT DATAグループ

- Example 1/2 : Application/Middleware Layer : (e.g., MySQL Cluster)
  - Pros Easy to Keep Consistency, Reasonable Performance
  - Cons Application/Middleware Dependent
- Example 3 : Block Device Layer : drbd
  - Pros Application/Middleware/File System Neutral <- Good !
  - Cons Poor Performance, Small Room to Optimize
- Example 4 : Hardware Layer
  - Pros Software Neutral
  - Cons Hardware Dependent, (Very Much) Expensive,
    Poor Performance, Very Small Room to Optimize
- **Example 5 : File System Layer**
  - Pros Application/Middleware Neutral, Large Room to Optimize
  - Cons Needs Kernel Level Programming

3

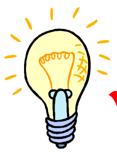
Problem Analysis and Basic Ideas

Poor Performance

- The Lower Layer a Replication Mechanism is Implemented, the More Sensitively its Throughput is Affected Under Geologically Distributed Environment (LFP).
- **Find Out the Best Place / Way to do Replication Work in terms of Performance.**
    - Not Sufficient Tenant Data Replication Performance against Network Line Investment

3. Problem Analysis and Basic Ideas

- **drbd Replication**
  - ☐ Transmits each (Random) Write I/O Request to the Remote Site
    - Inherently Uses Short Packets – Poor Throughput
  - ☐ Secure Replication is Provided by only Protocol C, which waits for I/O Completions at the Remote Site
    - Affects the Source Side I/O Requests Latency
- **Idea**
  - ☐ Make Use of Filesystem Journal
    - Naturally Converts Random (Write) I/Os into Sequential I/Os
    - Aggregates Multiple (Random) I/O Payloads

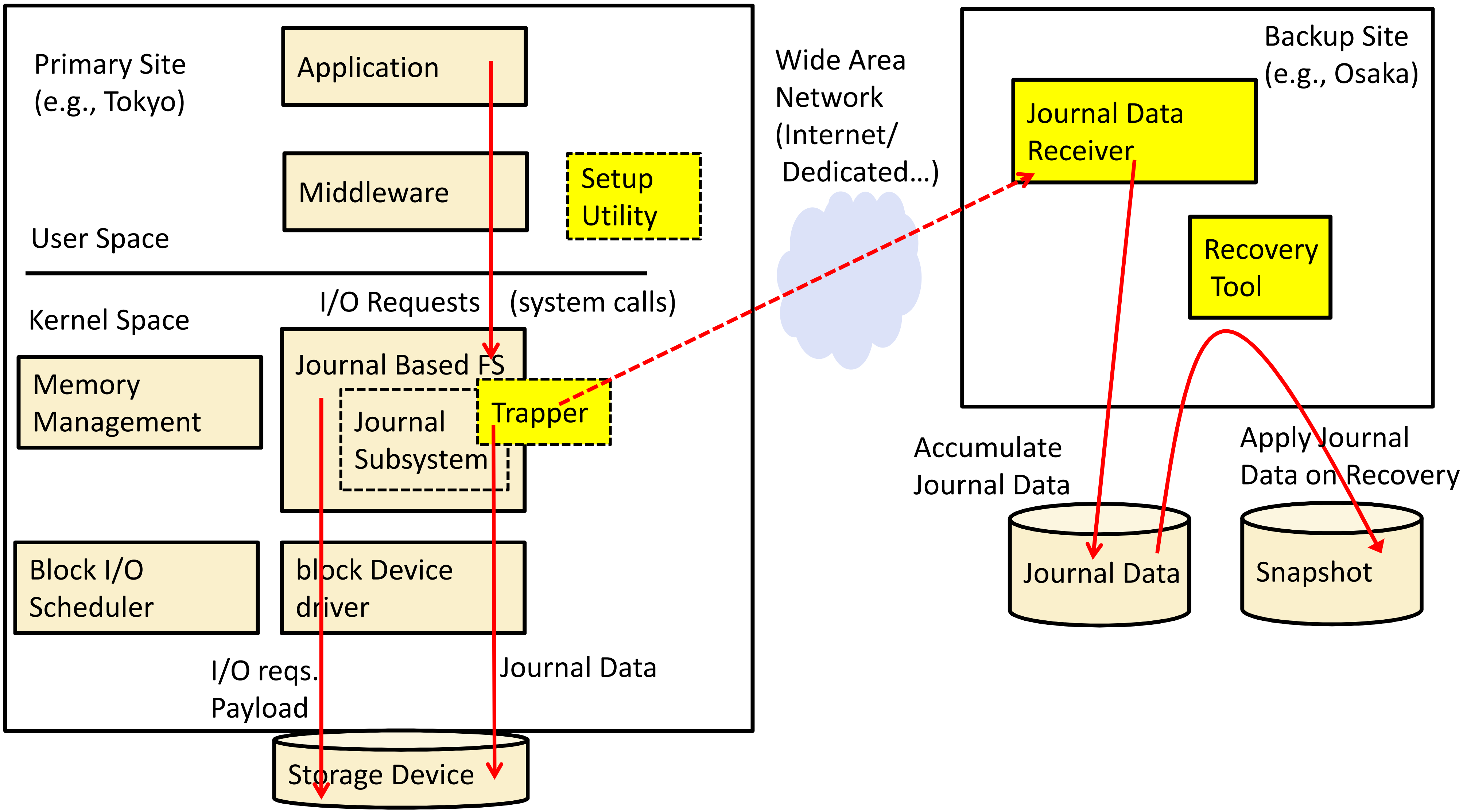


✓ Good Place to Implement Tenant Data Replication

4

Design and Implementation

- Overall Architecture



- Principles of Operation : Source Side
  1. Take a Snapshot of the Source File System (Partition Image (e.g., sdb1)) and Transfer it to the Remote Site
  2. Mount the Source File System
     - Establish (a) Connection(s) with the Remote Site
  3. Begin Journal Data Transfer (Both Meta Data and Filesystem Payload)
- Principles of Operation : Receiver Side
  1. Receive Journal Data and Store them Locally/Sequentially
- Principles of Operation : On Recovery
  1. Apply the Journal Data to the Snapshot

4. Design and Implementation

- Prototype Implementation
  - Base Platform
    - Fedora 14 (x86_64) + Fedora15 kernel (linux-2.6.37-2.fc15)
  - Base Filesystem
    - ext4 + jbd2
- Souce Lines
  - Trapper (Modified jbd2 driver) 4Ks
  - Setup Utility (user land) : 1Ks
  - Receiver (user land) : 4Ks
  - Recovery Tool (user land) : 1Ks
  - c.f. drbd source lines: kernel 30Ks + user land 30Ks

In Total, 10Ksteps
(Including bunch of debug codes)

- Optimizations in Prototype Implementation
  1. Use Multiple TCP Connections per Mount
     - ☐ Avoid Modification to TCP/IP Protocol Stack
  2. Overlapping Local Journal I/O and Transmission over TCP connections
     - ☐ Make Use of Parallelism and Issue Transmissions Frequently
  3. ext4 Mount Options with respect to Journaling

     data=ordered(default), data= journal, data=writeback
     - ☐ Created a Combined Mode of data=ordered and data=journal, and on the Source side:
       - Write metadata only
       - Transfer both metadata and data to the receiver side.

5

Evaluation

Features Test

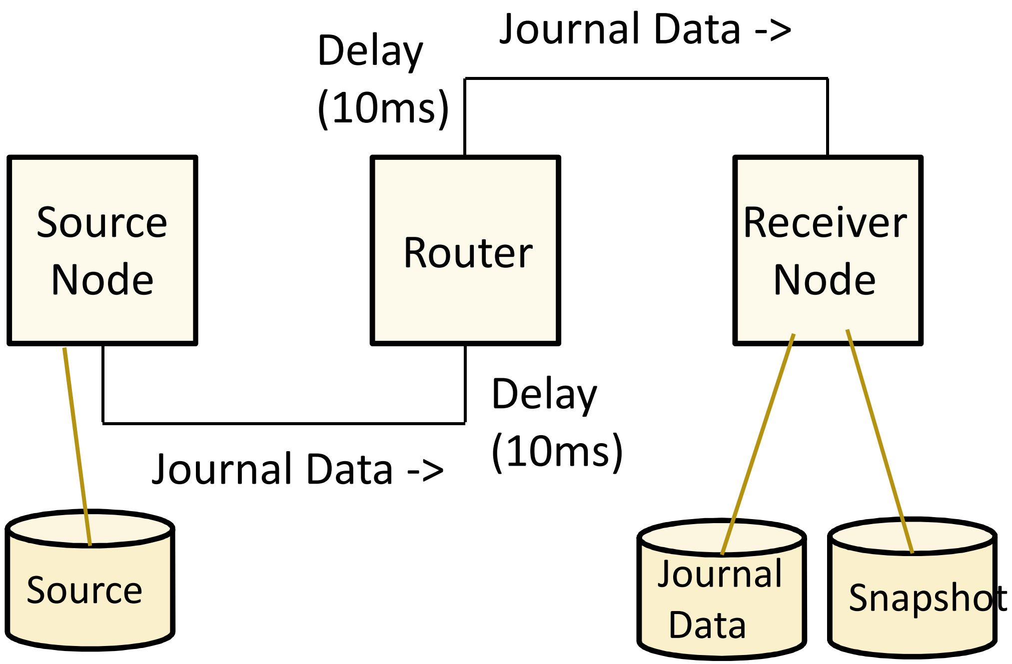
- Content of Files and Meta-data of them are Restored Correctly

Performance Measurement

- Hardware/Software
  - Xeon L5520 2P4C, 32GB, 146G SAS HDD (RAID 1) x 2, GbE NIC
  - 2Gbps FC RAID, 146GB Volume (RAID10)
  - Fedora14 (x86_64) + Modified Fedora 15 kernel (2.6.37-2.fc15)
- Network Delay Generator
  - Linux netem (i.e., ‘tc’ command)
- Benchmark
  - bonnie++ : 1.96
  - pgbench (Postgresql 9.1.0) , scaling factor= 256, clients=64

Emulated Geologically Distributed Environment

- One-Way Latency : 10ms via netem
  - ~ Tokyo - Osaka



I/O Pattern (Benchmark)

- bonnie++, pgbench

Receiver Side Behavior

- Receiver Sends Back ACKs after I/O Completion (Equivalent to drbd protocol C)

5. Evaluation Results : bonnie++

Performance Impact

- 10 Times Faster than Compared to DRBD Protocol C

	Sequential Write (block)
without Overlap, 1 connection	0.19 MB/s
without Overlap,10 connections	1.77MB/s
without Overlap, 500 connections	26MB/s
with Overlap, 500 connections	33MB/s
DRBD (Protocol C)	3.3 MB/s

10 Times Better !

Performance Impact

- 10 Times Faster than Compared to DRBD Protocol C

	Sequential Write (block)
without Overlap, 1 connection	0.19 MB/s
without Overlap, 10 connections	1.77MB/s
without Overlap, 500 connections	26MB/s
with Overlap, 500 connections	33MB/s
DRBD (Protocol C)	3.3 MB/s
Upper Limit	(GbE = 125MB/s)
No Replication (Base)	158 MB/s

Need to Investigate…

変える力を、ともに生み出す。
NTT DATAグループ

6

Future Works

6. Future Works

- More Detailed Analysis (Especially, Performance)
  - Packet Level Analysis, etc.
- Further Evaluation
  - Try Other Application Level Benchmarks
- Further Optimization
  - Optimizing Journal Data Transmission Timing
  - Use SSD on the Receiver Side
  - Multiple-Tier Replication Data Chaining
- Use Secure Communication Channel (SSL?)
- Integration with the Inter-Cloud Federation Manager
- Other File Systems (e.g., jfs2, zfs?)

7

Summary

7. Summary

- Proposed Technique
  - A Journal Based File System Layer Tenant Data Replication Method
- Features
  - ✓ Application/Middleware Transparent
    - Suitable for Inter-Cloud Computing Environment
  - ✓ High Performance
    - 10 Times better than drbd
    - Lots of Room for Further Optimization
  - ✓ Generically Applicable to Any Journal Based File Systems
  - ✓ Minimum Implementation Impact

Acknowledgement

- This work is funded by the Ministry of Internal Affairs and Communications（総務省）, Japan.

- **平成22年度　情報通信技術の研究開発**
  (FY2010 R&D of Information Communication Technologies)

  http://www.soumu.go.jp/menu_news/s-news/02tsushin03_000024.html

  Ⅲ クラウドサービスを支える高信頼・省電力ネットワーク制御技術の研究開発

  (R&D of Highly Available and Green Network Control Technology for Cloud Services)

  Ⅲ-1 高信頼クラウドサービス制御基盤技術

  (Technologies for Highly Available Cloud Service Control Foundation)

Thank You !

Global IT Innovator

NTT DATA GROUP

NTT DATA

Q&A