優化 Kafka 的最後一哩路

從客戶端到伺服器端的負載平衡解決方案

蔡嘉平
2022/07/26
www.chia7712.tw

蔡嘉平
www.chia7712.tw

開源專案：Astraea
https://github.com/skiptests/astraea

① 總主持人 @ 三個Kafka系統開發案
② 顧問 @ 原昌、亦思、教育部人培
③ Committer | PMC @ Kafka、HBase、Yunikorn
④ Member @ Apache
⑤ PC Member @ Kafka Summit 2022

① Kafka 負載平衡工具
② 更現代化的Kafka APIs
③ Web Service over Kafka APIs
④ 效能評測工具

Powered by 原昌工業、成功大學、亦思科技、科學園區、其他公司的貢獻者

1. 什麼是 Apache Kafka

3. 如何處理負載不平衡

2. 什麼是 Kafka 負載不平衡

4. 如何聰明的處理負載不平衡

分散式系統的負載平衡

總是有薪水小偷

實際的速度

預期的速度

引入新系統

小叢集/蜜月期

中叢集/撞牆期

大叢集/離職期

What is Kafka ?

出生地

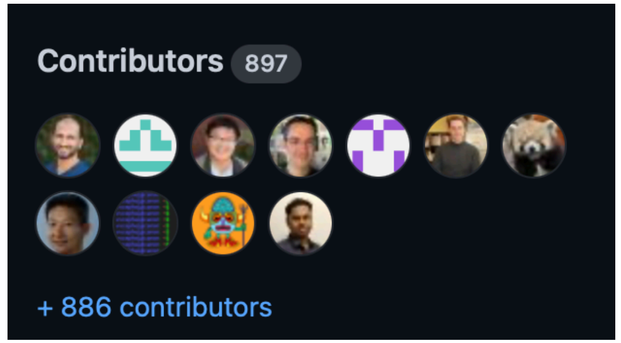
高擴充、高吞吐、低延遲
事件串流處理系統

定時舉辦國際演討會

支援各種語言 (官方 & 非官方)

C	Clojure	C-sharp/.Net	Go
Groovy	Java	Scala	Kotlin
Node.js	Python	Ruby	Rust

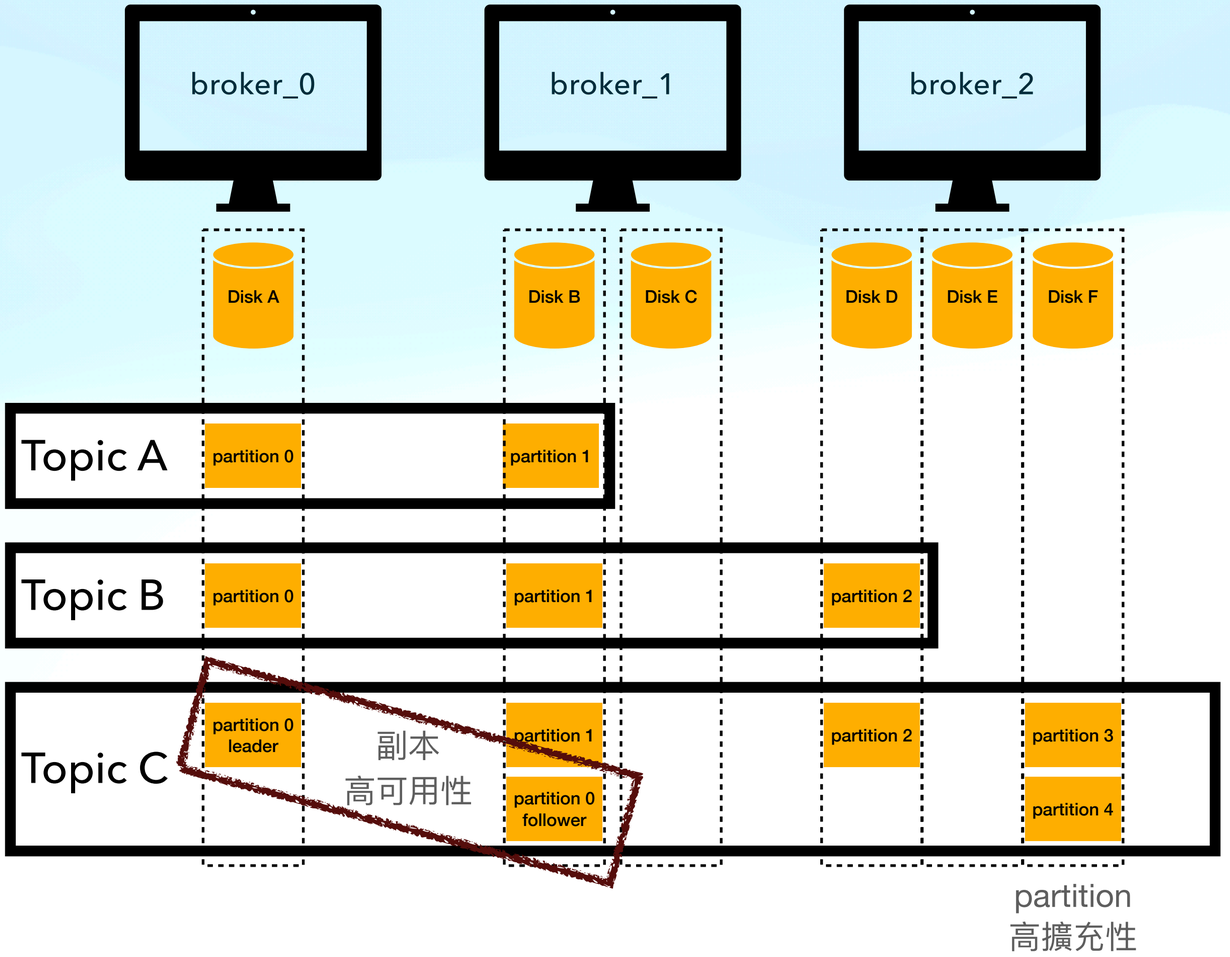
大量的貢獻者



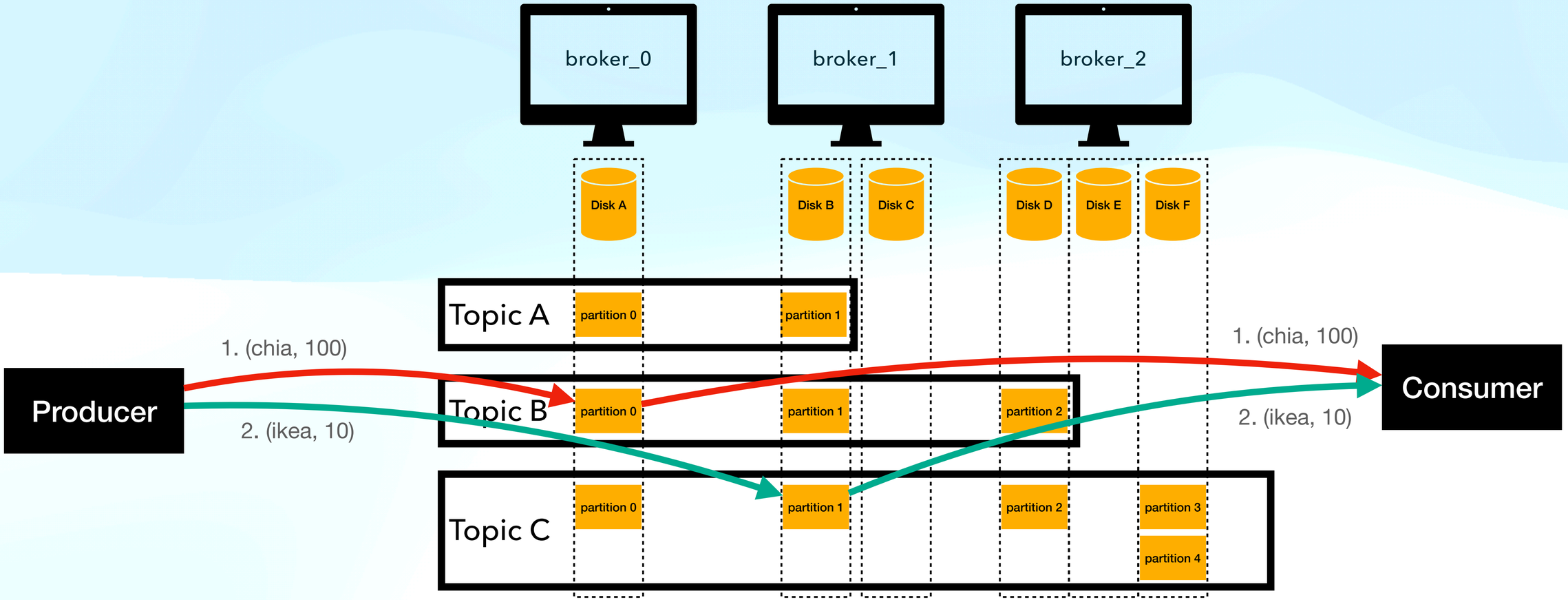
提供商業服務

CONFLUENT

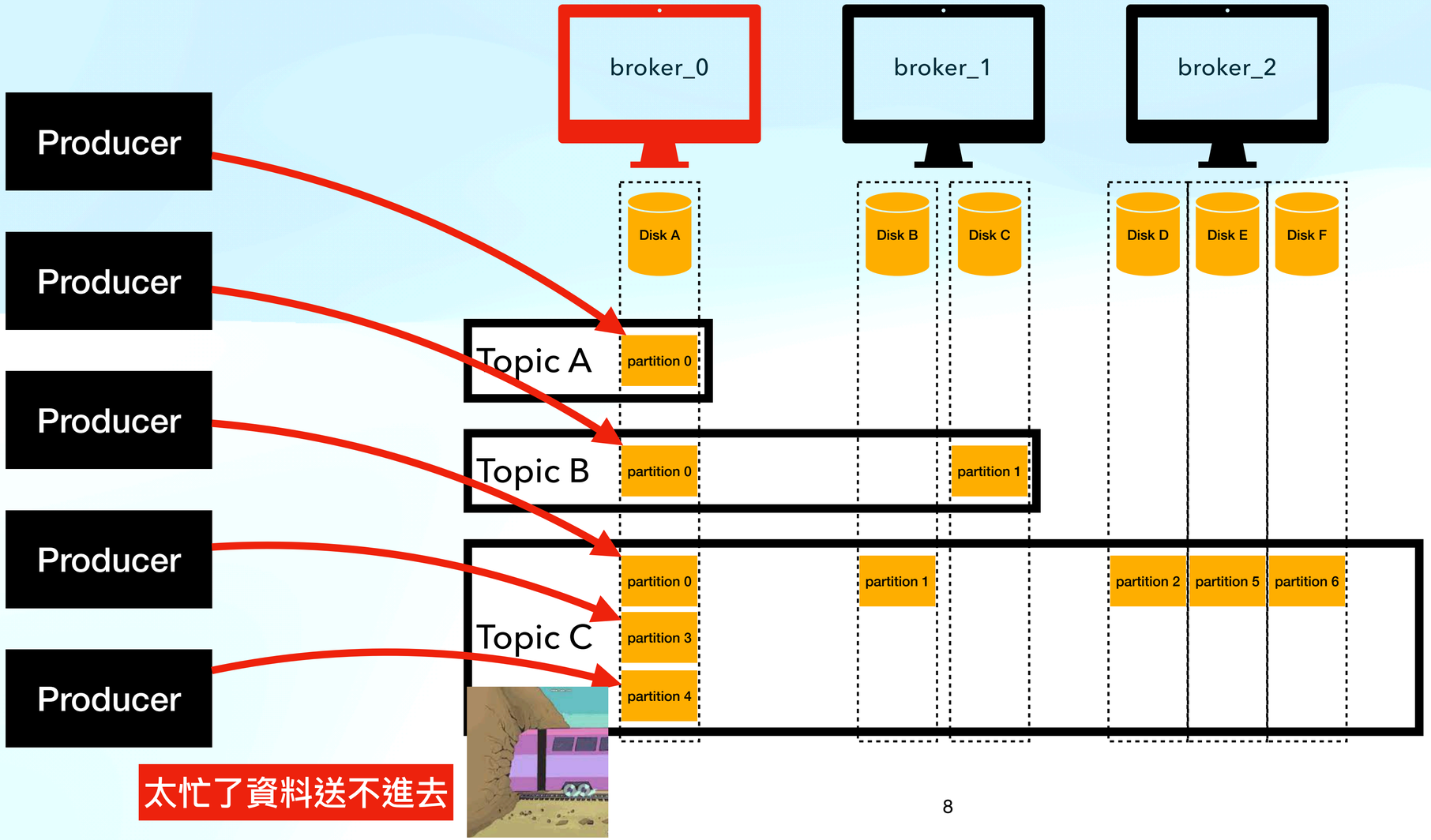
Topic, Partition and Replica

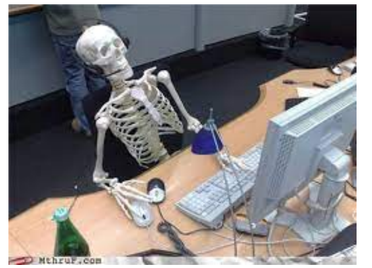


Producer and Consumer



Unbalanced

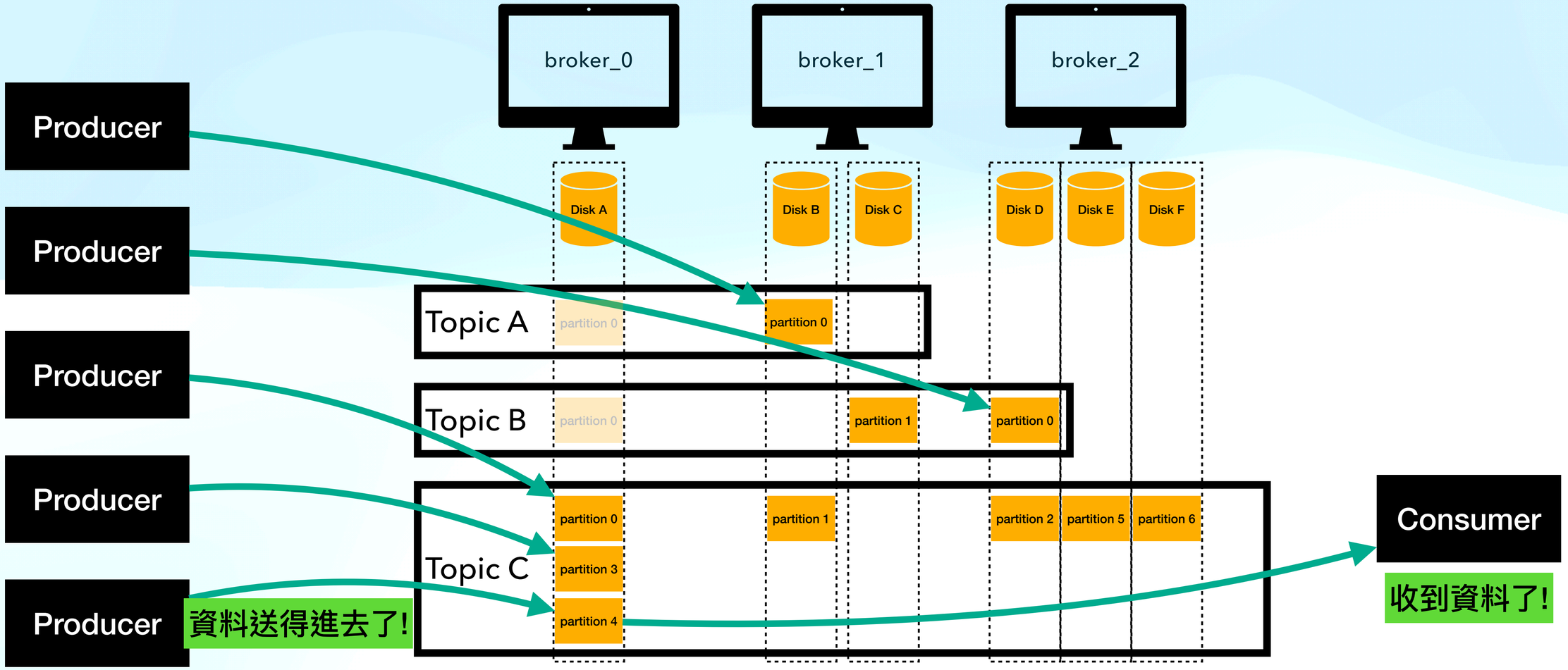




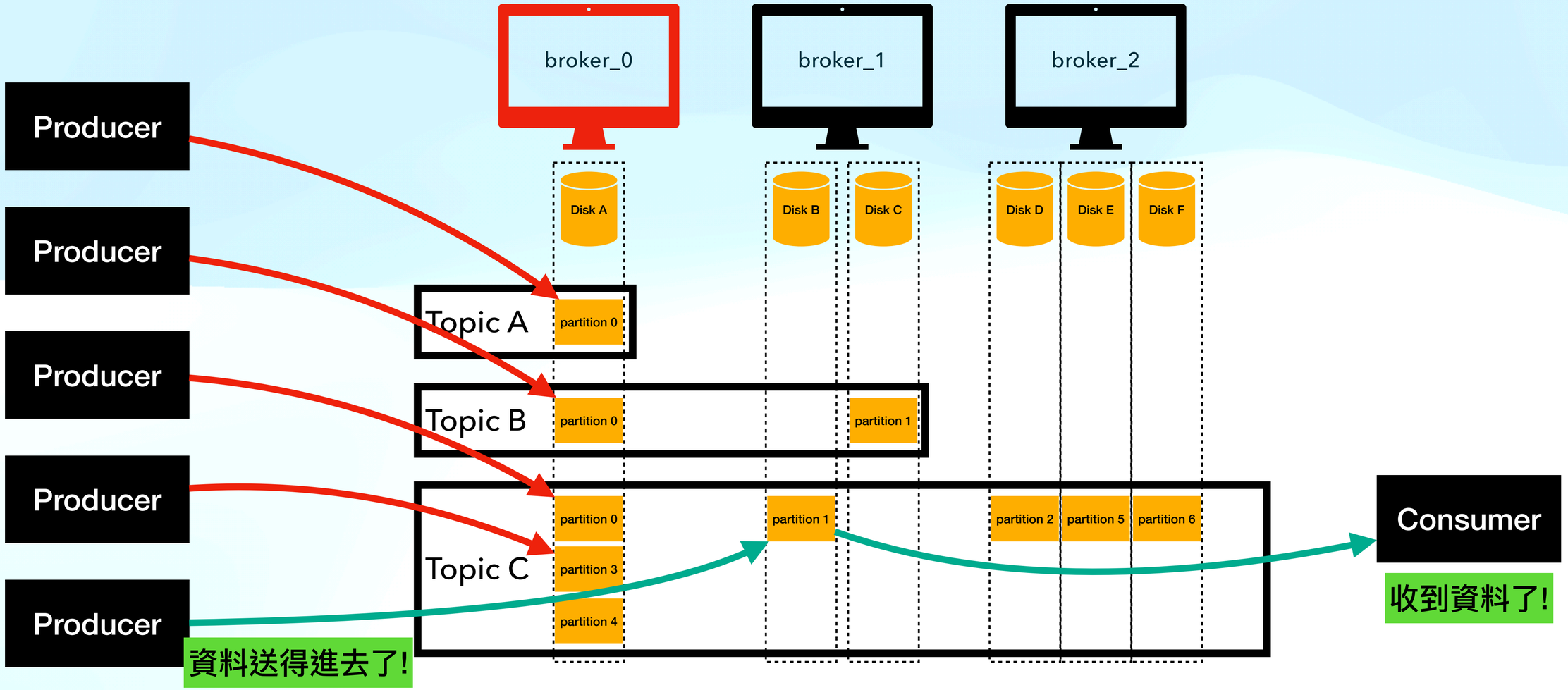
Consumer

等資料到天荒地老

Solution 1: Balance the Cluster



Solution 2: Write to Free Node



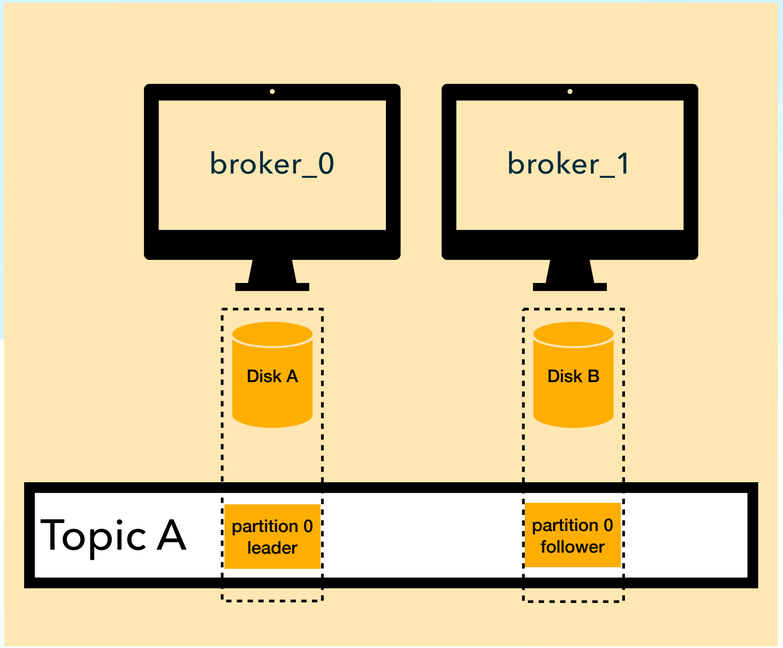
哪個比較好？

~~哪個比較好？~~

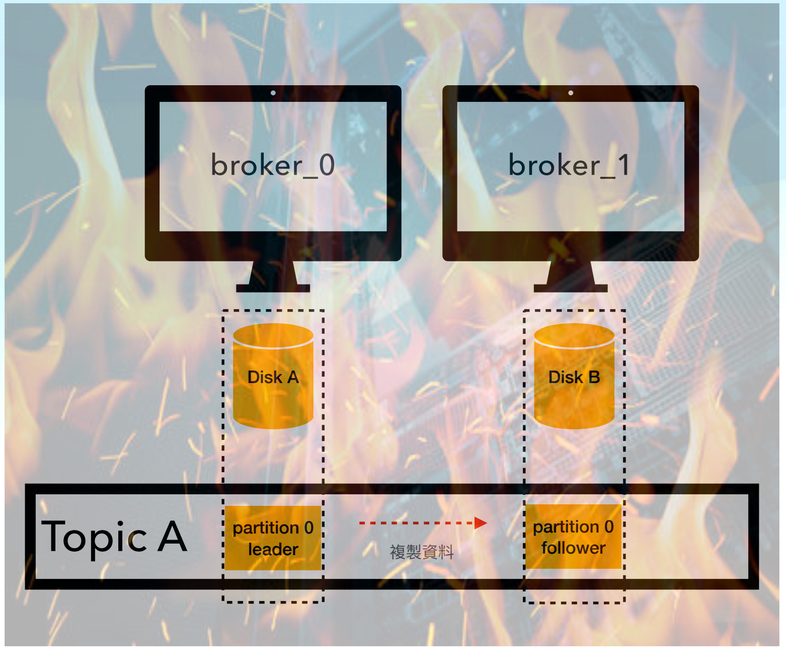
哪個比較適合?

重新配置 partitions

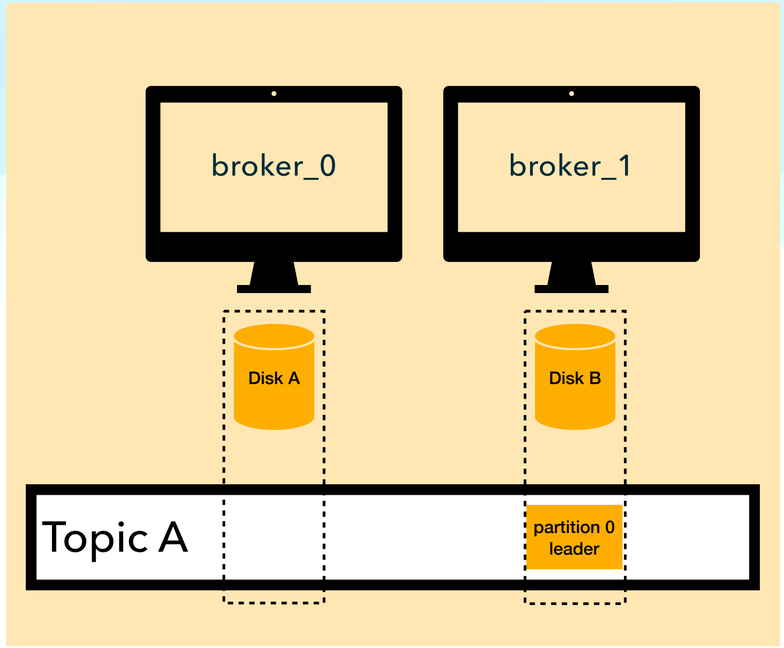
1. 新增一個Replica



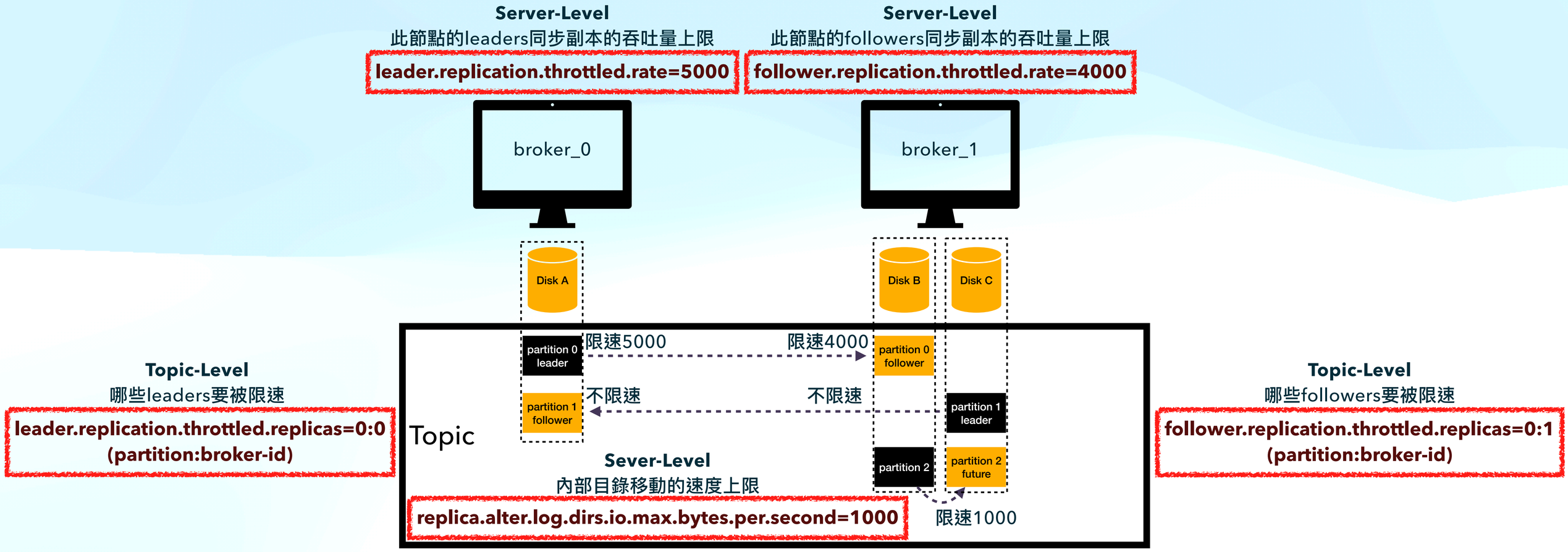
2. 等待資料同步完成



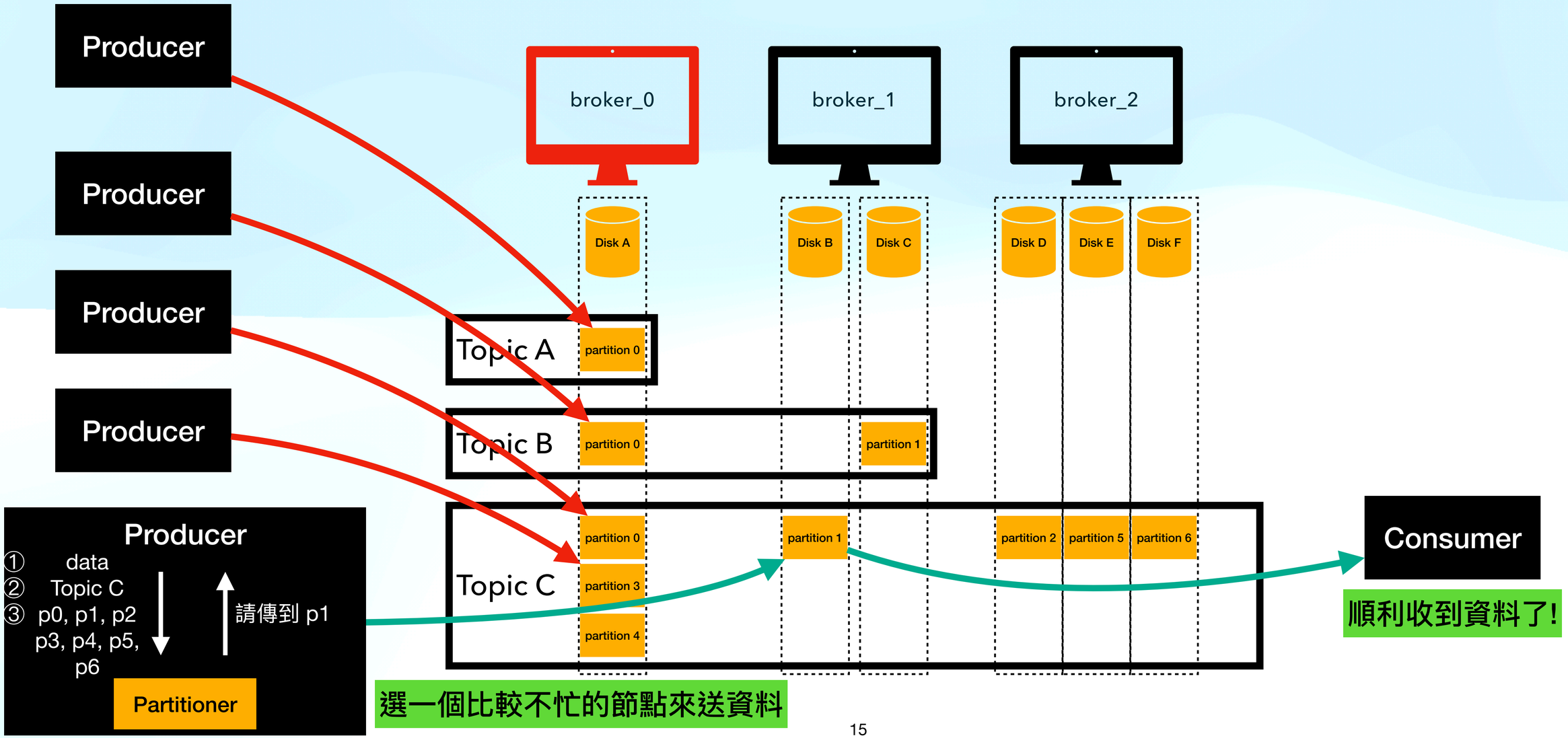
3. 移除舊有的replica並重選leader



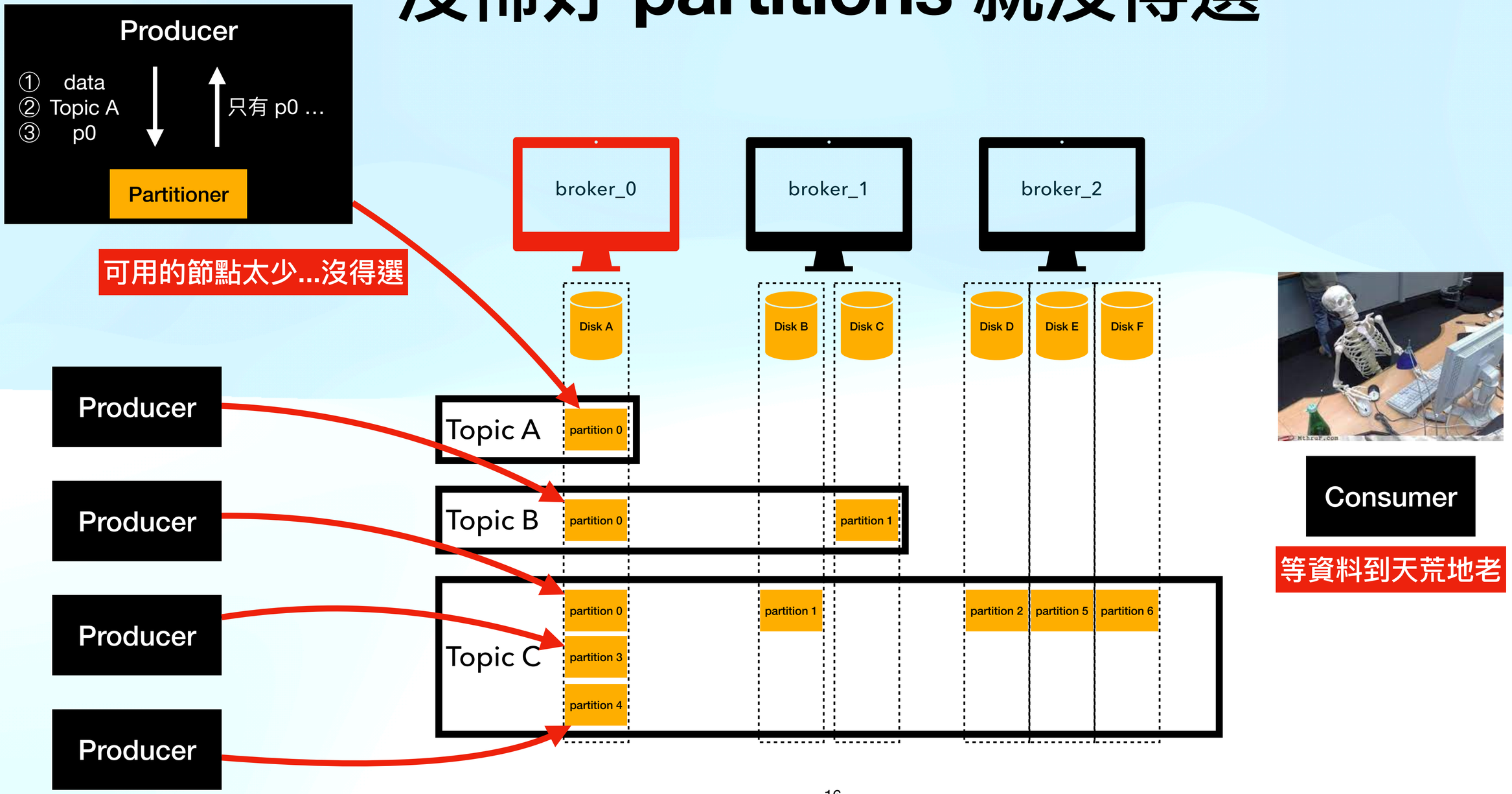
如何避免機房燒掉？


Server-Level
此節點的leaders同步副本的吞吐量上限
leader.replication.throttled.rate=5000
Server-Level
此節點的followers同步副本的吞吐量上限
follower.replication.throttled.rate=4000
broker_0
broker_1
Disk A
Disk B
Disk C
partition 0 leader
限速5000
限速4000
partition 0 follower
partition 1 follower
不限速
不限速
partition 1 leader
Topic
Sever-Level
內部目錄移動的速度上限
replica.alter.log.dirs.io.max.bytes.per.second=1000
partition 2
partition 2 future
限速1000
Topic-Level
哪些leaders要被限速
leader.replication.throttled.replicas=0:0
(partition:broker-id)
Topic-Level
哪些followers要被限速
follower.replication.throttled.replicas=0:1
(partition:broker-id)

自定義 Partitioner 做更好的選擇



沒佈好 partitions 就沒得選



Partition Reassignment v.s Partitioner 哪個比較適合?

想要更全面的平衡？還是在意平衡的成本？

心動不如手動

1. 將 replicas 部署到不同節點

```
admin.alterPartitionReassignments(
    Map.of(
        new TopicPartition("Topic A", 0),
        Optional.of(new NewPartitionReassignment(List.of(1)))));
```

指定要移動的 partitions

指定要移動到哪些節點

2. 將 replicas 部署到不同路徑 (硬碟)

```
adminClient.alterReplicaLogDirs(
  Map.of(new TopicPartitionReplica("Topic A", 0, 10),
  “/tmp/aa"));
```

指定要移動的 replicas

指定要移動到哪個目錄下

3. 自行實作 Partitioner

```
static class MyPartitioner implements Partitioner {

  @Override
  public int partition(
      String topic,
      Object key,
      byte[] keyBytes,
      Object value,
      byte[] valueBytes,
      Cluster cluster) {
    return 0;
  }

  @Override
  public void close() {}

  @Override
  public void configure(Map<String, ?> configs) {}
}
```

根據這些資訊選擇適當的 partition

手動不如自動

1. 伺服器端自動化負載平衡

Cruise Control for Apache Kafka

PASSED

Introduction

Cruise Control is a product that helps run Apache Kafka clusters at large scale. Due to the popularity of Apache Kafka, many companies have bigger and bigger Kafka clusters. At LinkedIn, we have ~7K+ Kafka brokers, which means broker deaths are an almost daily occurrence and balancing the workload of Kafka also becomes a big overhead.

Kafka Cruise Control is designed to address this operation scalability issue.

Features

Kafka Cruise Control provides the following features out of the box:

- Resource utilization tracking for brokers, topics, and partitions.
- Query the current Kafka cluster state to see the online and offline partitions, in-sync and out-of-sync replicas, replicas under `min.insync.replicas`, online and offline logDirs, and distribution of replicas in the cluster.
- Multi-goal rebalance proposal generation for:
  - Rack-awareness
  - Resource capacity violation checks (CPU, DISK, Network I/O)
  - Per-broker replica count violation check
  - Resource utilization balance (CPU, DISK, Network I/O)
  - Leader traffic distribution
  - Replica distribution for topics
  - Global replica distribution
  - Global leader replica distribution
  - Custom goals that you wrote and plugged in

2. 客戶端自動化負載平衡...???

自動不如好用

https://github.com/skiptests/astraea

一致的使用方式

1. 寫入端的自動化負載平衡

① 選擇適當的成本
② 給定成本的權重

2. 伺服器端的自動化負載平衡

① 選擇適當的成本
② 給定成本的權重

3. 讀取端的自動化負載平衡

① 選擇適當的成本
② 給定成本的權重

很一致所以說三次!!!

多樣化的成本選擇

延遲

吞吐量

leader數量

副本

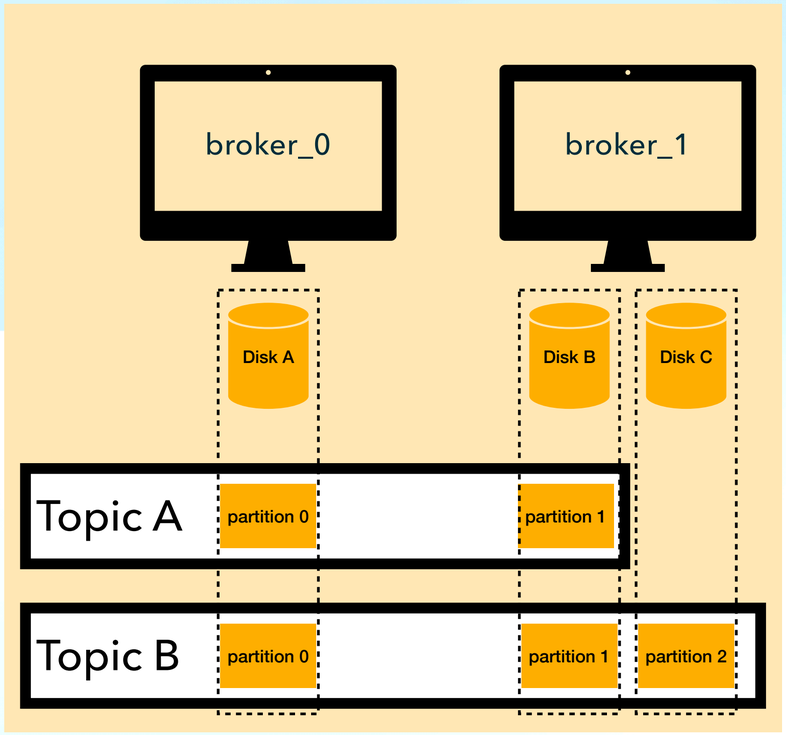
連線數量

Partitions數量

負載平衡就是
做適當的選擇

萬物皆有數字

1. 給定一個叢集描述



cost function {}

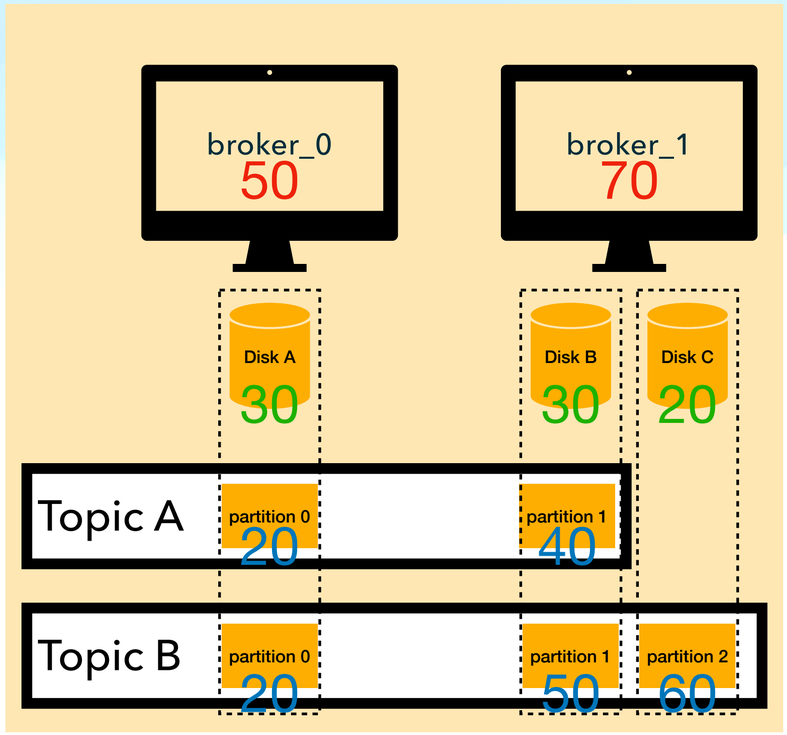
伺服器效能指標

叢集資訊

寫入端效能指標

讀取端效能指標

2. 輸出叢集各項分數
（ 這個叢集本身也是一個數字 ）



從寫入端做負載平衡

Phase 1

效能指標

伺服器效能指標

叢集資訊

寫入端效能指標

讀取端效能指標

節點分數

B1 5分

B2 2分

B3 10分

Cost Functions

f0 { } W=5

f1 { } W=3

f2 { } W=2

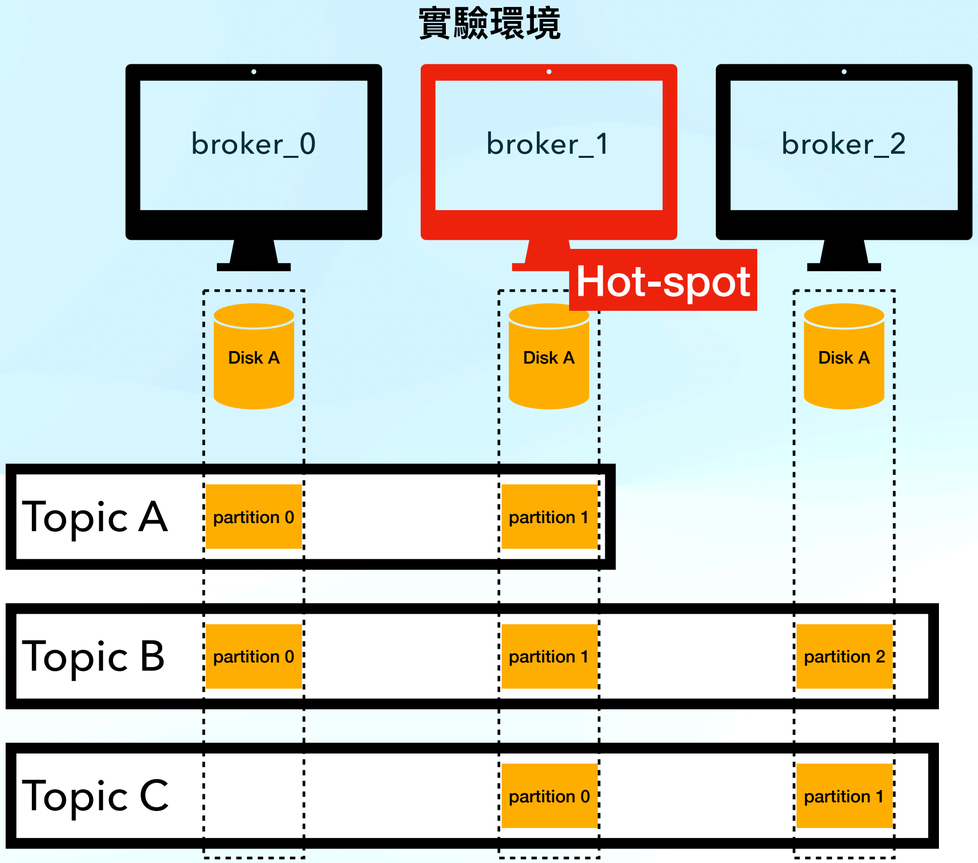
Round-Robin Algorithm

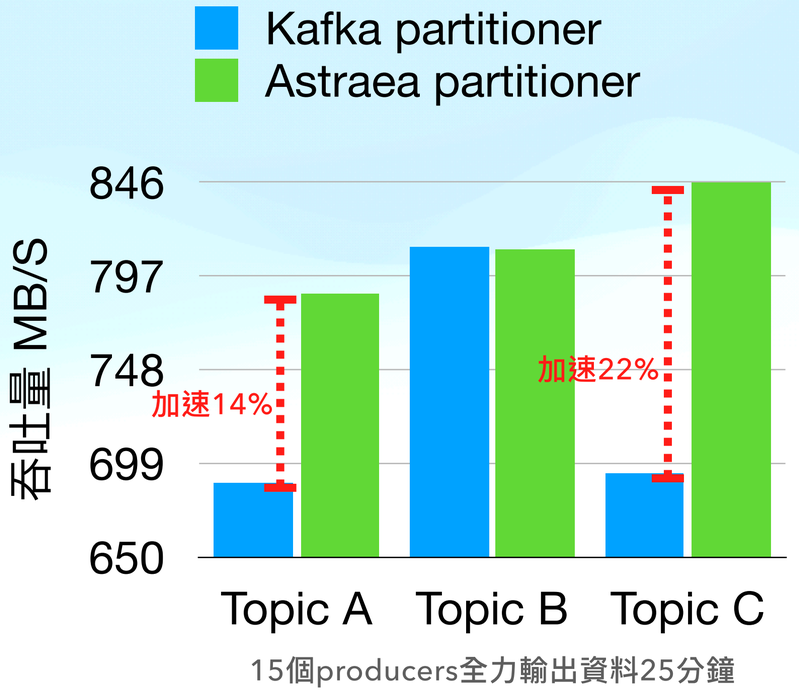
節點選擇順序

B3 B1 B3 B2 B3 B1 B3 B3 B1

Phase 2

Solution 2: Write to Free Node





15個producers全力輸出資料25分鐘

自動根據負載配置流量

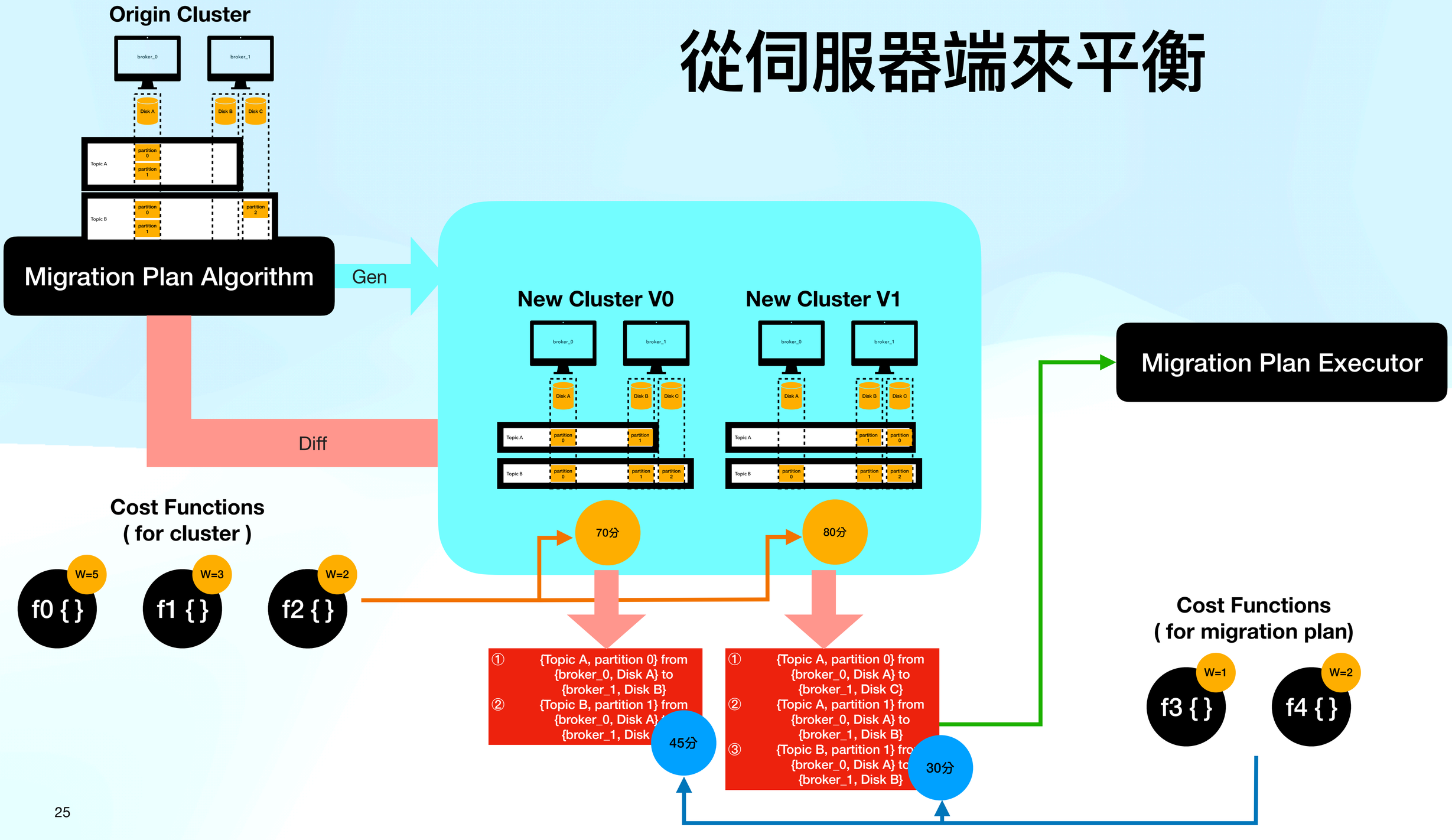
↓

均勻的吞吐量

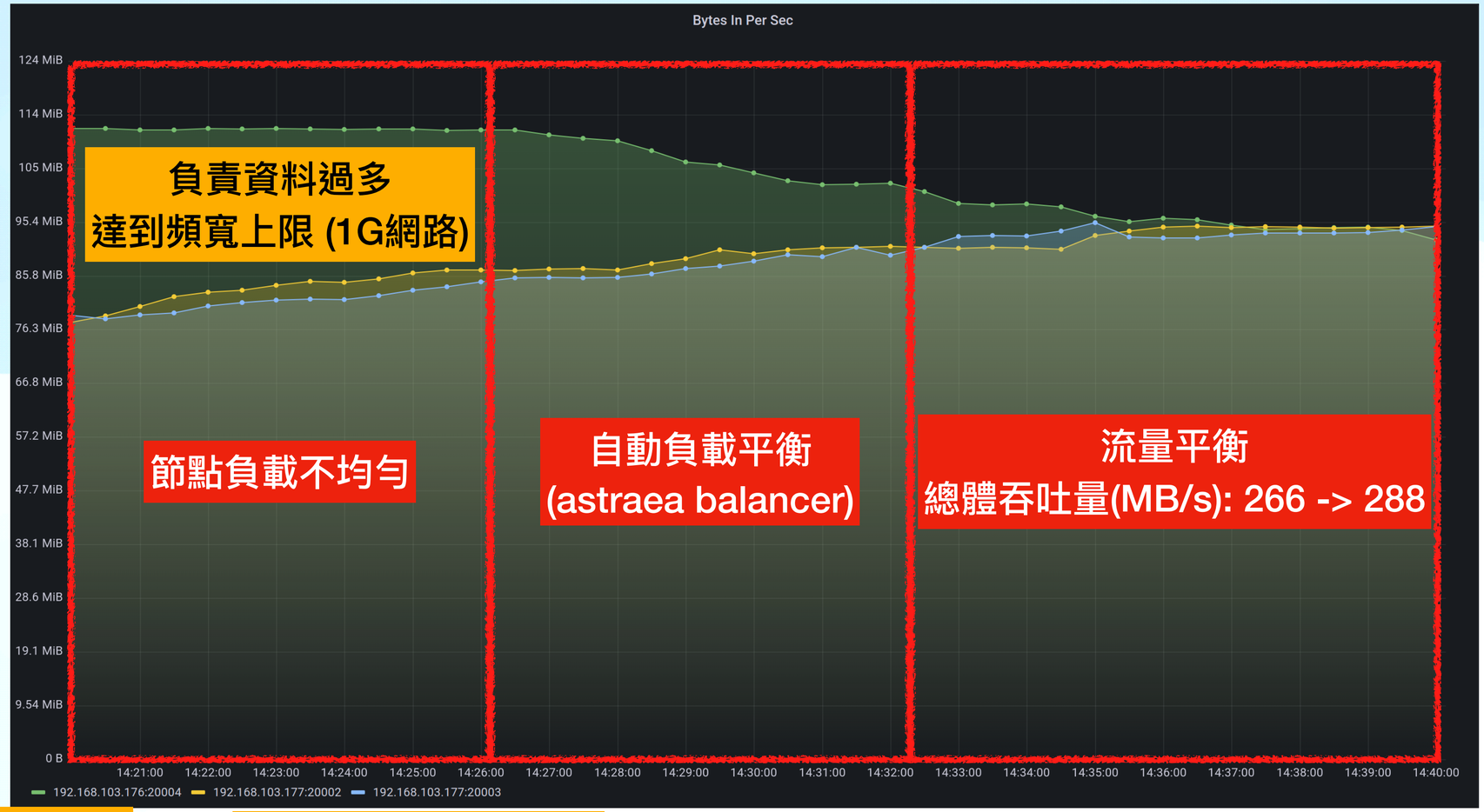
↓

硬體資源有效利用

從伺服器端來平衡


Origin Cluster
broker_0
broker_1
Disk A
Disk B
Disk C
Topic A
partition 0
partition 1
Topic B
partition 0
partition 1
partition 2
Migration Plan Algorithm
Gen
Diff
New Cluster V0
New Cluster V1
broker_0
broker_1
Disk A
Disk B
Disk C
Topic A
partition 0
partition 1
Topic B
partition 0
partition 1
partition 2
70分
80分
Migration Plan Executor
Cost Functions
(for cluster)
W=5
f0 { }
W=3
f1 { }
W=2
f2 { }
① {Topic A, partition 0} from {broker_0, Disk A} to {broker_1, Disk B}
② {Topic B, partition 1} from {broker_0, Disk A} to {broker_1, Disk
45分
① {Topic A, partition 0} from {broker_0, Disk A} to {broker_1, Disk C}
② {Topic A, partition 1} from {broker_0, Disk A} to {broker_1, Disk B}
③ {Topic B, partition 1} fro {broker_0, Disk A} to {broker_1, Disk B}
30分
Cost Functions
(for migration plan)
W=1
f3 { }
W=2
f4 { }

Solution 1: Balance the Cluster



最早運行的節點

事後加入的節點分到
比較少的partitions

ASTRAEA

拷問時間