2110625 - Data Science Architecture

Document Store

Asst.Prof. Natawut Nupairoj, Ph.D.
Department of Computer Engineering
Chulalongkorn University
natawut.n@chula.ac.th

CHULA ENGINEERING
Foundation toward Innovation

Document Store

เรียกว่า transaction base

- Data storage system designed for storing, retrieving and managing document (aka. semi-structured data)

- Data is stored as a document or item without any link or relation to other table

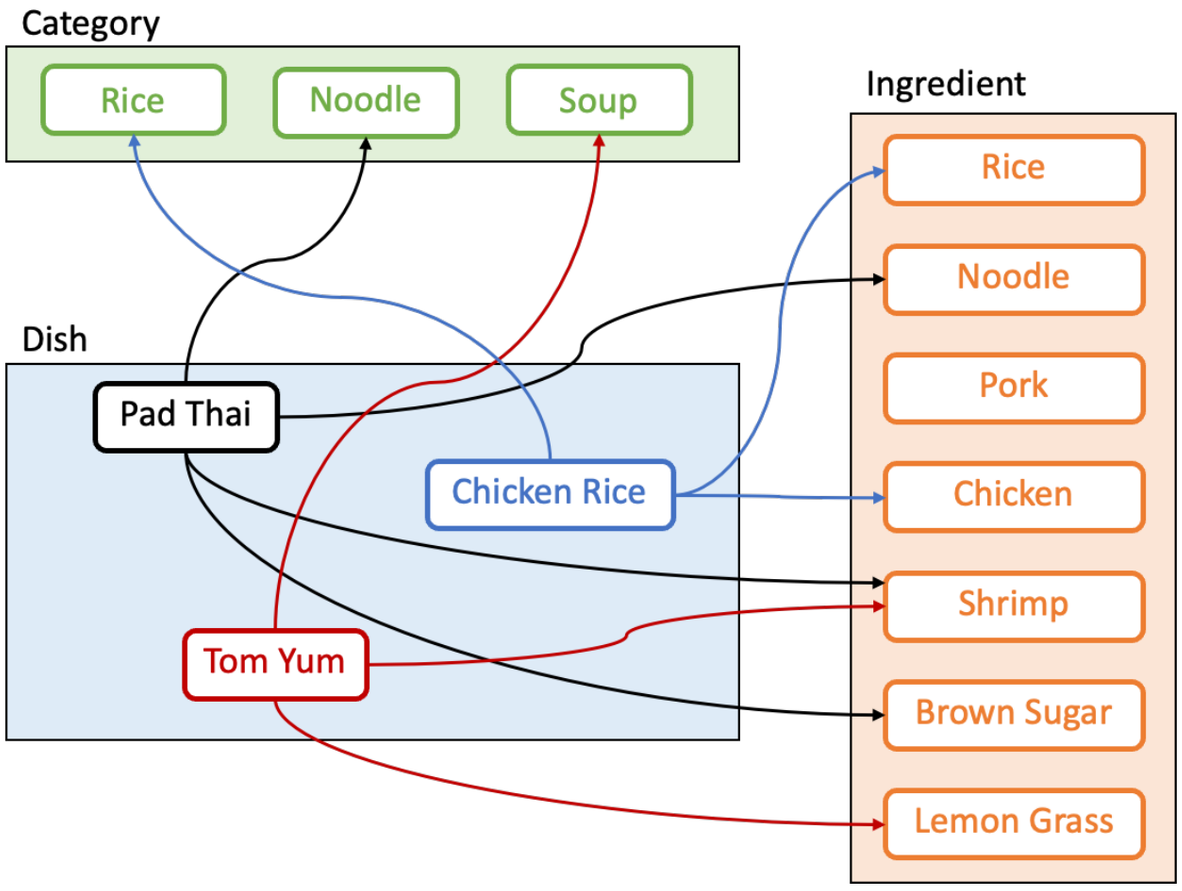
- Document store usually runs on a cluster of multiple (10s-1000s) storage nodes to allow horizontal scalability

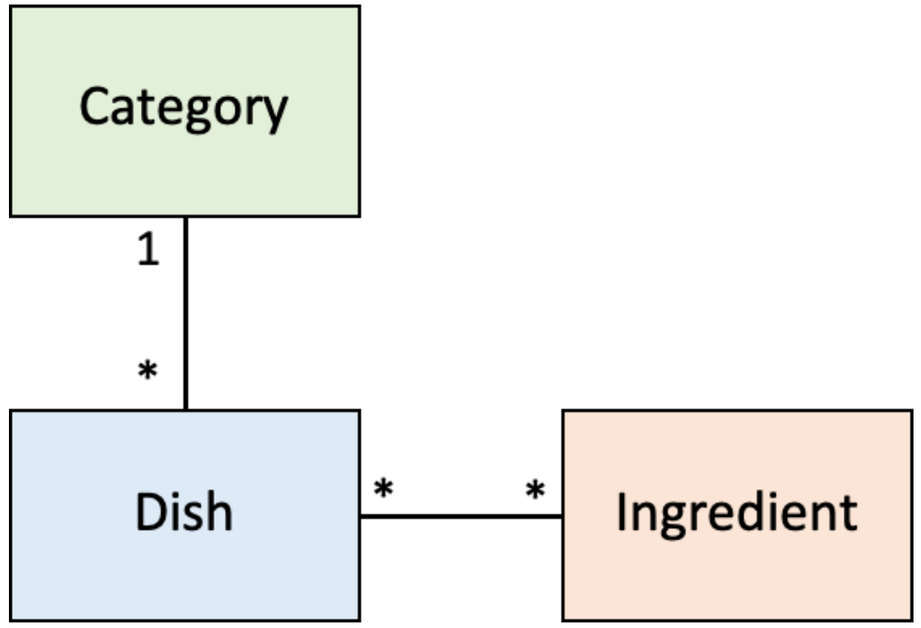
sharding / partitioning

- Shards or partitions are fundamental mechanism for both scaling and replicated capabilities

ทำให้ สำรองข้อมูล

Recipe Database Revisited

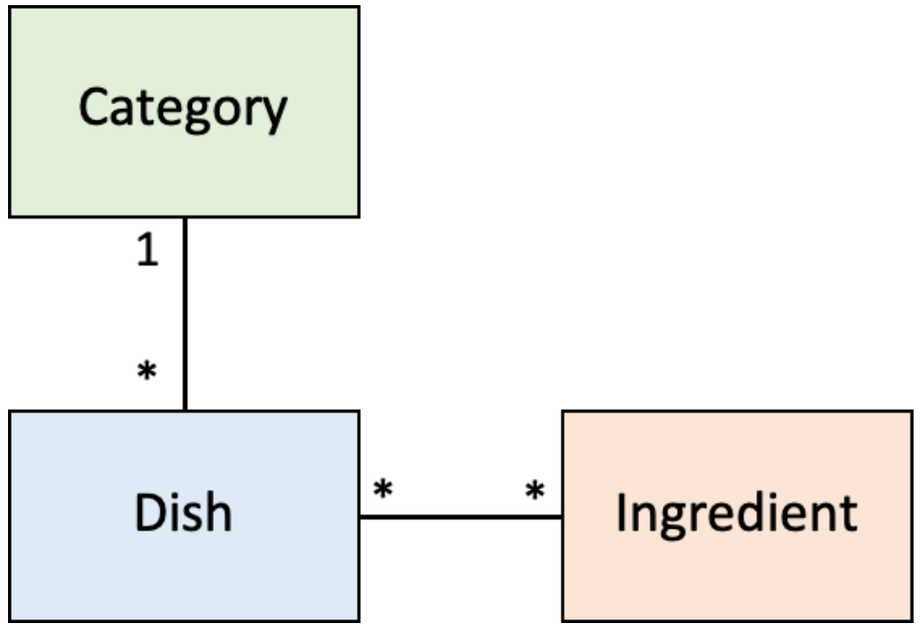




Example RDBMS: Recipe

Category

id	category
1	Soup
2	Noodle
3	Rice



Dish

id	name	category	…
1203	Pad Thai	2	
1288	Chicken Rice	3	

DI_link

d_id	i_id	quantity	unit
1203	45	125	g
1203	48	150	g
1203	52	3	tbsp

Ingredient

id	item
45	noodle
46	rice
48	chicken
52	brown sugar

Example MongoDB: Recipe

scale out ไม่ขอบ dependencies เพราะต้องคุยข้ามเครื่อง ทำให้ช้าลง
เผา data ขาดออกจากกันให้ได้ doc จบตัวมันเอง

```
{
  _id: ObjectId("507f191e810c19729de860ea"),
  name: "Pad Thai",
  cuisine: "Thai",
  category: "Noodle",
  rating: 4.7,
  votes: 9301,
  ingredients: [
    { item: "noodle", quantity: "125", unit: "g" },
    { item: "chicken", quantity: "150", unit: "g" },
    { item: "brown sugar", quantity: "3", unit: "tbsp" }
    ...
  ]
}
```

Document Store
MongoDB

MongoDB

- Document-Oriented NoSQL database
- Based on CP concept CP เน้น Consistancy แล้ว scale out ได้ดี
- BSON store (binary-format JSON)
- Databases – Collections - Documents
- Schemaless - documents in the same collections can have different structures ไม่จำเป็น ที่ทุก record มี field เหมือนกัน (ยกเว้น required เช่น alpha id)
- Query using javascript functions
- Support load-balancing, replication, scale-out, etc.

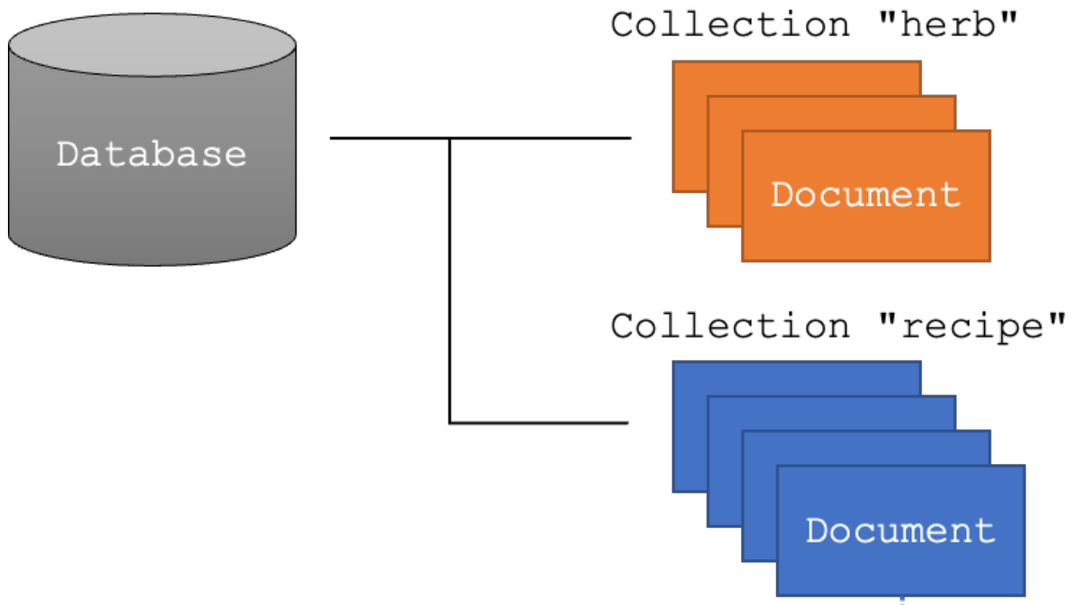
RDBMS vs. MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Row / Tuple	Document
Column	Field
Table Join เป็นอุปสรรค การ scale out	Embedded Document
Primary Key	Primary Key (Default key “_id” provided by mongodb itself)

Example Document: Recipe

```
{
  _id: ObjectId("507f191e810c19729de860ea"),
  name: "Pad Thai",
  cuisine: "Thai",
  category: "Noodle",
  rating: 4.7,
  votes: 9301,
  ingredients: [
    { item: "noodle", quantity: "125", unit: "g" },
    { item: "chicken", quantity: "150", unit: "g" },
    { item: "brown sugar", quantity: "3", unit: "tbsp" }
    ...
  ]
}
```

Basic Terminologies



Document

```
{
  _id: ObjectId("507f191e810c19729de860ea"),
  name: "Pad Thai",
  cuisine: "Thai",
  category: "Noodle",
  rating: 4.7,
  votes: 9301,
  ingredients: [
    { item: "noodle", quantity: "125", unit: "g" },
    { item: "chicken", quantity: "150", unit: "g" },
    { item: "brown sugar", quantity: "3", unit: "tbsp" }
    ...
  ]
}
```

MongoDB CRUD Operations - Insert

```
use("restaurants")

db.recipe.insertOne(
    {
      name: "Pad Thai",
      cuisine: "Thai",
      rating: 4.7,
      votes: 9301,
      ingredients: [
        { item: "noodle", quantity: "125", unit: "g" },
        { item: "chicken", quantity: "150", unit: "g" },
        { item: "brown sugar", quantity: "3", unit: "tbsp" }
      ]
    }
)
```

collection “recipe”

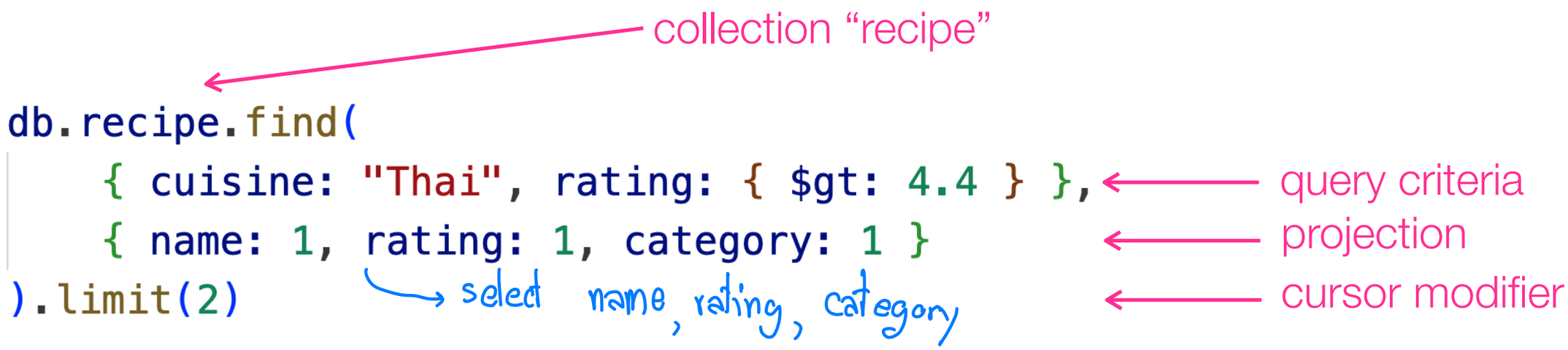
field: value

field: value

field: value

document

MongoDB CRUD Operations - Query and Others



- Other CRUD operations

  `db.collection.updateOne()`

  `db.collection.updateMany()`

  `db.collection.replaceOne()`

  `db.collection.deleteOne()`

  `db.collection.deleteMany()`

- See https://www.mongodb.com/docs/manual/crud/ for more information

More on Query Operation

Collection | Query Criteria | Modifier

```
db.recipe.find( {rating: {$gt: 4}} ).sort( {rating: -1} )
```

Aggregation Operations (sum, avg) ช่วยคำนวณ ไม่ได้ aggregate เครื่องเดียว

เหมือนกับ SQL ที่ทำบนเครื่องเดียว

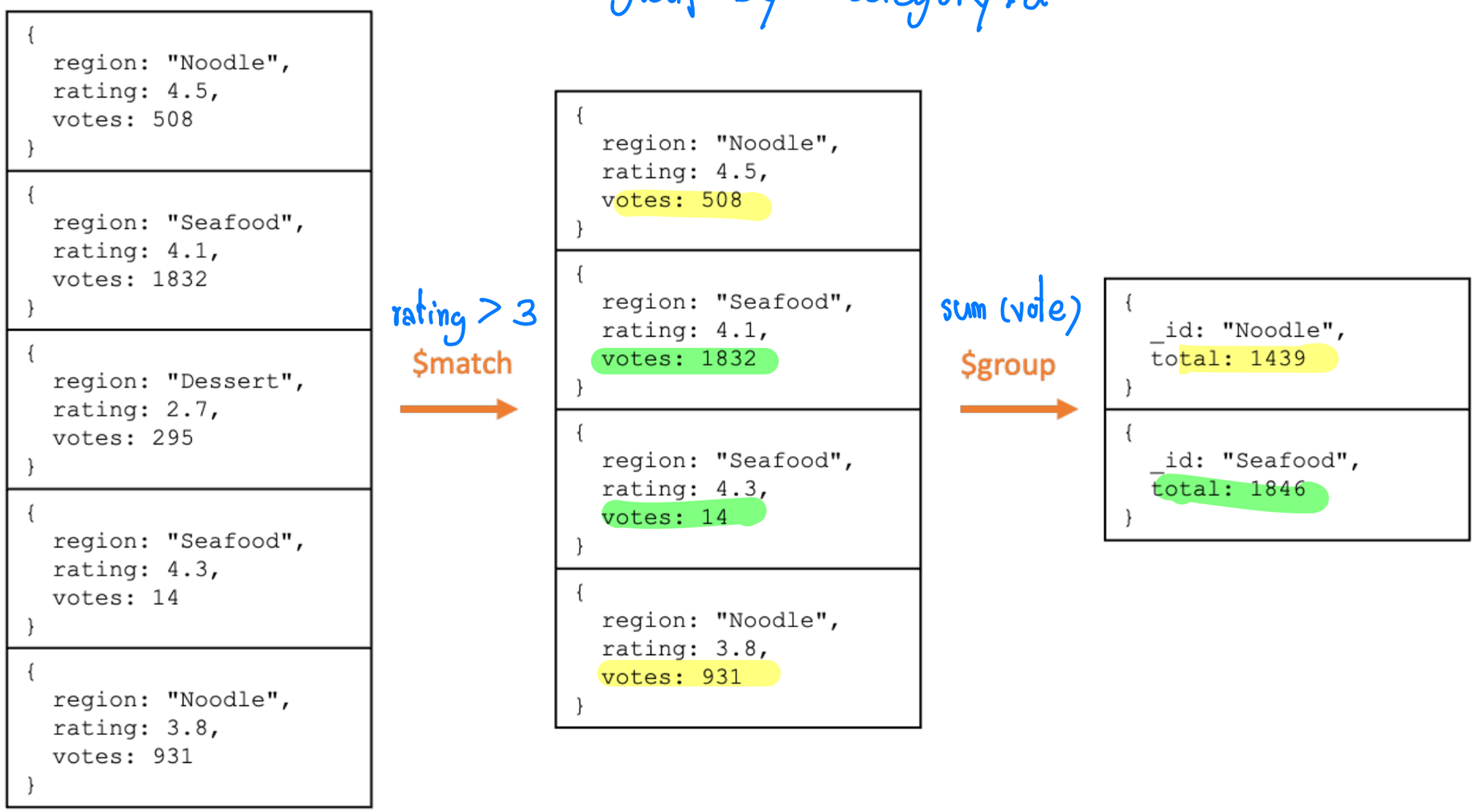
Collection

```
db.recipe.aggregate( [
  { $match: { rating: {$gt: 3.0} } },
  { $group: { _id: "$category", total: { $sum: "$votes" } } }
] )
```

$match stage → { $match: { rating: {$gt: 3.0} } },

$group stage → { $group: { _id: "$category", total: { $sum: "$votes" } } }

group by categoryId



Playing with MongoDB - Client

- For client-side, you have a couple options
  - Using MongoDB (JavaScript) with VS Code, install MongoDB for VS Code from the VS Code Marketplace
  - Using MongoDB with python, install pymongo

```
pip install pymongo
```

Playing with MongoDB - Server (container)

- Use official mongodb image at https://hub.docker.com/_/mongo

```
docker run -d --name mongo --rm -p 27017:27017 mongo
```

- You can also clone https://github.com/natawutn/datasci_architecture repo for docker compose and example files
  - To run single server (with mongo express), use docker-compose.yml in compose/single-server folder

```
// Select the database to use.
use('mongodbVSCodePlaygroundDB');

// Insert a few documents into the sales collection.
db.getCollection('sales').insertMany([
  { 'item': 'abc', 'price': 10, 'quantity': 2, 'date': new Date('2014-03-01T08:00:00Z') },
  { 'item': 'jkl', 'price': 20, 'quantity': 1, 'date': new Date('2014-03-01T09:00:00Z') },
  { 'item': 'xyz', 'price': 5, 'quantity': 10, 'date': new Date('2014-03-15T09:00:00Z') },
  { 'item': 'xyz', 'price': 5, 'quantity': 20, 'date': new Date('2014-04-04T11:21:39.736Z') },
  { 'item': 'abc', 'price': 10, 'quantity': 10, 'date': new Date('2014-04-04T21:23:13.331Z') },
  { 'item': 'def', 'price': 7.5, 'quantity': 5, 'date': new Date('2015-06-04T05:08:13Z') },
  { 'item': 'def', 'price': 7.5, 'quantity': 10, 'date': new Date('2015-09-10T08:43:00Z') },
  { 'item': 'abc', 'price': 10, 'quantity': 5, 'date': new Date('2016-02-06T20:20:13Z') },
]);

// Run a find command to view items sold on April 4th, 2014.
const salesOnApril4th = db.getCollection('sales').find({
  date: { $gte: new Date('2014-04-04'), $lt: new Date('2014-04-05') }
}).count(); >=                          <

// Print a message to the output window.
console.log(`${salesOnApril4th} sales occurred in 2014.`);

// Here we run an aggregation and open a cursor to the results.
// Use '.toArray()' to exhaust the cursor to return the whole result set.
// You can use '.hasNext()/.next()' to iterate through the cursor page by page.
db.getCollection('sales').aggregate([
  // Find all of the sales that occurred in 2014.
  { $match: { date: { $gte: new Date('2014-01-01'), $lt: new Date('2015-01-01') } } },
  // Group the total sales for each product.
  { $group: { _id: '$item', totalSaleAmount: { $sum: { $multiply: [ '$price', '$quantity' ] } } } }
]);
```

Python and MongoDB Introduction Notebook

This notebook bases on a mongodb tutorial blog "Getting Started with Python and MongoDB" available at https://www.mongodb.com/blog/post/getting-started-with-python-and-mongodb

In [ ]:
```
from pymongo import MongoClient
from bson.objectid import ObjectId
from random import randint
from pprint import pprint

# Connect to MongoDB, change the << MONGODB URL >> to reflect your own connection string
client = MongoClient()
# Connect to collection - Note: Change connection string as needed
db = client.restaurants
```

Basic operations: insert, query, and delete

In [ ]:
```
doc = { 'name': 'Juicy Burger', 'rating': 4, 'cuisine': 'American' }
result = db.reviews.insert_one(doc)
oid = result.inserted_id
print(oid)
```

In [ ]:
```
for doc in db.reviews.find():
    print(doc)
```

In [ ]:
```
db.reviews.delete_one({'_id': ObjectId(oid)})
```

In [ ]:
```
for doc in db.reviews.find():
    print(doc)
```

Generate sample data

In [ ]:
```
#Step 2: Create sample data
names = ['Kitchen','Animal','State', 'Tastey', 'Big','City','Fish', 'Pizza','Goat', 'Salty','Sandwich','Lazy', 'Fun']
company_type = ['LLC','Inc','Company','Corporation']
company_cuisine = ['Pizza', 'Bar Food', 'Fast Food', 'Italian', 'Mexican', 'American', 'Sushi Bar', 'Vegetarian']
for x in range(1, 501):
    restaurant = {
        'name' : names[randint(0, (len(names)-1))] + ' ' + names[randint(0, (len(names)-1))]  + ' ' + company_type[randi
        'rating' : randint(1, 5),
        'cuisine' : company_cuisine[randint(0, (len(company_cuisine)-1))]
    }
    #Step 3: Insert business object directly into MongoDB via isnert_one
    result = db.reviews.insert_one(restaurant)
    #Step 4: Print to the console the ObjectID of the new document
    print('Created {0} of 500 as {1}'.format(x,result.inserted_id))
```

Highly Available with Replica Set

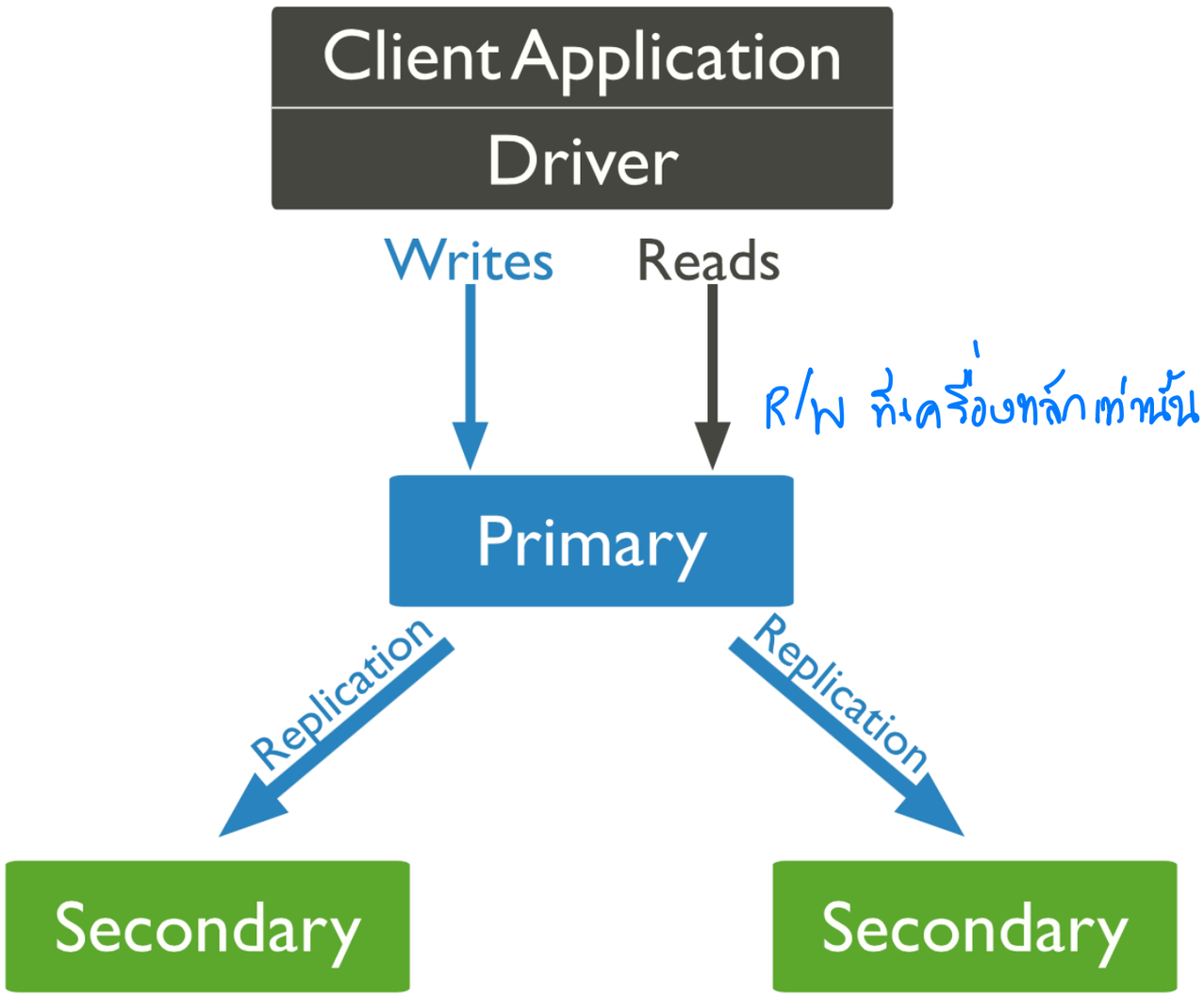
HA

set ของ server instance

replica set = 3

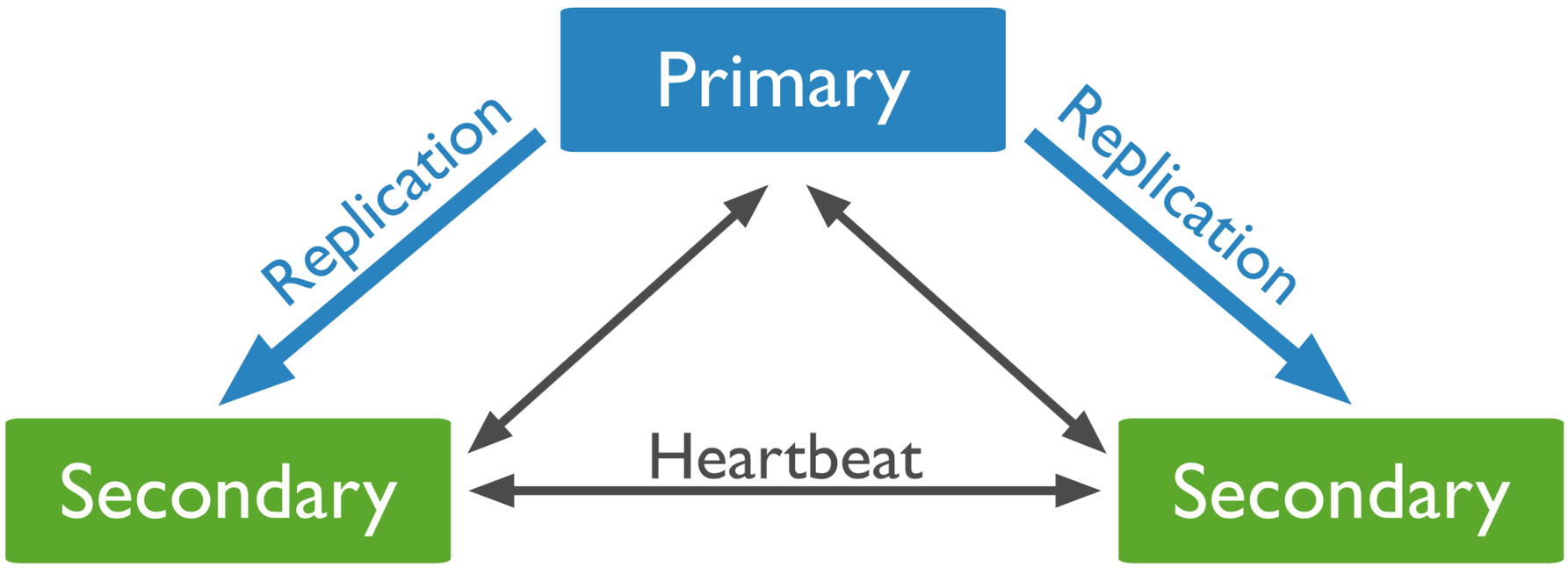
คือ 1 คือ primary 2 คือ secondary

- A group of mongod processes that maintain the same data set
- Provide redundancy and high availability
- Provide increased read capacity as clients can send read operations to different servers (in some cases)



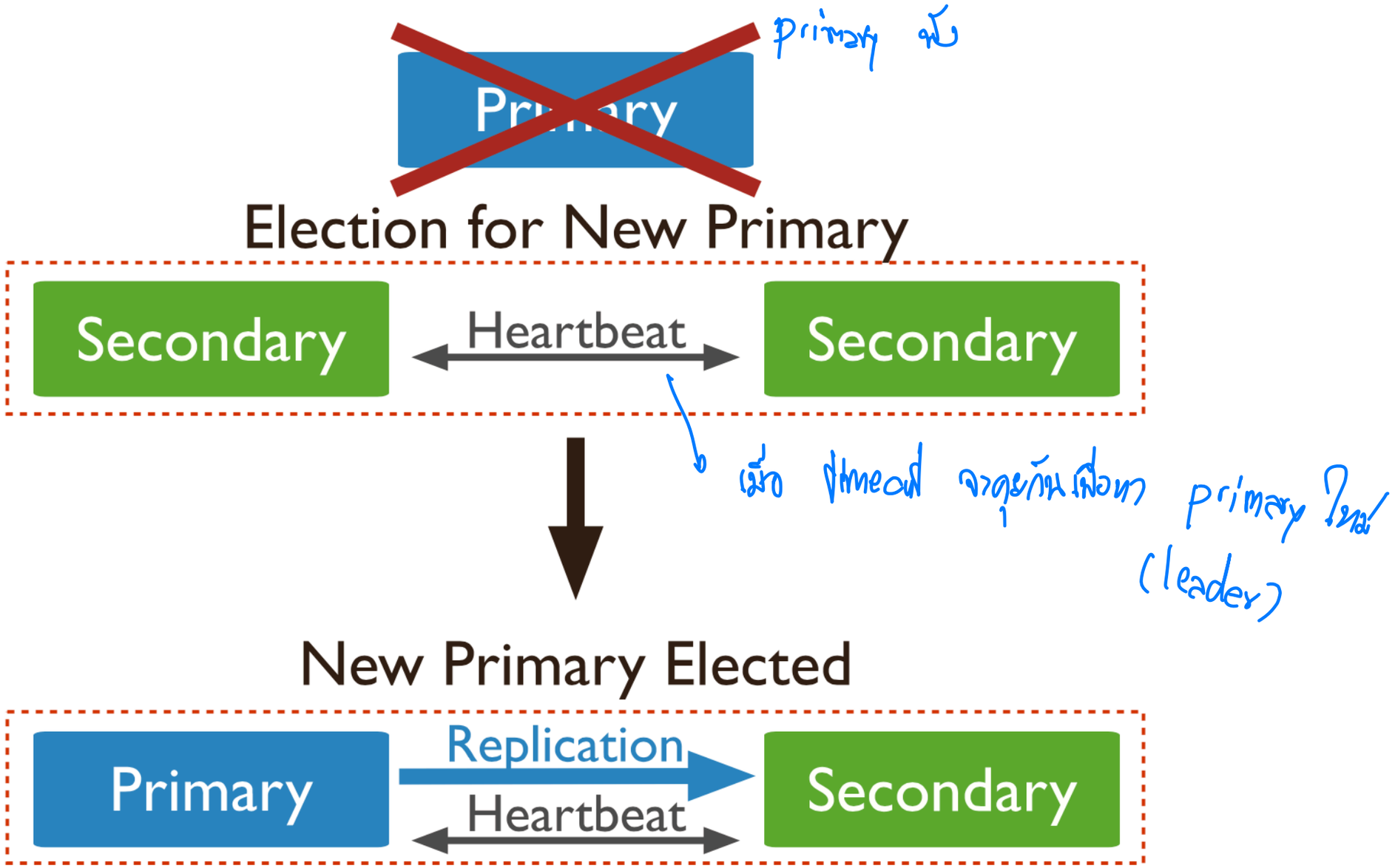
Source: https://docs.mongodb.com/manual/replication/

Replica Set in Actions



Source: https://docs.mongodb.com/manual/replication/

Replica Set in Actions



Source: https://docs.mongodb.com/manual/replication/

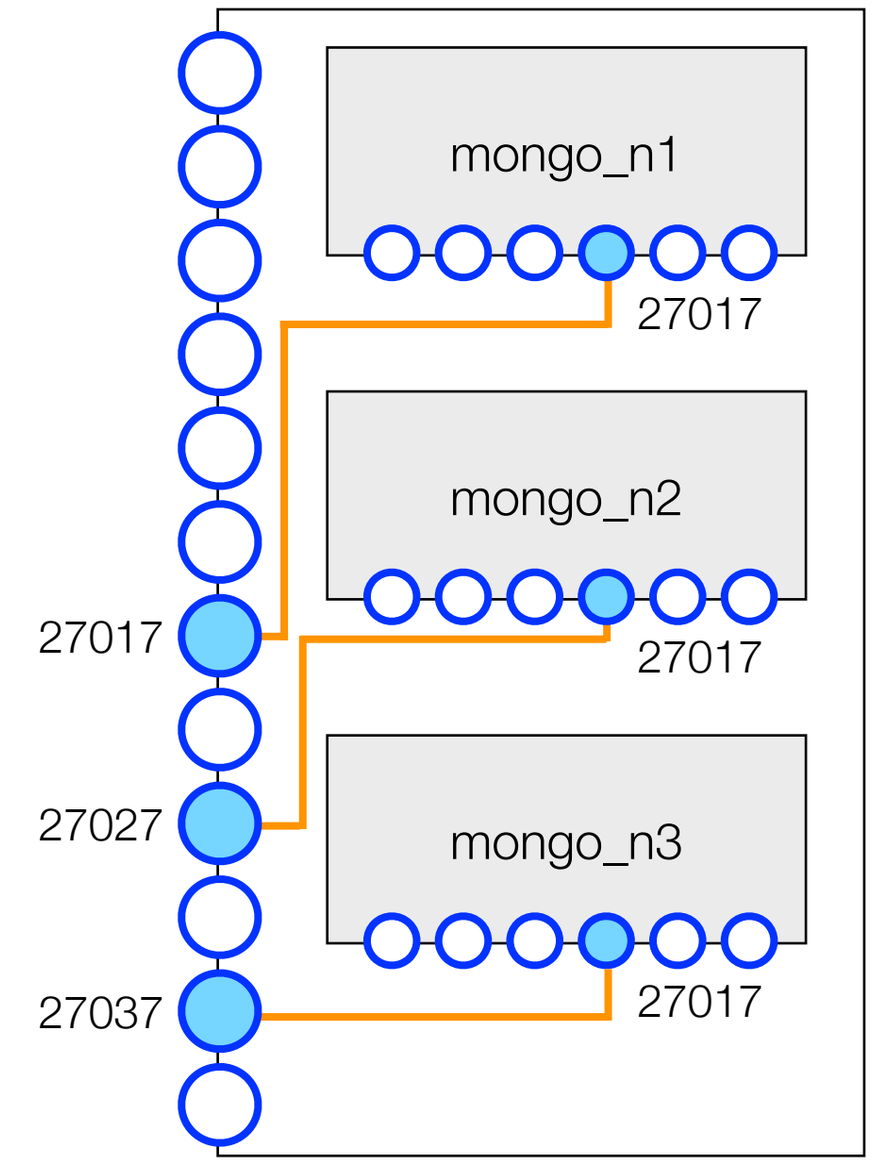
Playing with MongoDB Replica Set

- Start all containers

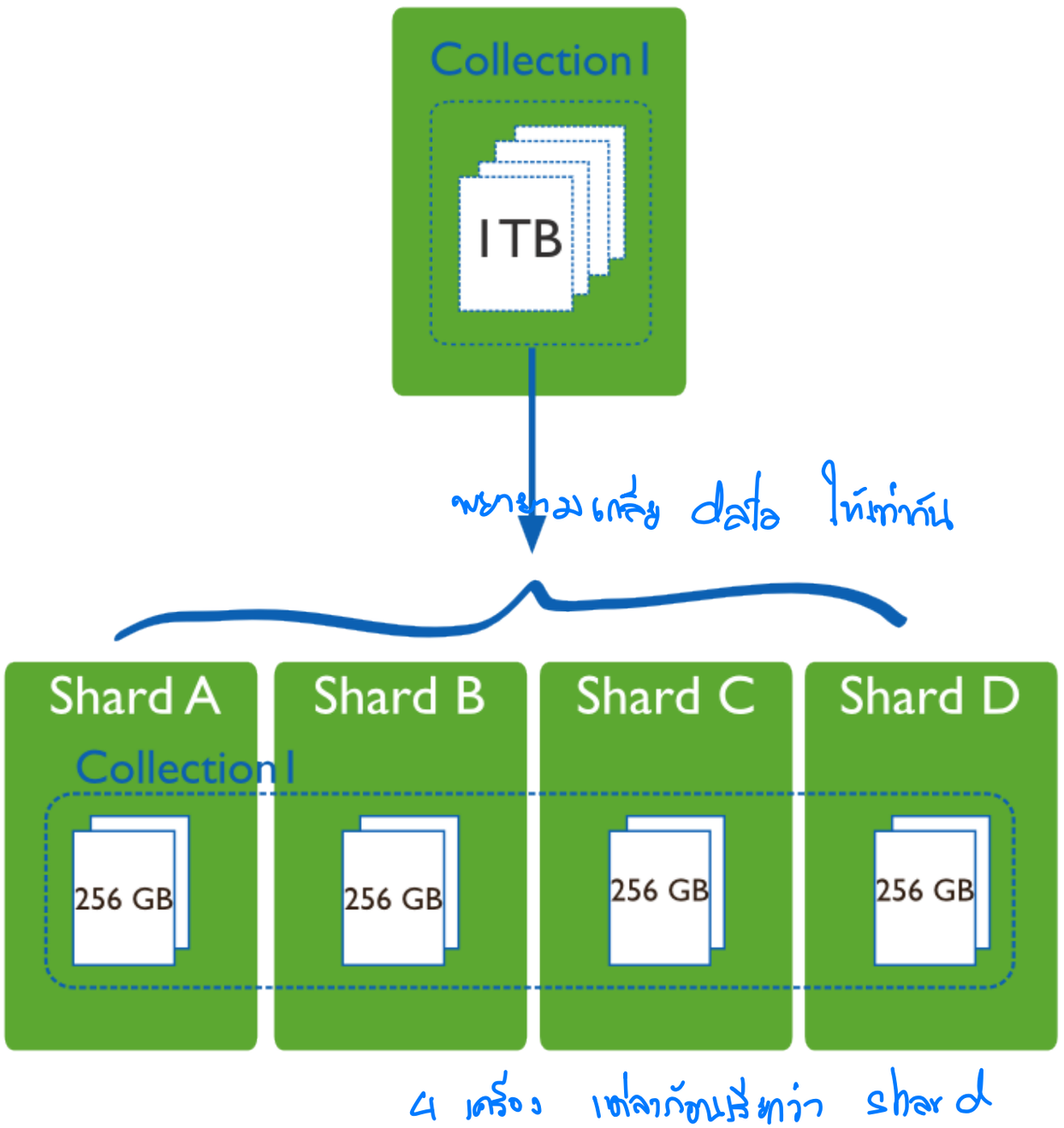
  docker-compose up

- Initialize the replica set (inform just one node)

```
docker exec -it mongo_n1 bash -c
"echo 'rs.initiate({_id : \"rs1\",
members: [{ _id : 0, host :
\"mongo_n1\" },{ _id : 1, host :
\"mongo_n2\" },{ _id : 2, host :
\"mongo_n3\" }]})' | mongo"
```



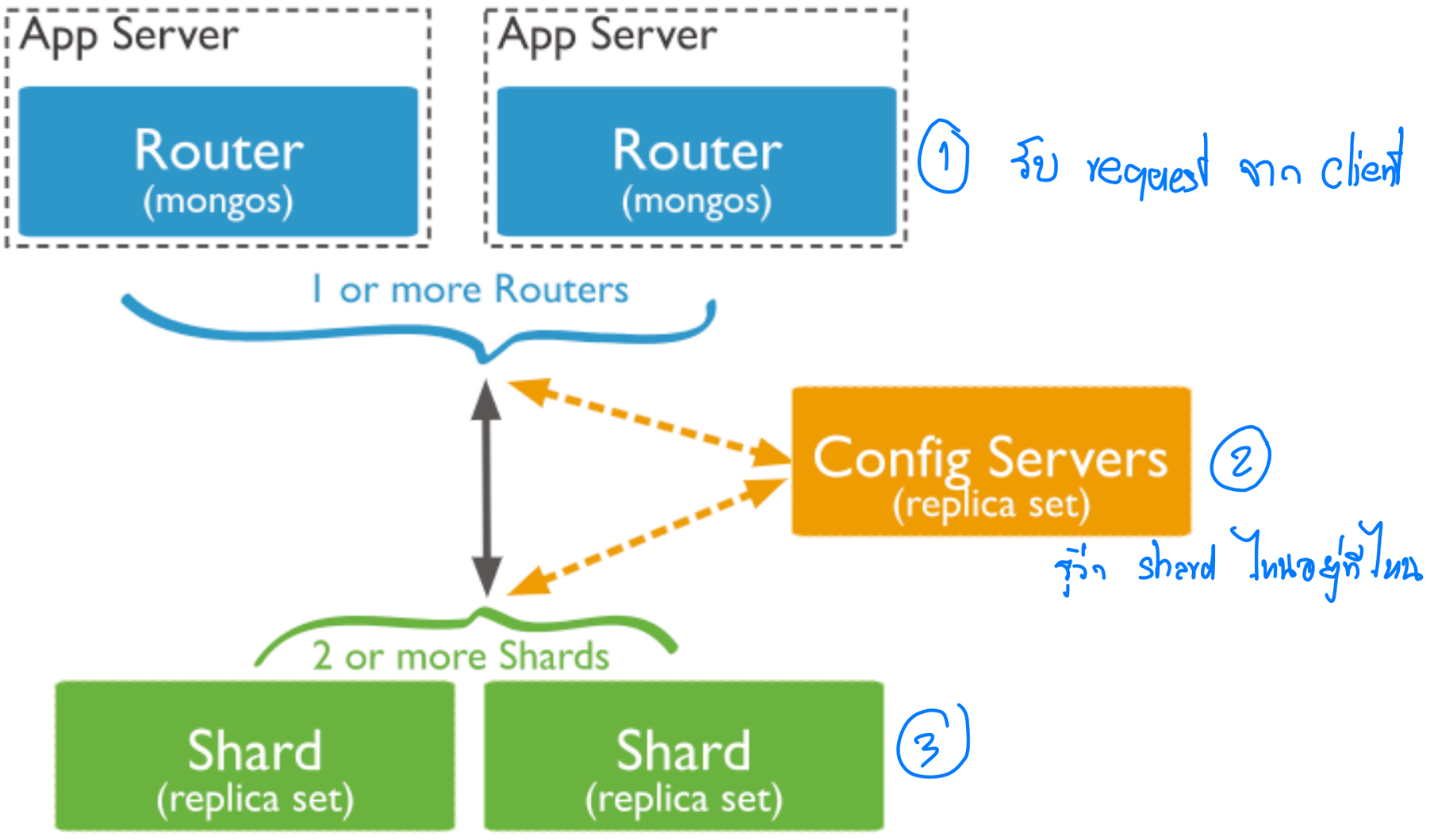
Scalable with Sharding



- Large-Scale MongoDB bases on Horizontal Scaling mechanism
- Storing shards (blocks of data) across multiple machines
- Data partitioned into shards with shard keys
- Each shard handles only operations related to its block

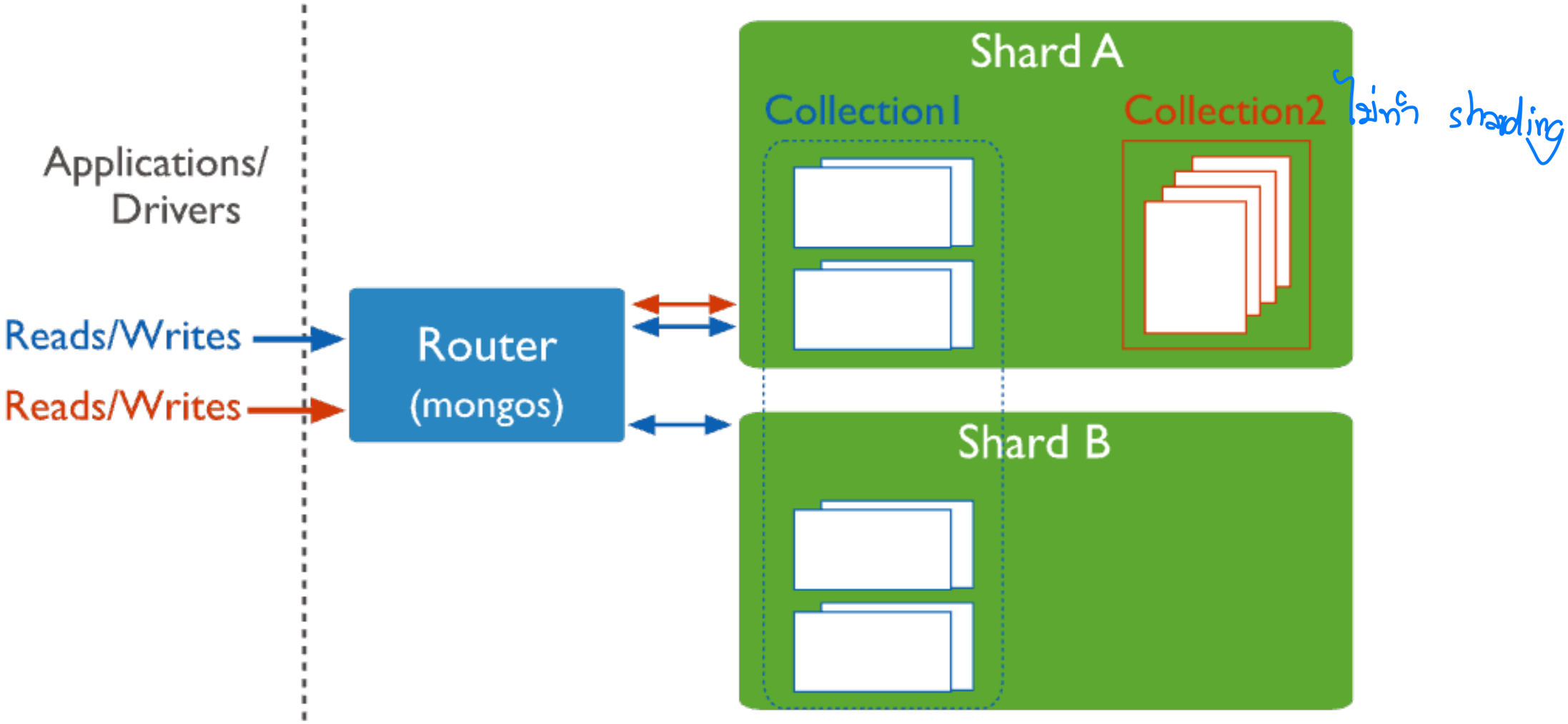
Source: https://docs.mongodb.org/manual/core/sharding-introduction

Shard Cluster Deployment



Source: https://docs.mongodb.com/manual/core/sharded-cluster-components/

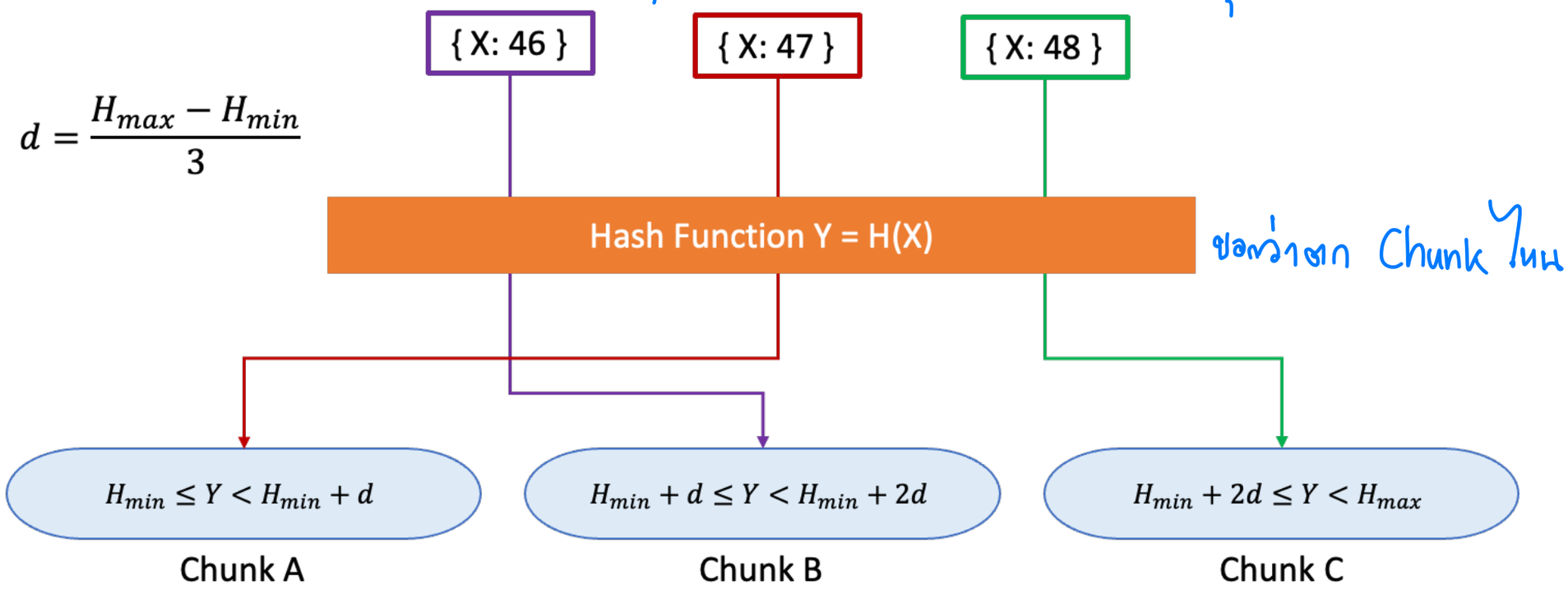
MongoDB Sharding



Shard Key Concepts: Hash Shading

ข้อดี: ทำให้ไม่ load เครื่องใดเครื่องหนึ่งเกินไป

ข้อเสีย : ถ้า query เป็นช่วง (0-50) ต้องถามทุกเครื่อง



ของว่างตก Chunk ไหน

ดีตรง random access ตรงที่ แม้ว่าข้อมูลช่วงติดกัน

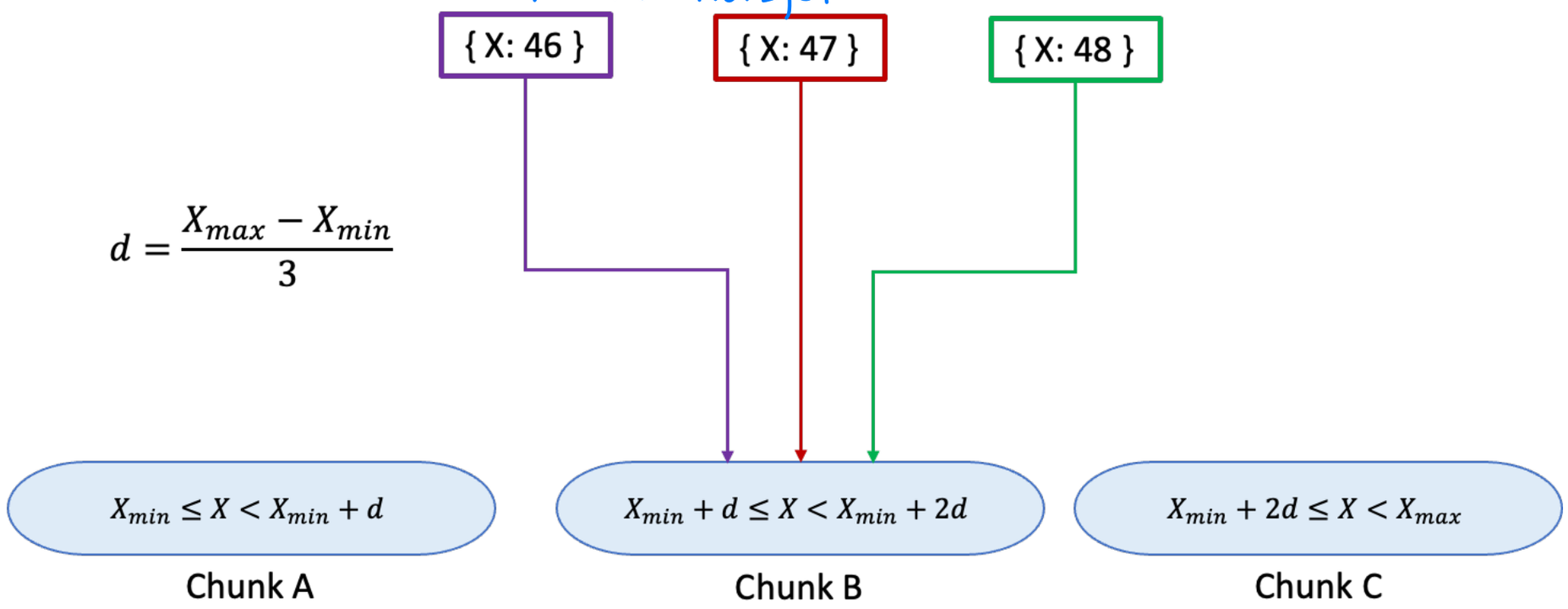
ไม่มี hotspot ด R/W เพราะกระจายเครื่อง)

Shard Key Concepts: Range Shading

ระบุช่วง Chunk

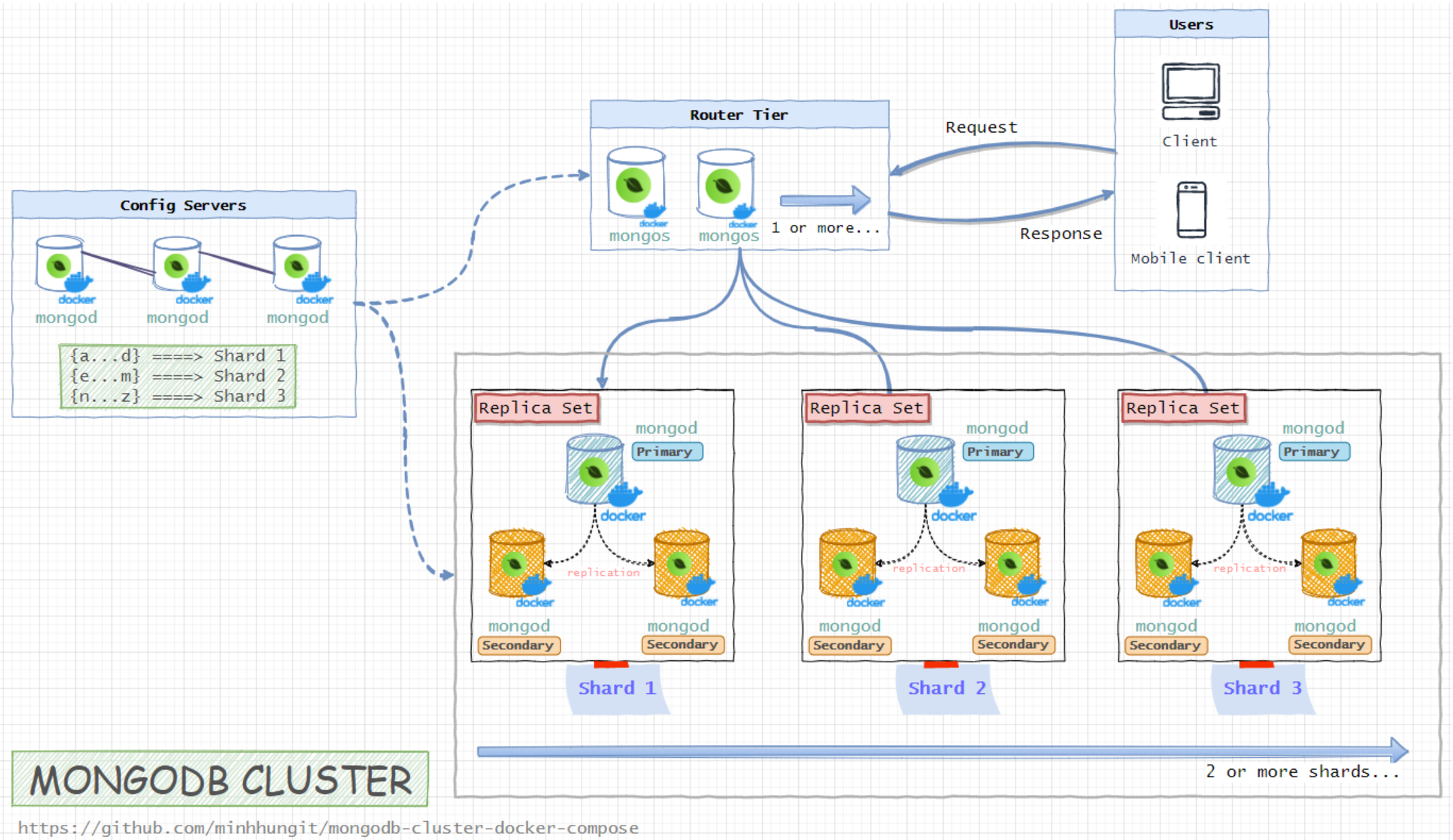
ข้อดี : ค้นหาเป็นช่วงเดียว

เสีย : เกิด hotspot



Playing with Shards

- Refer to https://github.com/minhhungit/mongodb-cluster-docker-compose



DynamoDB

Mongo ต้นไอดอลแทน RDBMS

Dynamo เน้น performance บน cloud service

Document Store
Amazon DynamoDB

Amazon DynamoDB

- Managed NoSQL Database Services on Amazon Web Services (also available in On-Premise)
- Document-oriented and key-value models
- (Somewhat) flexible schema - Table / Items / Attributes
- Highly available with 3x replication
- Trade off scalability with relax data consistency (AP NoSQL)
- Very fast, milliseconds latency at any scale
- Utilize Solid State Drive
- Push consistency toward read operations
- Support several language bindings:
  Java, Node.js, C# .NET, Perl, PHP, Python, Ruby, Haskell and Erlang

RDBMS vs. DynamoDB

RDBMS	~~MongoDB~~ DynamoDB
Database	Database
Table	Table
Row / Tuple	Item
Column	Attribute
Table Join	Embedded Item
Primary Key	Partition Key + Sort Key

Dynamo ทำเหมือน document store ยังไง

DynamoDB Data Model

มีไว้กระจาย partition (shard) ผ่าน hash

ใช้ sort data ที่เก็บในแต่ละ partition เพื่อ search ใน node/partition นั้น

Partition Key Sort Key Partition Key + Sort Key = must unique Table

เพื่อ search โดย binary search ใน node นั้น

คล้ายๆ กับทำ indexing

Item

Partition Key 1	Sort Key 1	attribute1 = value	attribute2 = value	attribute4 = value	
Partition Key 1	Sort Key 2	attribute1 = value			
Partition Key 2	Sort Key 5	attribute2 = value	attribute3 = value	attribute4 = value	attribute8 = value
Partition Key 3	Sort Key 5	attribute1 = value	attribute5 = value		

ถ้า query รวมๆ ทั้งหมดด้วย category แต่เรา by name ก็จบทุกอย่าง

Recipe

category	name	cuisine	rating	votes	(list of) ingredients

ขึ้นกับ design ตาม query

Recipe

name	category	cuisine	rating	votes	(list of) ingredients

DynamoDB API

CreateTable

UpdateTable

DeleteTable

DescribeTable

ListTables

GetItem

Query

Scan

BatchGetItem

PutItem

UpdateItem

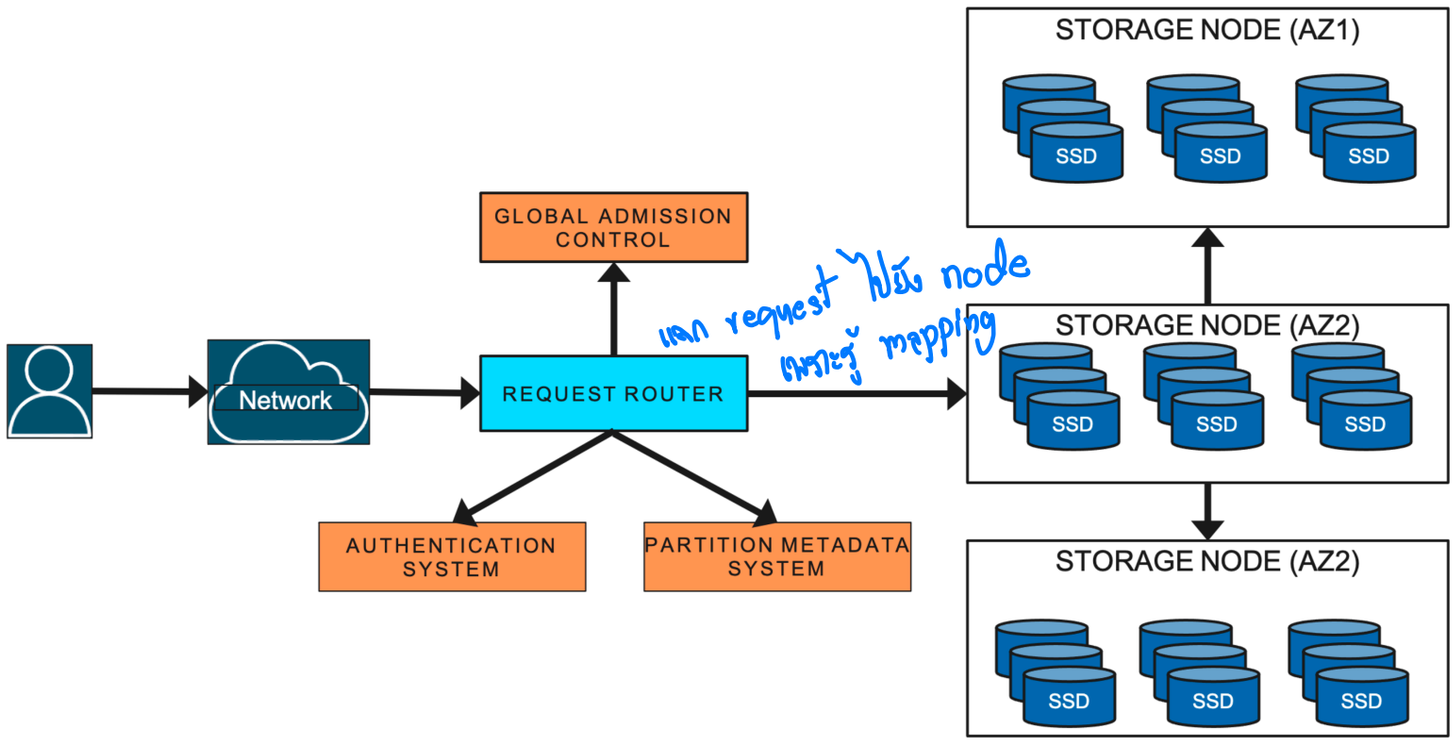
DeleteItem

BatchWriteItem

Data Types for Attribute

- String (S)
- Number (N)
- Binary (B)
- String Set (SS)
- Number Set (NS)
- Binary Set (BS)
- Boolean (BOOL)

DynamoDB Architecture



- Items are distributed among partitions resided in storage nodes using hashed Partition Key (aka. Hash Key)
- At each storage node, items are stored in B-Tree structure stored by Sort Key

DynamoDB Partition

- Partitions host disjoint and contiguous parts of the table's key-range
- Each partition can store up to 10GB with 3000 RCUs and 1000 WCUs; partition is splitter if exceeding these numbers
- Each partition has multiple replicas distributed across different Availability Zones for high availability and durability

ทั้ง copy ได้

- The replicas for a partition form a replication group and one replica in the group is elected as the leader (or primary)
  - With write request, leader replica generates a write-ahead log record and sends it to its peer (replicas) and returns a successful response to the client when one of the replicas responds with a successful write — eventually consistent write (W=2)
  - With read request, client can choose either strongly consistent read (R=N-1) and eventually consistent read (R=1)

Playing with Amazon DynamoDB Locally

- Amazon provides DynamoDB docker container to try locally

```
docker run -d -p 8000:8000 --name dynamo --rm amazon/
dynamodb-local
```

- You can access local DynamoDB services using aws CLI or Python with boto3 package

- Note: you will need aws access credential including AWS Access Key ID and AWS Secret Access Key
  - create them from AWS console and config to your machine with aws configure command

Why Document Oriented NoSQL?

- "Natural-fit" to lots of information
  - Parent-child objects (e.g. story-chapters-comments)
- Schema changed over time
- Expecting high write load
- Horizontal scaling is needed in the future (e.g. data will grow very large)
- Data backup/restore with point-in-time approach

Assignment Readings

- M. Elhemali, et al. "Amazon DynamoDB: A Scalable, Predictably Performant, and Fully Managed NoSQL Database Service”, 2022 USENIX Annual Technical Conference (USENIX ATC 22). 2022.

References

- Nic Raboy, “Getting Started With MongoDB As A Docker Container Deployment”, https://www.thepolyglotdeveloper.com/2019/01/getting-started-mongodb-docker-container-deployment/

- อภิศิลป์ ตรุงกานนท์, “ทำไม Pantip ปิดปรับปรุงบ่อย บทเรียน ความเจ็บปวด และคราบน้ำตาจาก MongoDB”, http://macroart.net/2013/10/mongodb-lessons-learned-on-pantip/ เกิดจากการออกแบบ Sharding ผิด

- T. Singthong, “แนวทางการสร้าง Index และ การเลือก Sharding strategy ใน MongoDB”, https://medium.com/iamgoangle/แนวทางการสร้าง-index-และ-การเลือก-sharding-strategy-ใน-mongodb-d370408df0f6

- T. Gamage, “Running AWS DynamoDB Local with Docker-Compose”, https://medium.com/platform-engineer/running-aws-dynamodb-local-with-docker-compose-6f75850aba1e