ROMA

User-Customizable NoSQL Database in Ruby

Rakuten, Inc., Rakuten Institute of Technology | Masaya Mori

Introduction

- 森 正弥（もり まさや）
- 楽天株式会社 執行役員
- 楽天技術研究所 所長
- 職掌
  - 開発部署のマネジメント
  - 研究開発の推進・統括

Masaya Mori

Twitter: @emasha

クラウド大全
The Complete Cloud Computing
（サービス詳細から基盤技術まで）
Google App Engine、Amazon EC2、Amazon S3、Windows Azure、Force.com、Hadoop、MapReduce、RDMA、Fairy、
Android、iPhone
広く深く知る一冊!
各界第一線の執筆陣による決定版!

ウェブ大変化
パワーシフトの始まり
森 正弥
"ネットのあちら側"からやってくる未来がこの本でよくわかる!
クラウドもツイッターもiPadも、すべてはこのうねりの中にある!
ウェブが変わる 社会も変わる

R.I.T.
Rakuten Institute of Technology

Rakuten Institute of Technology

Strategic R&D organization for Rakuten group

Concept

More Than Web

- Your great reality through emerging technologies -

Mission

Turning emerging and growing new technology seeds into new business/service opportunities to enrich the internet life (& real life) all over the world

Tokyo & NY

Global R&D has begun.

30 in Tokyo & 10 in NY

R.I.T.
Rakuten
Institute of Technology

Tokyo

New York

Issues and our R&D area

The internet has been growing to be diverse, huge, complicated and high-valued.

- On basis of that, we progress three following R&D area to provide solution in the near future.

Power

- Distributed computing
- High performance computing

Intelligence

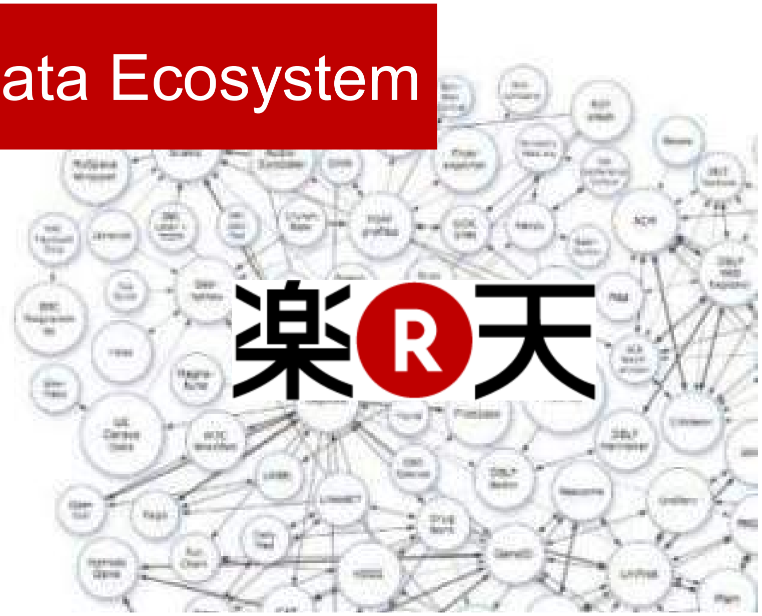
- Knowledge mining
- NLP / Recommender

Reality

- Multimedia Processing
- Ubiquitous / next UI

Unite, Contribution to Academia

Data Ecosystem



R&D symposium

Post / publish

Promote academic researchers
to exploit Rakuten’s web public data

Expected results

- share R&D with external researchers
- increase data & service awareness of people

Agenda

- Background
- Features of ROMA
- Overall architecture of ROMA
- Plug-in architecture and its domain specific language
- Conclusion

Internet Service

- User-driven service
  - Release is just a beginning.
  - As per user’s request, always Improve, always Advance.
- Software runs on Sever side
  - Can change any time
  - Big gap between package software and server side application
- Flexibility, Speed > Perfect
  - Lightweight Language
    - Ex. Ruby, Perl, Python, etc.
  - A.R.C.
  - Schema-less
    - CouchDB, MongoDB
  - Virtualization, Cloud

Internet technology

- Advantage of Open Source
  - Risk of vendor rock-on
  - Easy to start
  - Collective Inteligence, Collective Development
- Simple & Loose
  - HTTP
  - REST + JSON > SOAP or EBXML or EJB
  - MySQL, memcached
- Scalability
  - Load balancing
  - Cache
  - Distributed Cache
  - Disk I/O is always too slow

High Performance

- Importance of good performance
  - Amazon : 0.1s of latency -> down 1% sales
  - Google : 0.5s extra rending time -> drop 20% traffic
  - Generally, 1sec delay means ..
    - Down 11% Page Views
    - Down 7% Conversions
    - Down 16% Customer Satisfaction
  - Lost tens of millions yen by 5 minutes service down

- To achieve high performance website
  - Reduce size of HTML, Javascript
  - Local Cache, Ajax
  - On memory solution with consistency
  - SSD > HDD
  - Less latency
  - GPGPU

High availability

- 24/7
  - Downtime = Lose profit
- Redundancy
  - Power
  - Network
  - Load balancer
  - BGP
  - RAID
  - Replication and Backup
  - Data Center
  - Operator
- BCP for disaster
  - Big earchquake

Big Data, the direction

- New value from big data
  - Data mining
  - Suggestion, Recommendation
  - Personalization
- Technology to deal with big data
  - Cassandra
  - Hadoop
  - Lucene + Solr Cloud
  - It's referred to as **NoSQL** roughly.
    - Processing ex. MapReduce
    - Storing ex. KVS
- Search
  - Answer within 0.1s from over 70 million items
- BI
  - Can provide analyzed data to sales & marketing
  - Also provide tools reflecting knowledge of statics

Storing: NoSQL databases

- Needs for NoSQL databases are on the increase in order to easily store surging data

- Basic features of several NoSQL databases are
  - High scalability
  - High availability
  - High throughput
  - Other features vary by databases

- ROMA
  - Started development with Matz in 2007
  - Open sourced in Oct. 2009
    - See http://github.com/roma/roma/

ROMA Usage in Rakuten

- ROMA is used in various Rakuten services
  - ROMA runs on dozens of servers
  - Various types of data are stored
    - E.g. session data, personal page view history, etc

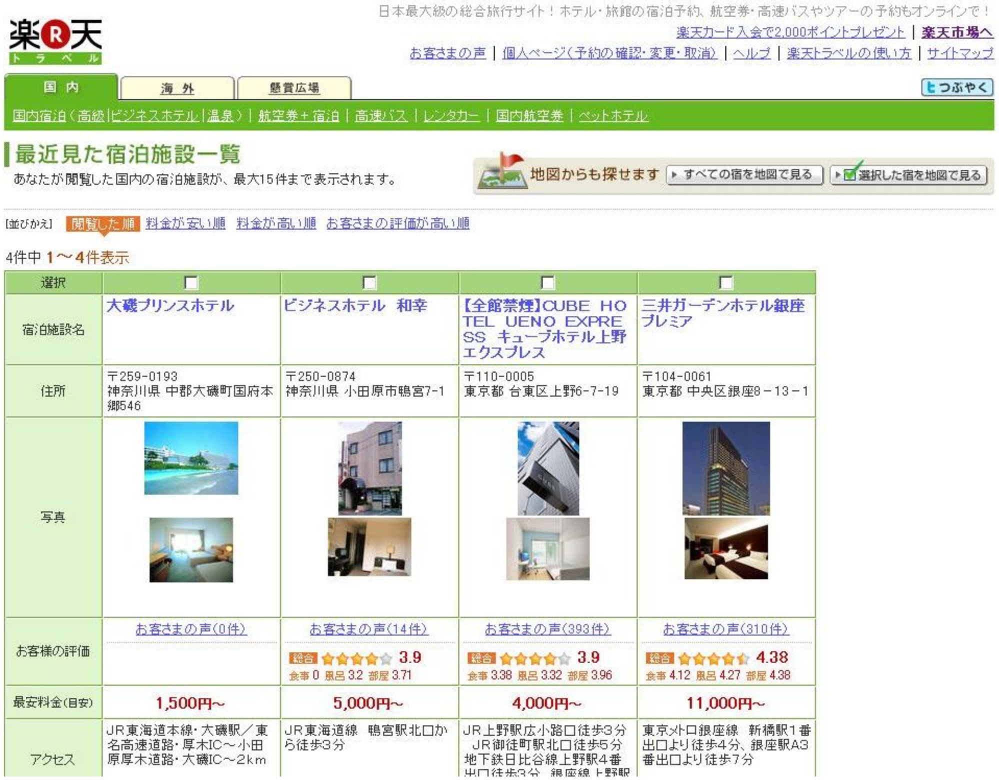
- Rakuten, Inc.
  - It provides many e-commerce platforms
    - E.g. Rakuten Ichiba, Rakuten Travel, Rakuten Books, etc
  - Rakuten has over 70 millions of users

Application-Specific Needs

- Many specific needs from application-side
  - These specific needs come from actual service development sites
- For example, users say
  - "How can we store structured data?"
    - Not value, but…. Map? List?
  - "How can we easily process stored data on DWH or Hadoop?"
  - "How can we delete null character that was stored somewhere by error?"
  - "How can we delete duplicate data that were accidentally stored somewhere, maybe by bugs of app?"
- We want to respond to all of these needs
  - We are trying to solve these problems, one by one.
  - Thereby, we hope we focus on what really matter on site.

Case Study: List Operation in Rakuten Travel

- “How can we easily access list data stored in ROMA?”
  - Request from Rakuten Travel to apply page view history using ROMA
  - Page view history function is useful function for users


楽天トラベル
楽天カード入会で2,000ポイントプレゼント | 楽天市場へ
お客さまの声 | 個人ページ（予約の確認・変更・取消） | ヘルプ | 楽天トラベルの使い方 | サイトマップ
国 内
海 外
懸賞広場
つぶやく
国内宿泊（高級|ビジネスホテル|温泉） | 航空券＋宿泊 | 高速バス | レンタカー | 国内航空券 | ペットホテル
地図からも探せます
すべての宿を地図で見る
選択した宿を地図で見る
[並びかえ] 閲覧した順 料金が安い順 料金が高い順 お客さまの評価が高い順

日本最大級の総合旅行サイト！ホテル・旅館の宿泊予約、航空券・高速バスやツアーの予約もオンラインで！

最近見た宿泊施設一覧

あなたが閲覧した国内の宿泊施設が、最大15件まで表示されます。

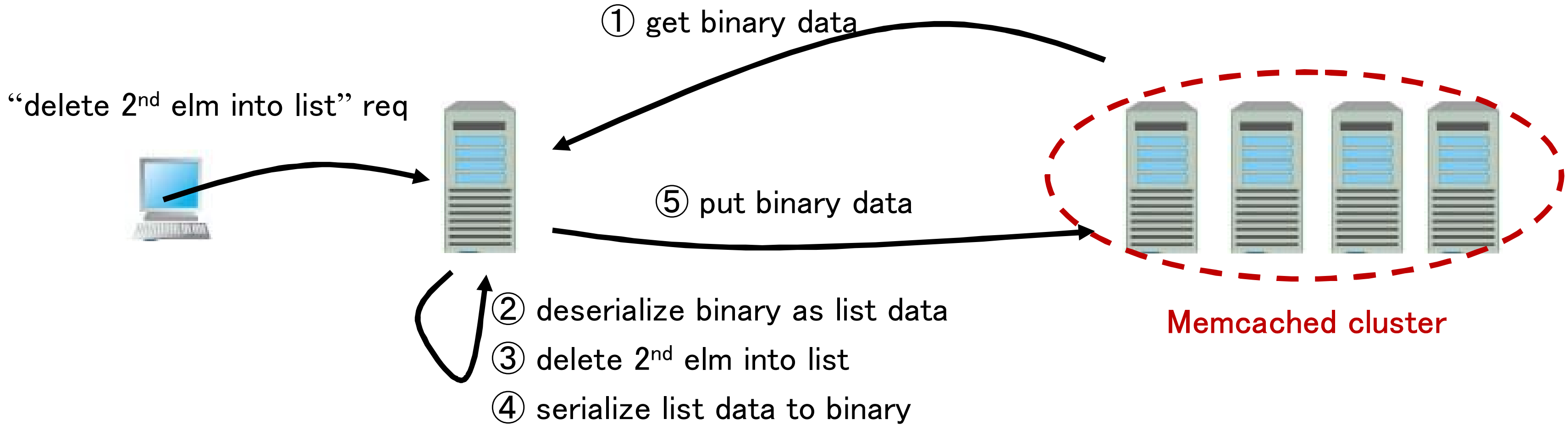
4件中 1～4件表示

選択	□	□	□	□
宿泊施設名	大磯プリンスホテル	ビジネスホテル　和幸	【全館禁煙】CUBE HOTEL UENO EXPRESS キューブホテル上野エクスプレス	三井ガーデンホテル銀座プレミア
住所	〒259-0193 神奈川県 中郡大磯町国府本郷546	〒250-0874 神奈川県 小田原市鴨宮7-1	〒110-0005 東京都 台東区上野6-7-19	〒104-0061 東京都 中央区銀座8－13－1
写真				
お客様の評価	お客さまの声（0件）	お客さまの声（14件） 総合 ★★★★☆ 3.9 食事 0 風呂 3.2 部屋 3.71	お客さまの声（393件） 総合 ★★★★☆ 3.9 食事 3.38 風呂 3.32 部屋 3.96	お客さまの声（310件） 総合 ★★★★☆ 4.38 食事 4.12 風呂 4.27 部屋 4.38
最安料金（目安）	1,500円～	5,000円～	4,000円～	11,000円～
アクセス	JR東海道本線・大磯駅／東名高速道路・厚木IC～小田原厚木道路・大磯IC～2km	JR東海道線　鴨宮駅北口から徒歩3分	JR上野駅広小路口徒歩3分　JR御徒町駅北口徒歩5分　地下鉄日比谷線上野駅4番出口徒歩3分　銀座線上野駅	東京メトロ銀座線　新橋駅1番出口より徒歩4分、銀座駅A3番出口より徒歩7分

Case Study: List Operation in Rakuten Travel

- The application stores list data for users in database
  - Key: user ID, Value: a list of pages that the user viewed

- In such case as memcached,
  to delete list data stored in NoSQL database, application…
  1. Gets binary data of specified key
  2. Deserializes it as list data
  3. delete the list data according to user requirement
  4. Serializes the list data to binary data
  5. Set it to database


① get binary data
"delete 2nd elm into list" req
⑤ put binary data
② deserialize binary as list data
③ delete 2nd elm into list
④ serialize list data to binary
Memcached cluster

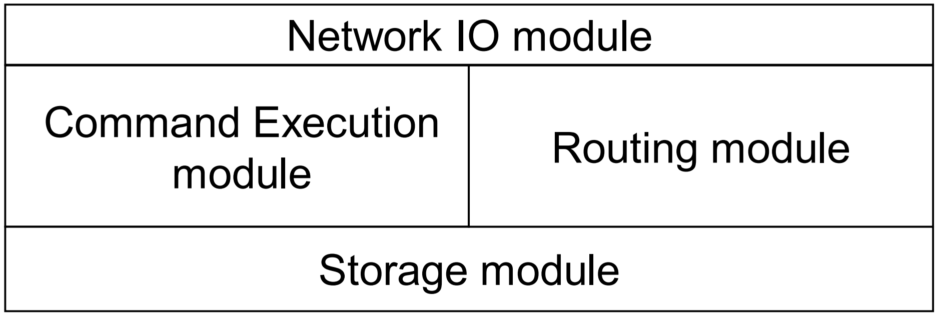
ROMA

- User-customizable NoSQL database in Ruby
- Features
  - **Key-value model**
  - High scalability
  - High availability
  - Fault-tolerance
  - Better throughput
  - And…
- To meet application-specific needs, ROMA provides
  - Plug-in architecture
  - Domain specific language (DSL) for Plug-in
- ROMA enables meeting the above need in Rakuten Travel

Overall Architecture

- ROMA integrates several well-known techniques to achieve scalability, availability and fault-tolerance
  - For example, consistent hashing, virtual nodes, chain replication-like mechanism, lamport clocks, etc

- ROMA node consists of 4 modules
  - Network IO module: Receiving data from clients and other ROMA nodes
  - Command exec module: Creating and executing commands
  - Routing module: Maintaining ring information
  - Storage module: Storing data



Network IO module

Command Execution module

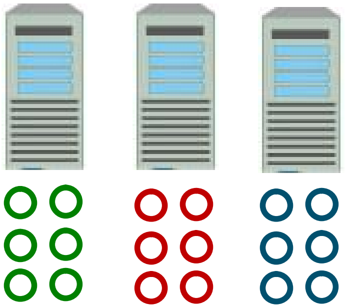
Routing module

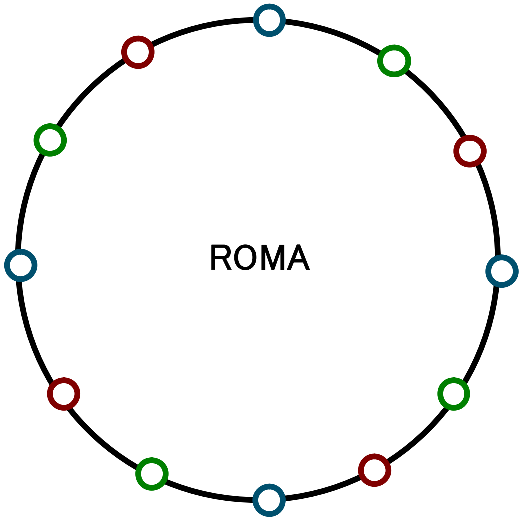
Storage module

Data Partitioning

- Consistent hashing and virtual nodes
  - ROMA consists of several nodes that run on servers
  - Many virtual nodes are allocated on 1-dimensional hash space of 32-bits
- Each virtual node has a 32-bits ID
  - To determine which ROMA node to store key in.
  - SHA-1 hash

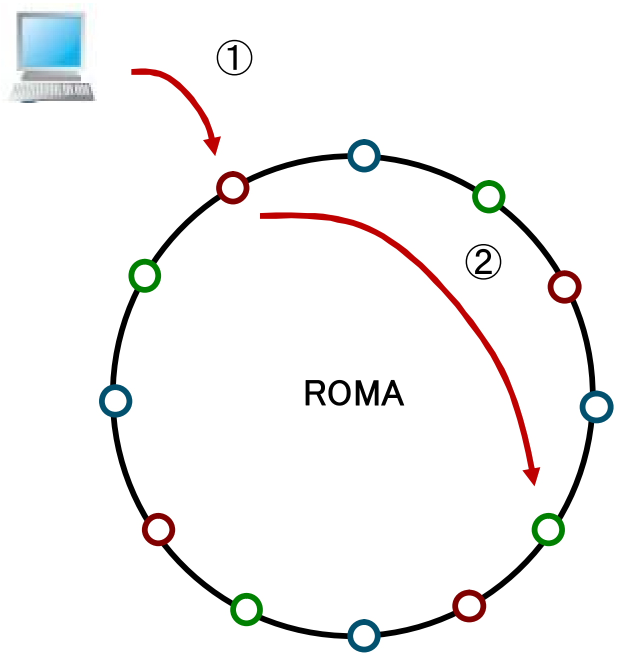
E.g. ROMA consists of three nodes



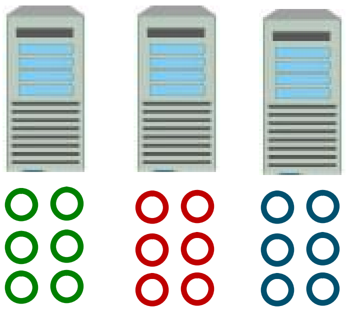


Data Accessing

- In getting value of specified key from ROMA
  1. User accesses to ROMA nodes
  2. The node determines others that are responsible for value of specified key
  3. The node gets the value from the other node
  4. The node returns it to user



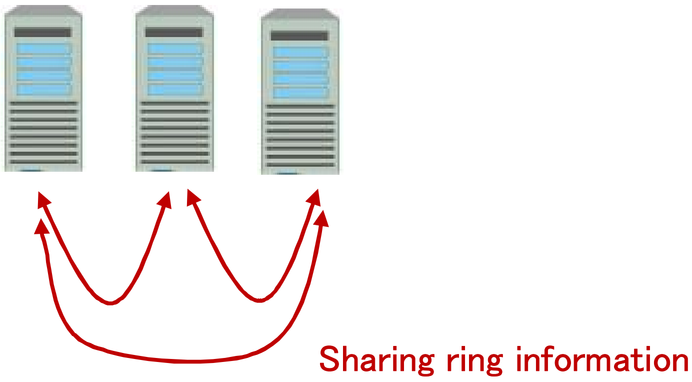
E.g. ROMA consists of three nodes

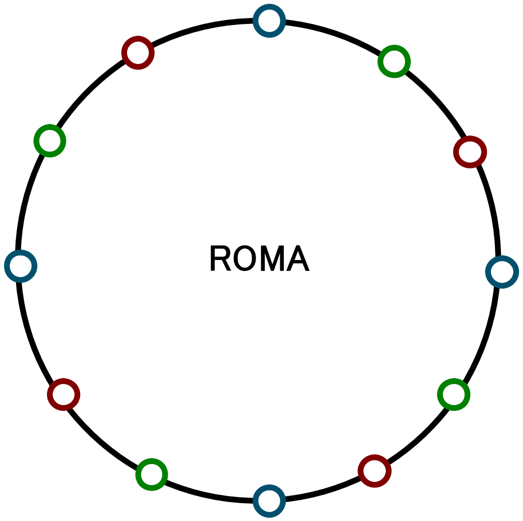


Sharing routing table

- Each node maintains and periodically shares routing table with others
  - Routing tables: range of hashs, machines, port
  - If several versions of routing table exist, node updates the latest version.
  - Lamport clocks and Merkle hash tree
- ROMA node multicasts with others to share routing tables
  - We use multicast though being aware of scalability.
  - Current version of ROMA doesn’t use gossip-based protocol.

E.g. ROMA consists of three nodes



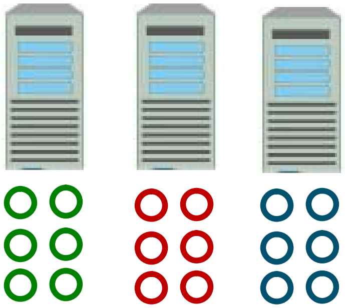


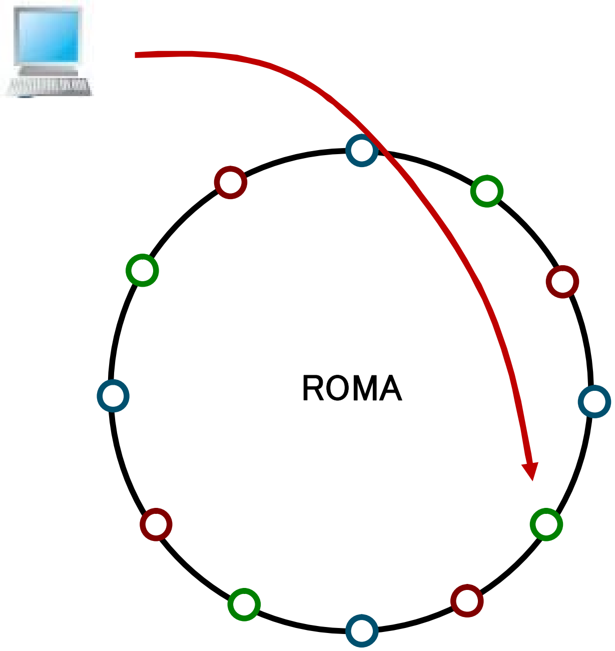
Client has cache of routing table

- ROMA client enables direct accessing to data
  - Client has a cache of routing table.
  - It checks to see if routing table is updated or not every 3 sec.

- In getting value of specified key, ROMA client
  1. Determines nodes according to cache
  2. Gets the value from the node directly

E.g. ROMA consists of three nodes



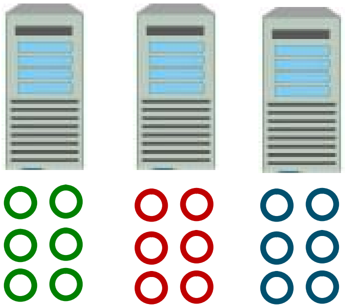


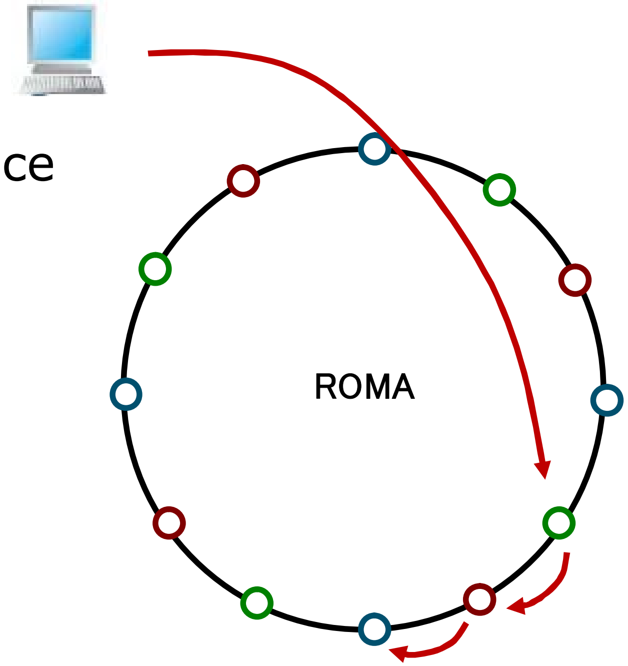
Data Replication

- Automatic data replication
  - Client waits until data replication finishes successfully.
  - If data replication failed, the data is push to asynchronous queue in node, which it will retry replication.
  - Eventual consistency

- Each data is replicated at N nodes
  - N: this parameter is configured in advance

E.g. ROMA consists of three nodes

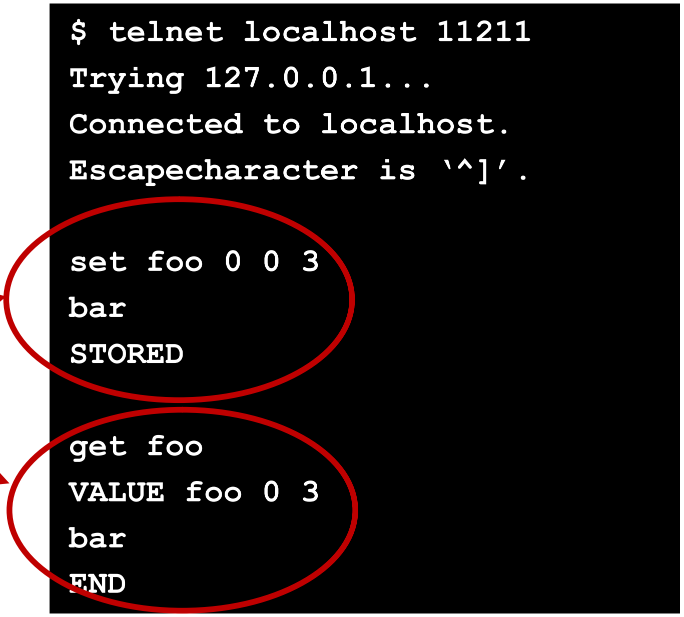




Message Protocol

- Extended Memcached protocol over TCP
  - Used between clients and ROMA nodes

- Memcached client libraries are also available.
  - Without distribution concern
    - User can access any node and ROMA forwards it later.
  - Users can use telnet.



```
$ telnet localhost 11211
Trying 127.0.0.1...
Connected to localhost.
Escapecharacter is '^]'.

set foo 0 0 3
bar
STORED

get foo
VALUE foo 0 3
bar
END
```

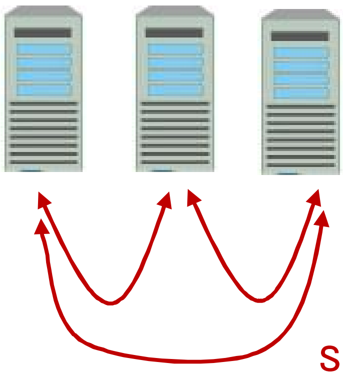
User can set/get data in ROMA
with telnet like memcached

Failure Detection

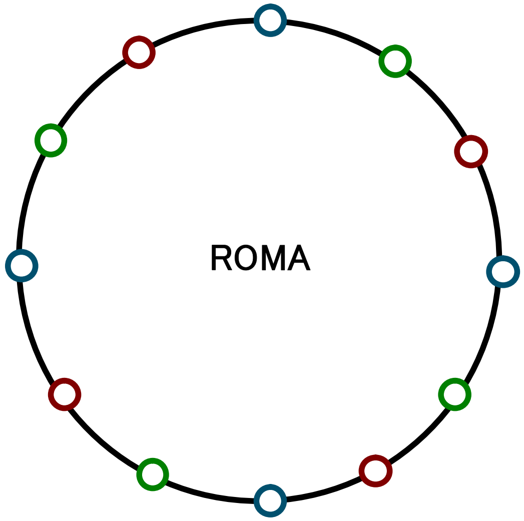
- Heartbeat detection
  - Each node multicasts periodic heartbeat with others
  - Heartbeat is flooded every 1 sec.

- If heartbeat is missed continuously, the node is declared as failed
  - Failover
  - Removal of the node

E.g. ROMA consists of three nodes

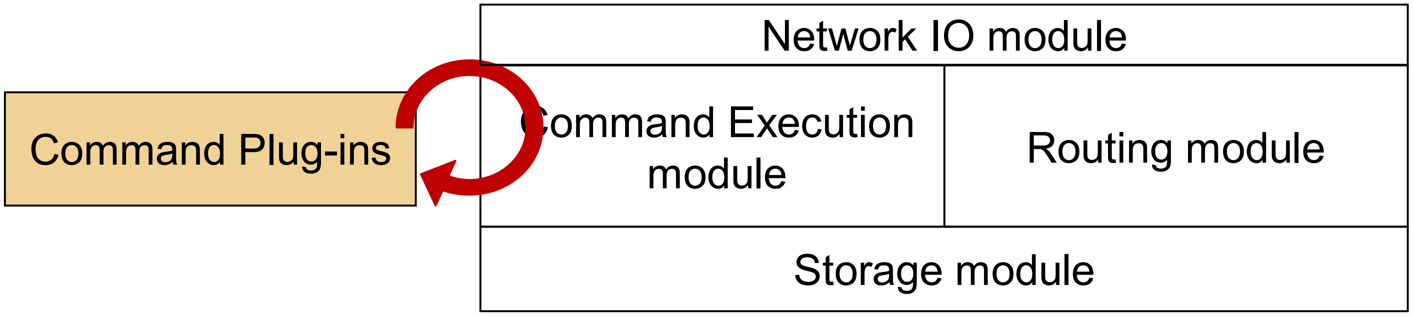


Sending heartbeat



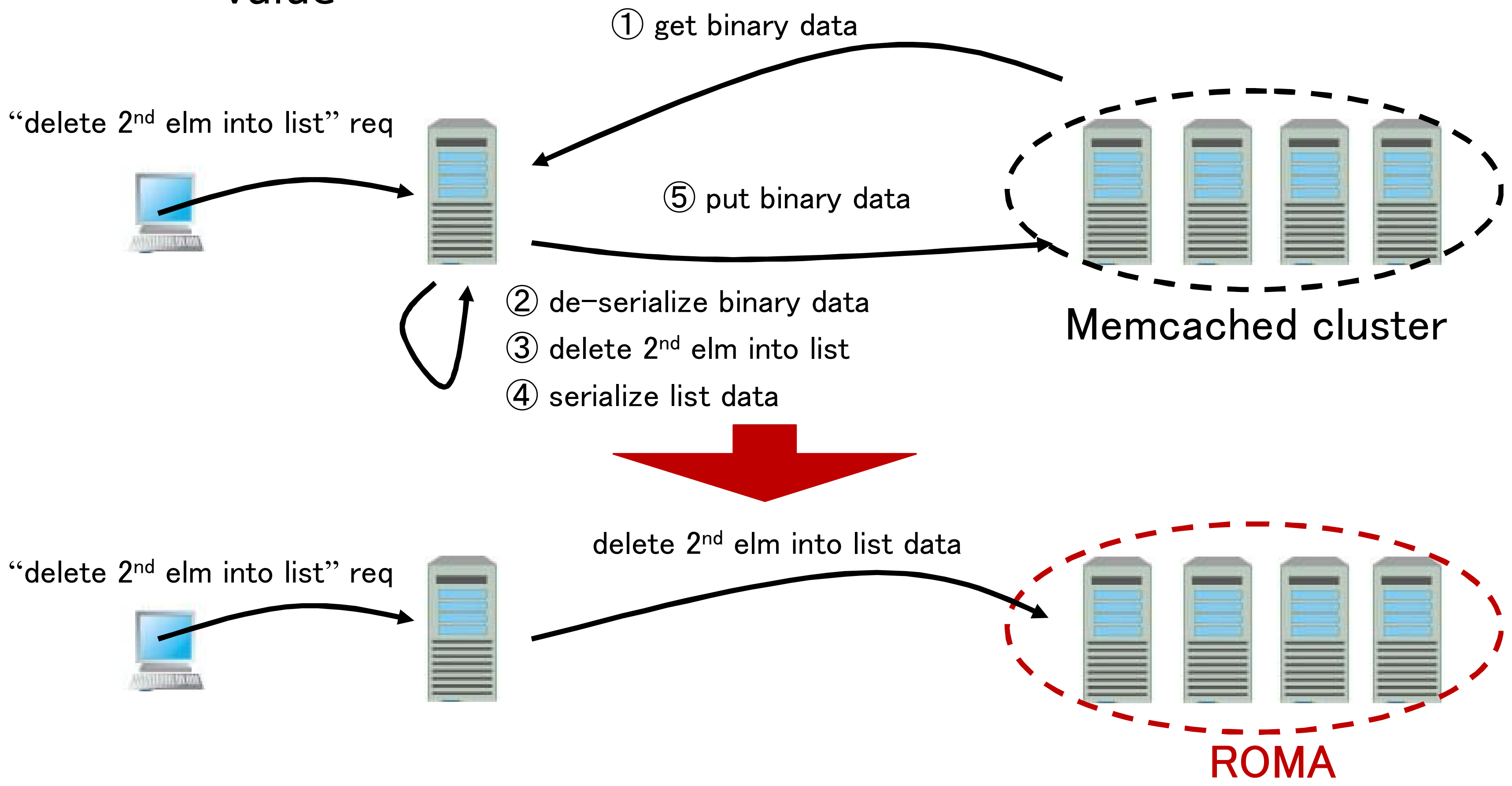
Plug-in Architecture

- Plug-ins allow users to extend behavior of ROMA
- For example,
  - Command plug-ins enable to change behavior of command module
  - Users can append user-defined commands to ROMA
  - Current version provides plug-ins for command module only
    - Plug-ins for other modules coming soon
    - For example, storage-plug-in.



Case Study: Commands for List Operations

- ROMA allows defining commands for list operations as plug-ins
  - Users can atomically access list data stored in ROMA as value



① get binary data

"delete 2nd elm into list" req

⑤ put binary data

② de-serialize binary data

③ delete 2nd elm into list

④ serialize list data

Memcached cluster

"delete 2nd elm into list" req

delete 2nd elm into list data

ROMA

Case Study: Commands for List Operations

```
# alist_insert <key> <index> <bytes> [forward]¥r¥n
# <data block>¥r¥n
#
# (STORED|NOT_STORED|SERVER_ERROR <error message>)¥r¥n
def ev_alist_insert(s)
  hname, k, d, vn, nodes = calc_hash(s[1])
  data = read_bytes(s[3].to_i); read_bytes(2)
  return forward2(nodes[0], s, data) if nodes[0] != @nid
  ddata = @storages[hname].get(vn, k, d)
  v = [[], []] unless ddata
  v = Marshal.load(ddata) if ddata
  v[0].insert(s[2].to_i, data)
  v[1].insert(s[2].to_i, Time.now.to_i)
  expt = 0x7fffffff
  ret = @storages[hname].set(vn, k, d, expt, Marshal.dump(v))
  @stats.write_count += 1
  if ret
    redundant(nodes[1..-1], hname, k, d, ret[2], expt, ret[4])
    send_data("STORED¥r¥n")
  end
  send_data("NOT_STORED¥r¥n") unless ret
end
```

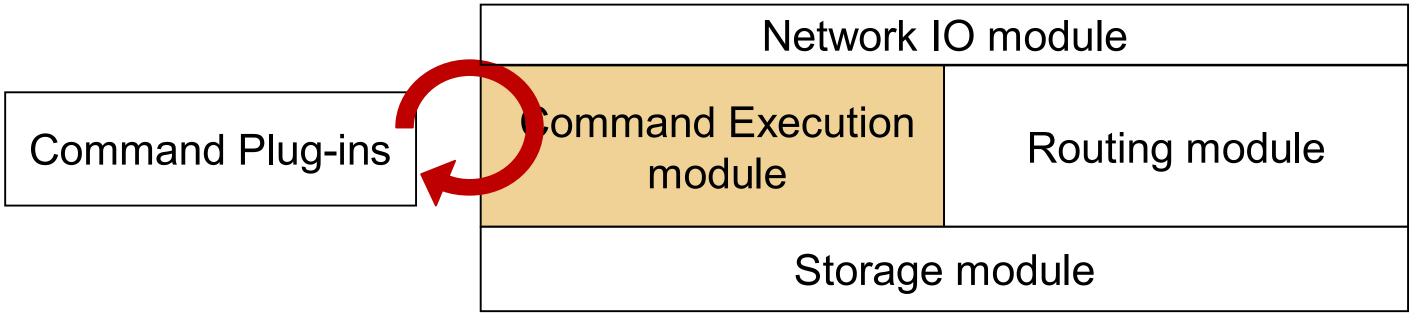
Method declaration for list command named “alist_insert”

As for lines, like this.

User can use also Telnet.

Command Plug-in Mechanism

- Command exec module
  - Loads plug-in and registers plug-in method at startup
    - Plug-in method has method name starting with `ev_`
    - For example, `ev_alist_insert` is plug-in method
    - Ruby allows adding new methods to classes dynamically
  - In calling plug-in method
    1. Receives data from network IO module and creates new command
    2. Finds registered plug-in method responding to the command
    3. Calls plug-in method
    - Uses `send` method provided by Ruby.



DSL for Command Plug-in

- DSL enables users to simply declare commands
  - Without distribution concern (data replication, data partitioning)
  - `def_write_command_with_key_value`
    - Allows easily defining commands for storing structured data in ROMA
- For example,
  - User can declare a `alist_insert` command with DSL

```
# alist_insert <key> <index> <bytes> [forward]¥r¥n
# <data block>¥r¥n
#
# (STORED|NOT_STORED|SERVER_ERROR <error message>)¥r¥n
def_write_command_with_key_value :alist_insert, 3 do |ctx|
  v = [[], []]
  v = Marshal.load(ctx.stored.value) if ctx.stored
  v[0].insert(ctx.argv[2].to_i, ctx.params.value
  v[1].insert(ctx.argv[2].to_i, Time.now.to_i)
  expt = 0x7fffffff
  [0, expt, Marshal.dump(v), :write, 'STORED']
end
```

Commands for Map Operations with DSL

- Another example: User can declare map_set command with DSL
  - User can store map data in ROMA as value of specified key

```
# map_set <key> <mapkey> <flags> <expt> <bytes>¥r¥n
# <data block>¥r¥n
#
# (STORED|NOT_STORED|SERVER_ERROR <error message>)¥r¥n
def_write_command_with_key_value :map_set, 5 do |ctx|
  v = {}
  v = Marshal.load(ctx.stored.value) if ctx.stored
  v[ctx.argv[2]] = ctx.params.value
  expt = ctx.argv[4].to_i
  if expt == 0
    expt = 0x7fffffff
  elsif expt < 2592000
    expt += Time.now.to_i
  end
  [0, expt, Marshal.dump(v), :write, 'STORED']
end
```

DSL Mechanism

- `def_write_command_with_key_value` is declared as method in Ruby

```
def def_write_command_with_key_value(cmd, idx_of_val_len, forward = :one_line, &block)
  define_method "ev_#{cmd}" do |s|
    params = CommandParams.new
    params.key, params.hash_name = s[1].split("¥e")
    params.digest = Digest::SHA1.hexdigest(params.key).hex % @rttable.hbits
    params.vn = @rttable.get_vnode_id(params.digest)
    params.nodes = @rttable.search_nodes_for_write(params.vn)
    params.value = read_bytes(s[idx_of_val_len].to_i)
    read_bytes(2)
    stored = StoredData.new
    stored.vn, stored.last, stored.clk, stored.expt, stored.value =
        @storages[params.hash_name].get_raw(params.vn, params.key, params.digest)
    stored = nil if stored.vn == nil || Time.now.to_i > stored.expt
    ctx = CommandContext.new(s, params, stored)
    ret = instance_exec(ctx, &block)
    if ret
      redundant(ctx.params.nodes[1..-1], ctx.params.hash_name,
          ctx.params.key, ctx.params.digest, ret[2], expt, ret[4])
      send_data("#{msg}¥r¥n")
    end
    send_data("NOT_#{msg}¥r¥n") unless ret
end
```

Conclusion

- Background
  - Spread of NoSQL databases
  - application-specific needs
- Features of ROMA
  - To respond to specific needs, ROMA provides user-customizable interfaces
  - Plug-in architecture and its domain specific language
- Overall architecture of ROMA
  - It integrates several well-known techniques
    - Consistent hashing, chain replication-like mechanism, lamport clocks, etc
- Plug-in architecture and DSL
  - Plug-ins allow enhancing behavior of ROMA easily

Thank you for kindly attention

楽天

乐酷天

TARAD.com

Buy.com

PRICEMINISTER
l'Achat - Vente Garanti

Rakuten BETA
BELANJA ONLINE