# Apache Kafka 叢集負載平衡

演講者 : 李政憲

July 2022 @ COSCUP

Link to the Astraea Project
https://github.com/skiptests/astraea

# 關於我

- 成功大學的研究生
  - 研究 Kafka 的負載平衡問題
  - https://www.linkedin.com/in/zheng-xian-li-03b039247
- 維護 Astraea 專案
  - 一系列 Kafka 維運工具
  - https://github.com/skiptests/astraea

- 演講者
  - 李政憲
- 核心開發者 (sort by unicode order)
  - 孫祥鈞, 方竫泓, 李宜桓, 李政憲, 王懿宸, 蔡嘉平, 蕭宏章, 鄧智懋, 魏連興
- 特別感謝
  - 科管局
  - 成功大學
  - 原昌工業
  - 亦思科技
  - 教育部
  - 來自其他公司的工程師的貢獻

# 今天的演講

大綱

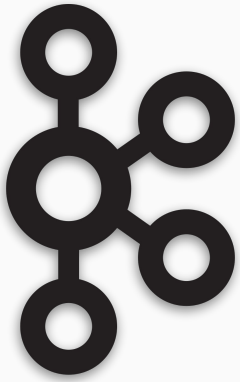
1. Apache Kafka, The Good & Ugly.
2. 說明負載 & 平衡問題的源頭
3. 叢集負載平衡難處
4. 我們的解決方法

# Apache Kafka, The Good & Ugly.

說明負載 & 平衡問題的源頭
叢集負載平衡難處
我們的解決方案

# Kafka The Good

Event Streaming
Transaction
Compression
JBOD
Fault Tolerance
Encryption
Resource Quota
Replication

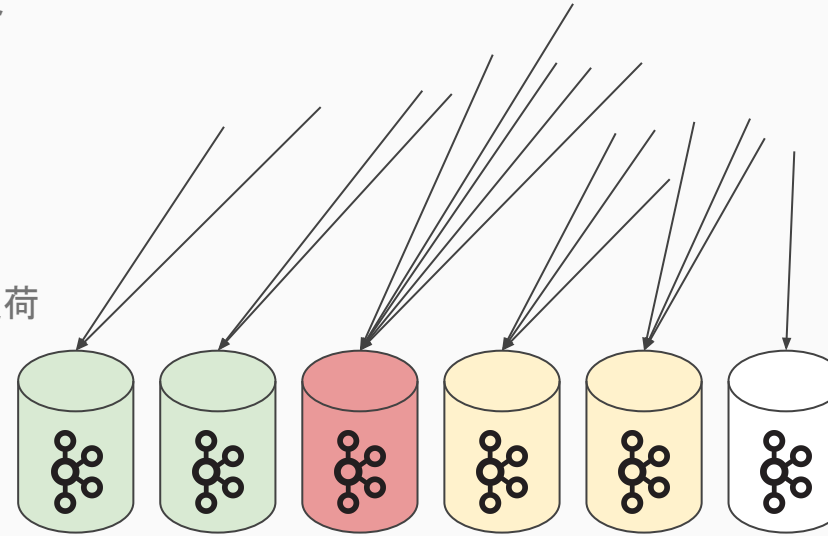


Authentication
Authorization
At-most once delivery
At-least once delivery
Pub/Sub Pattern
Exactly once delivery
Multi-Tenancy
High Throughput
Low Latency

Apahce Kafka 解放了許多應用的開發難度，讓我們少刻一些輪子

# 存在負載不平衡的問題!

- Apache Kafka 內建負載平衡存在一些缺失
  - 維運者需要想辦法解決這些問題
- 運行大規模的叢集. 然後...
  - 某些節點的 incoming/outgoing 流量超過硬體負荷
  - 某些節點的下線造成很顯著的服務品質影響



Text filled by me. Meme generated by https://imgflip.com/memegenerator/Boardroom-Meeting-Suggestion

# 服務效率被影響

- 負載不平衡的影響
  - 上游應用可能運作不穩定
- 如何解決
  - 從應用端優化？應用端可能不受你的控制？
  - 從伺服器端優化？叢集負載平衡！
  - 添購更多設備？
    - 資源過剩
    - 無法解燃眉之急

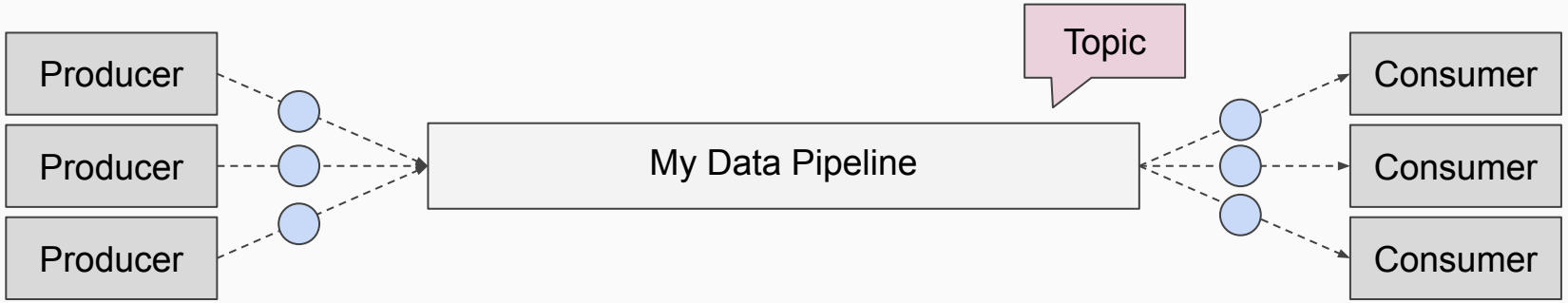
Apache Kafka, The Good & Ugly.

# 說明負載 & 平衡問題的源頭

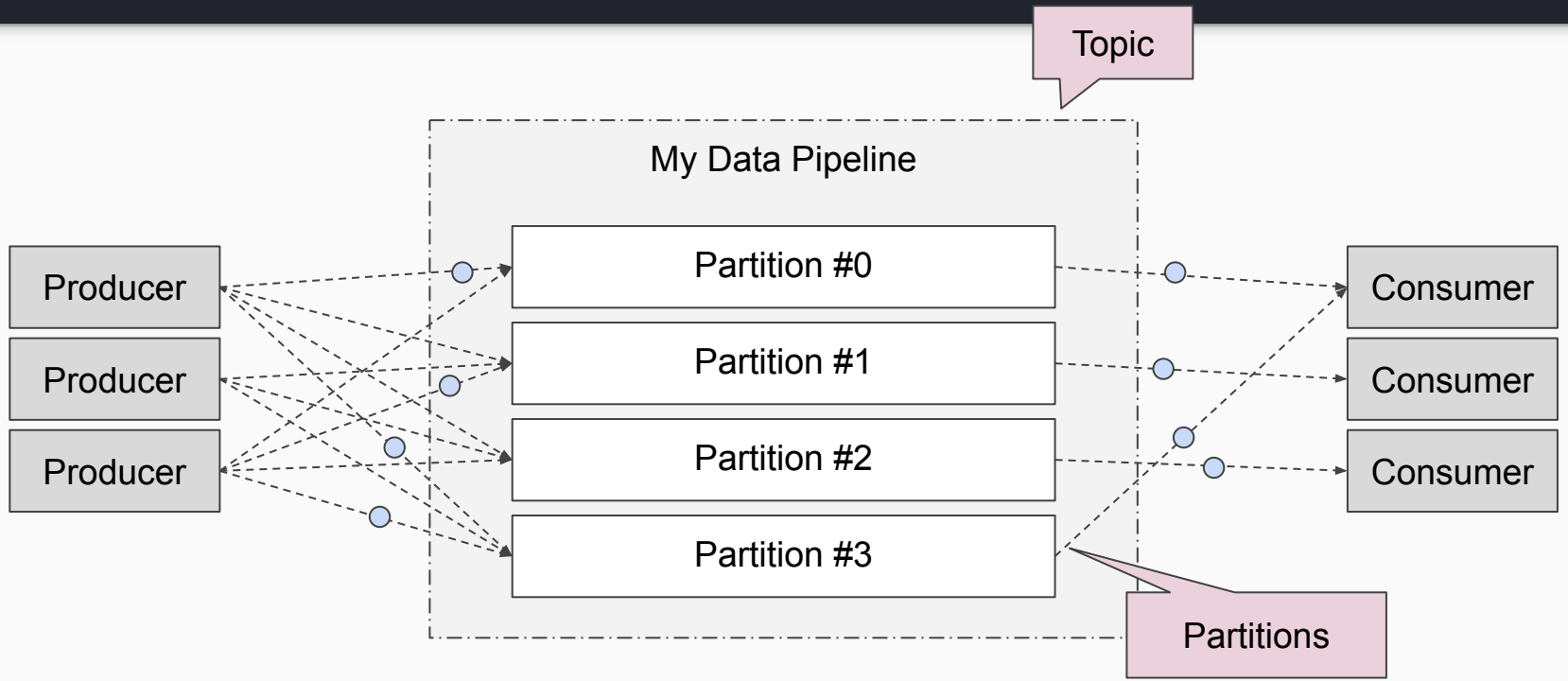
叢集負載平衡難處

我們的解決方案

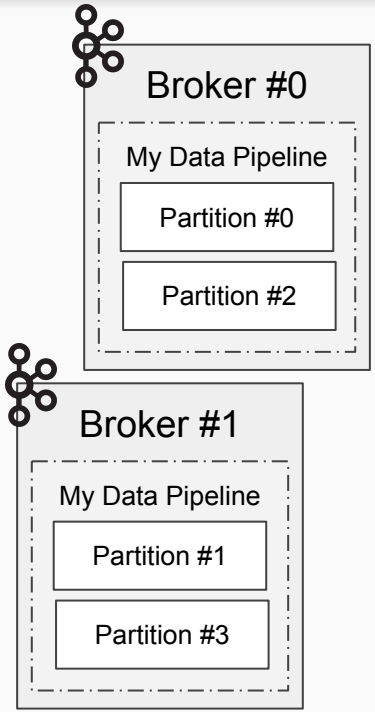
# Apache Kafka 的模型 (ignore brokers)



# Apache Kafka 的模型 (ignore brokers)



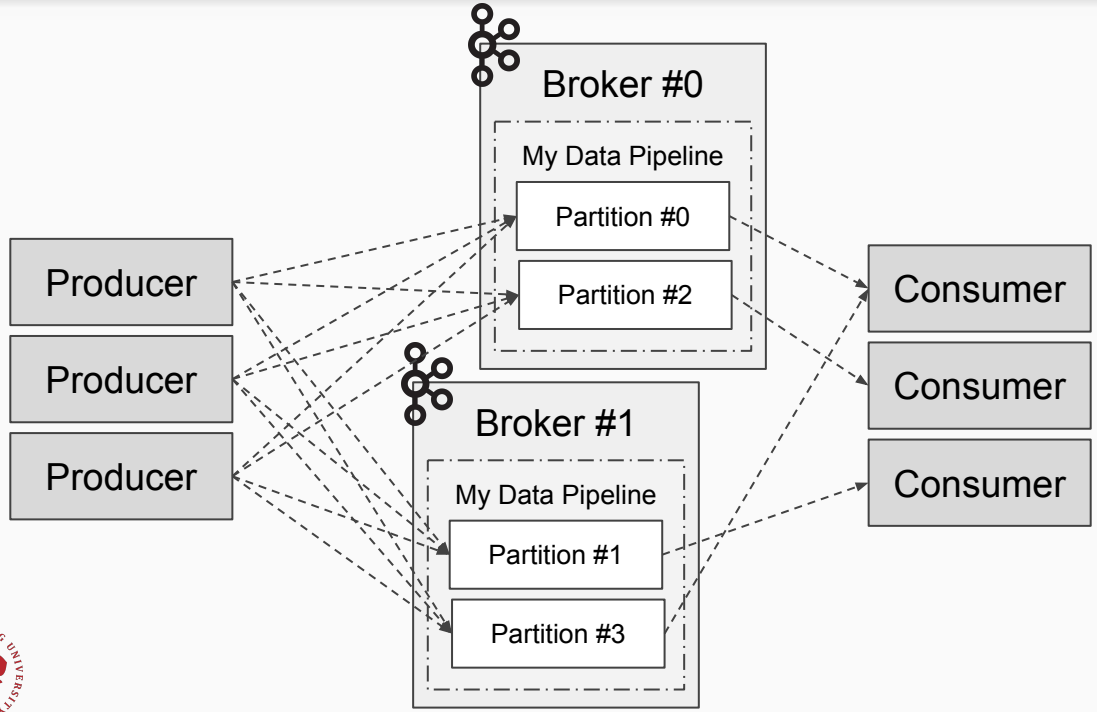
# Apache Kafka 的模型 (with Brokers)




Partition 負載分佈表格

| 節點 | 負載 |
| --- | --- |
| 0 | MyDataPipeline#0<br>MyDataPipeline#2 |
| 1 | MyDataPipeline#1<br>MyDataPipeline#3 |

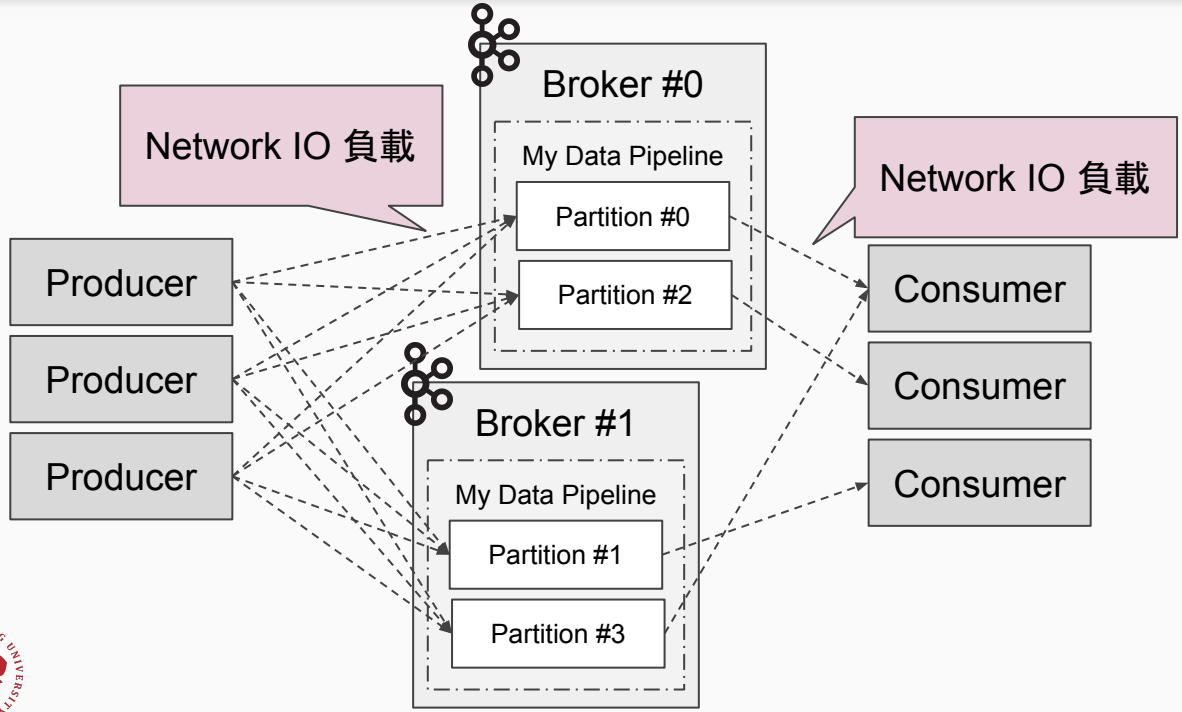
# Apache Kafka 的模型 (with Brokers)




Partition 負載分佈表格

| 節點 | 負載 |
| --- | --- |
| 0 | MyDataPipeline#0<br>MyDataPipeline#2 |
| 1 | MyDataPipeline#1<br>MyDataPipeline#3 |

# Apache Kafka 的模型 (with Brokers)




Partition 負載分佈表格

| 節點 | 負載 |
| --- | --- |
| 0 | MyDataPipeline#0<br>MyDataPipeline#2 |
| 1 | MyDataPipeline#1<br>MyDataPipeline#3 |

# Apache Kafka Partition 負載分佈如何決定?

- Kafka 如何決定 Partition 分佈?

| 節點 | 負載 |
| --- | --- |
| ??? | MyDataPipeline#0 |

# Apache Kafka Partition 負載分佈如何決定?

- Kafka 如何決定 Partition 分佈?

| 節點 | 負載 |
| --- | --- |
| ??? | MyDataPipeline#0 |

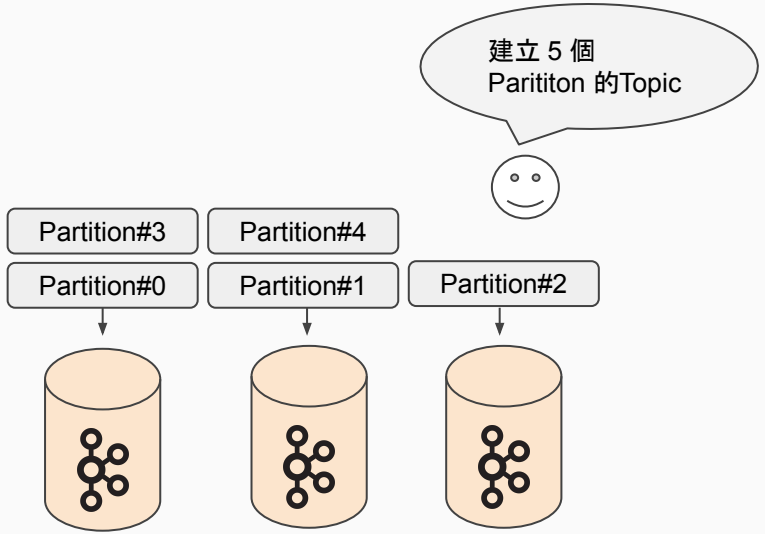
  - 負載分佈的方法沒有被記載在任何文獻
  - 這是 Implementation-Detail

# Apache Kafka Partition 負載分佈如何決定?

- Kafka 如何決定 Partition 分佈?

| 節點 | 負載 |
| --- | --- |
| ??? | MyDataPipeline#0 |

  - 負載分佈的方法沒有被記載在任何文獻
  - 這是 Implementation-Detail
- Partition 分佈決定的實作細節
  - **儘可能確保各節點的 Partition 數量一樣多**
  - 分佈決定的時間點在建立 Topic 時
  - 分佈建立後即固定，除非人為改變




(fig) Kafka 預設實作嘗試平分每個節點的 log 數量

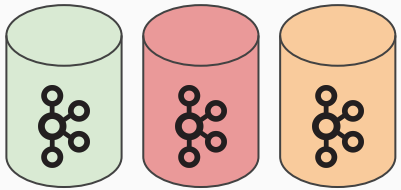
# Apache Kafka 的其他負載來源

- 其他節點端的硬體資源開銷
  - Broker 端的資料壓縮操作（CPU)
  - Broker 端運行 CPU 操作
  - Memory for Filesystem Cache
- Network IO 是比較顯著的負載來源
  - Let’s ignore the others :3

# 確保大家有跟上
# 負載的主要源頭是?

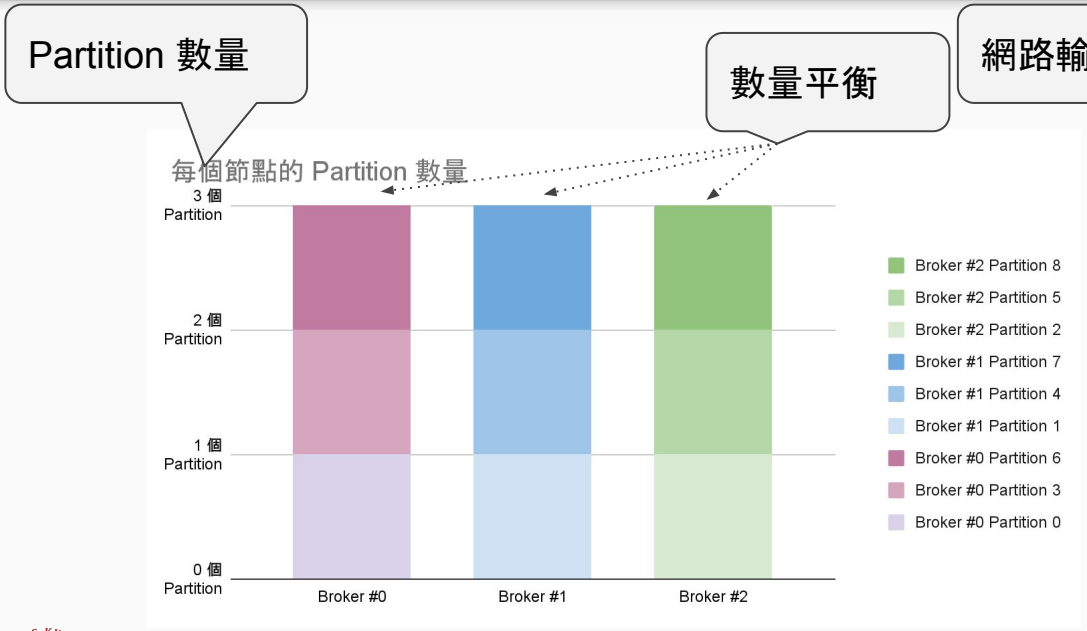
現在大家都知道 Kafka 最主要的負載源自 _____partition_____ 的分佈關係.

接下來要提到負載不平衡的原因

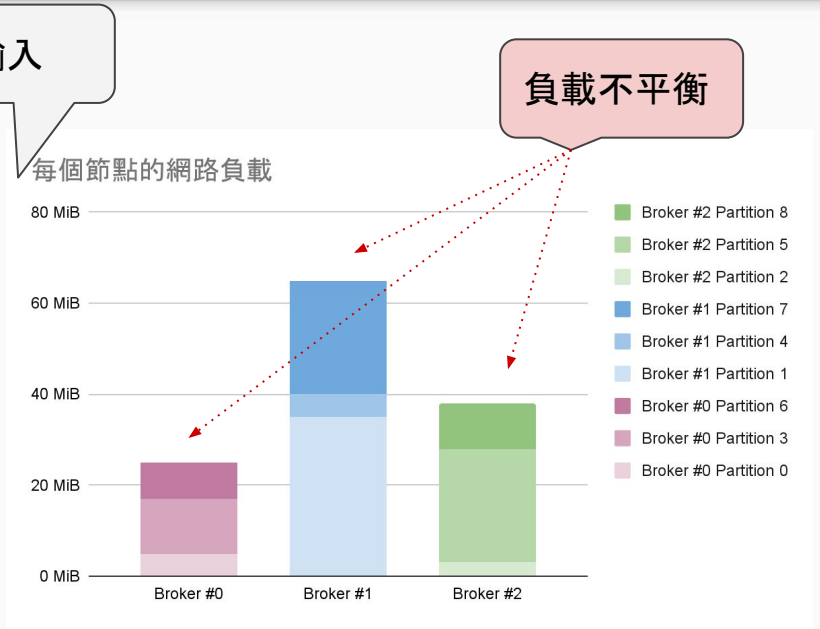


| 節點 | 負載 |
| --- | --- |
| 0 | MyDataPipeline#0<br>MyDataPipeline#2 |
| 1 | MyDataPipeline#1<br>MyDataPipeline#3 |

# 源頭 #1: Partition 數量平衡 != 負載平衡



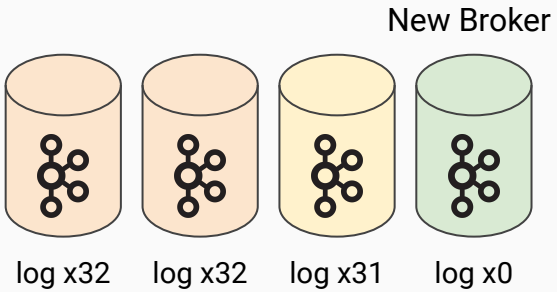

(fig) Kafka 預設實作嘗試平衡每個節點的 Partition 數量




(fig) 但每個 Partition 的負載量可能不太一樣

# 源頭 #2: 擴張叢集，添加節點

替叢集添加新節點，既有的負載不會自動平衡過去



Apache Kafka, The Good & Ugly.

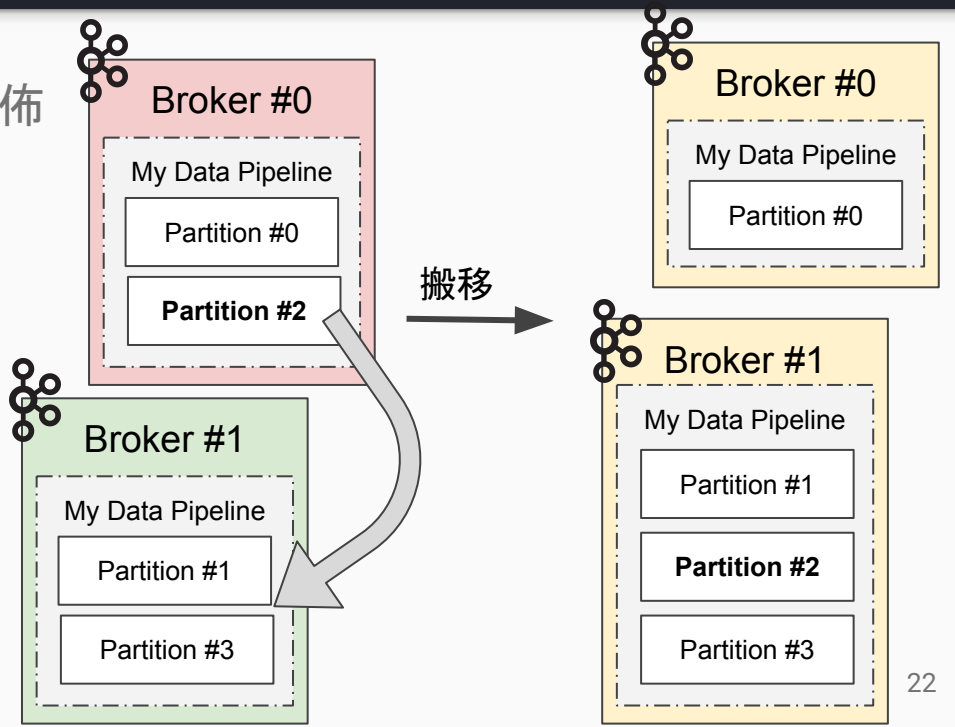
說明負載 & 平衡問題的源頭

# 叢集負載平衡難處

我們的解決方案

# 如何手動負載平衡

- 調整 Partition 分佈可以改變流量分佈
  - Script
    `bin/kafka-reassign-partitions.sh`
  - Java
    Use Apache Kafka AdminClient API
- Call API then call it a day?
  - 真的有那麼簡單嗎 OAO



# 難題 #1: 大量的負載平衡選擇/限制/目標

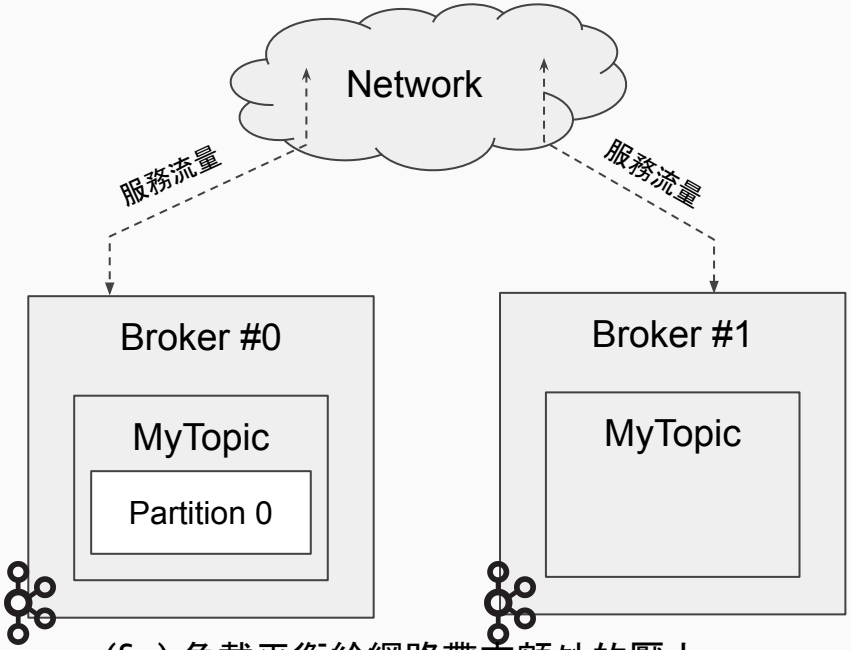
- 叢集有 n 個 Brokers，則每個 Log 有 n-1 種潛在的新位置
  - A typical cluster might contain more than 10+ brokers. And 3000+ logs.
- 多種平衡/滿足的目標
  - Incoming Bandwidth for broker or network switch.
  - Outgoing Bandwidth for broker or network switch.
  - Leader Count.
  - Rack Awareness Requirement.
  - …
- 重點關心項目
  - Average Latency
  - 99th Percentile Latency
  - Throughput

# 難題 #2: 負載平衡的過程存在顯著的 IO 成本

- 網路壓力
  - 服務流量

(table) 現在的分佈

| 節點 | TopicPartition |
| --- | --- |
| 0 | MyTopic#0 |




(fig) 負載平衡給網路帶來額外的壓力

# 難題 #2: 負載平衡的過程存在顯著的 IO 成本

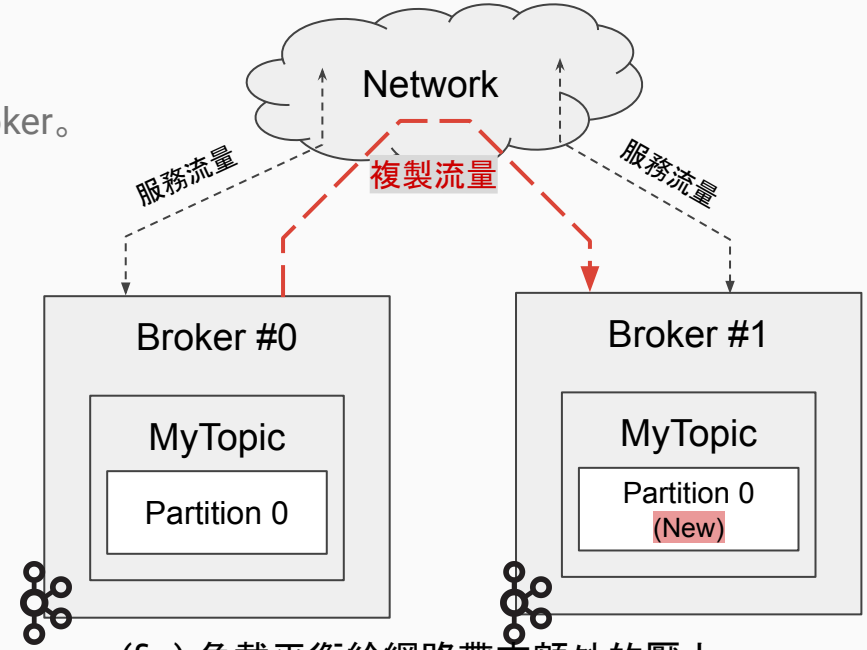
- 變更 Partition 分佈
  - 需要自當前 Partition 複製資料至新 Broker。
- 網路壓力
  - 服務流量 + **複製流量**
  - 儘量避免劇烈平衡操作，影響上游服務
  - 搬移速度太慢，有可能搬不完

(table) 分佈變更前

| 節點 | TopicPartition |
|---|---|
| 0 | MyTopic#0 |

(table) 分佈變更後

| 節點 | TopicPartition |
|---|---|
| **1** | MyTopic#0 |




(fig) 負載平衡給網路帶來額外的壓力

Apache Kafka, The Good & Ugly.
說明負載 & 平衡問題的源頭
叢集負載平衡難處

# 我們的解決方案

# 提出負載平衡計劃的工具 - Astraea Balancer

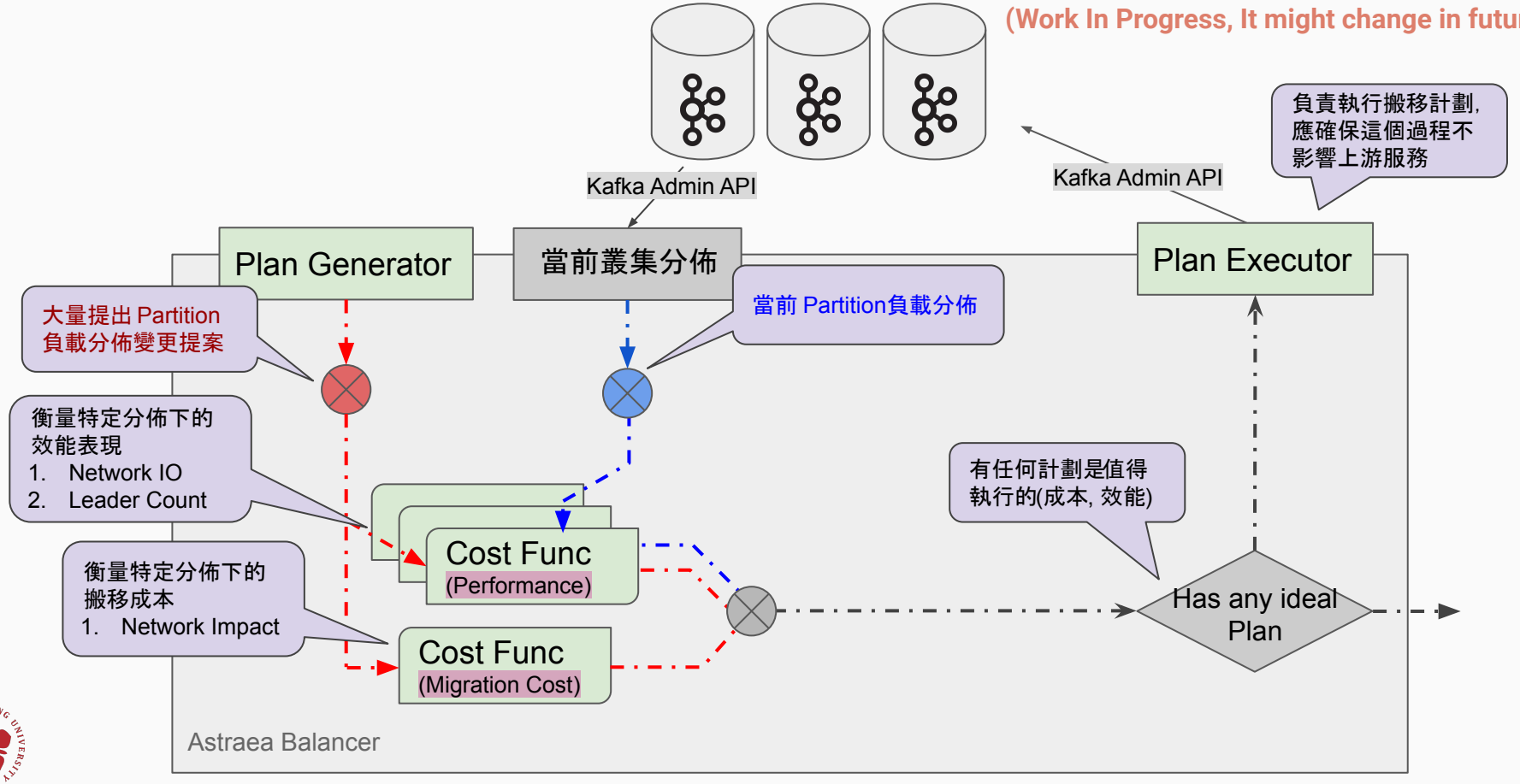
| 節點 | 負載 |
| --- | --- |
| 0 | MyDataPipeline#0<br>**MyDataPipeline#2** |
| 1 | MyDataPipeline#1<br>MyDataPipeline#3 |
| … | … |

| 節點 | 負載 |
| --- | --- |
| 0 | MyDataPipeline#0 |
| 1 | MyDataPipeline#1<br>**MyDataPipeline#2**<br>MyDataPipeline#3 |
| … | … |

& 叢集變得更美好(defined by user)
整體執行代價系統可接受

幫你提出 & 執行一個負載平衡計劃

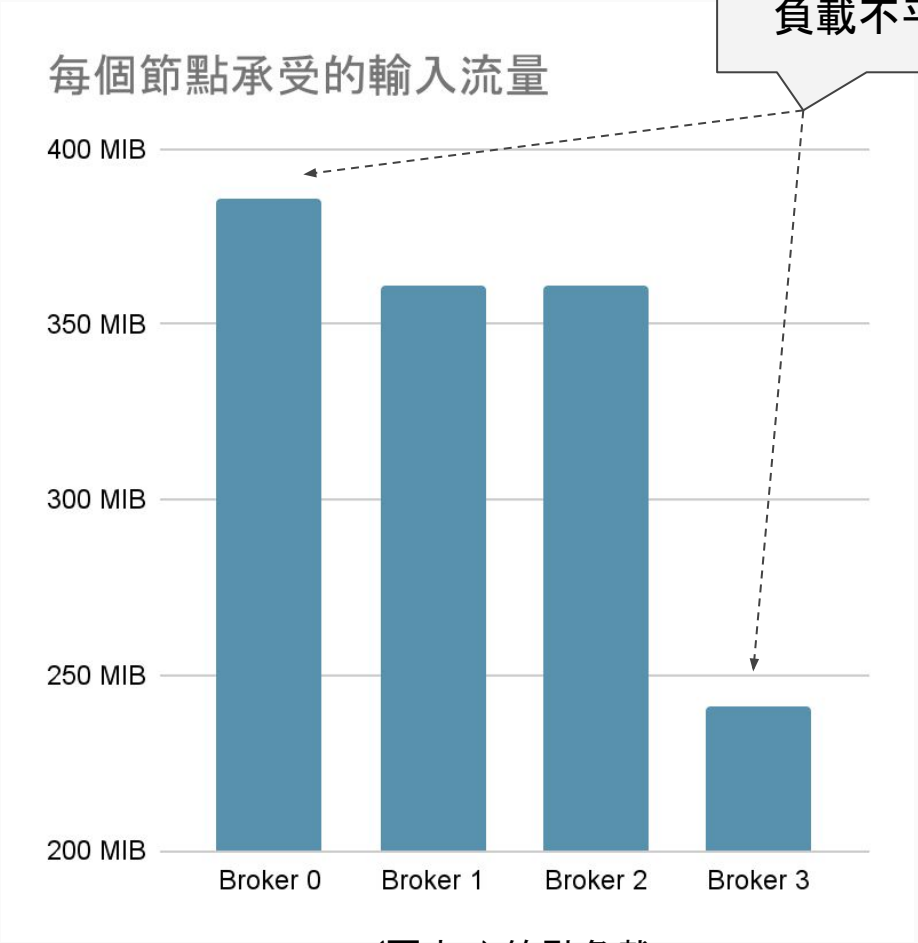
# Astraea Balancer 流程簡述



# 實驗設計

- 平衡目標
  - 節點輸入流量
- 系統 & 硬體環境
  - 4 Brokers
  - 2 Producers
  - 1 GB Net Bandwidth
- 情境

| | |
|---|---|
| Topics | 95 |
| Partitions | 803 |
| Broker 輸入負載 | (見圖表1) |



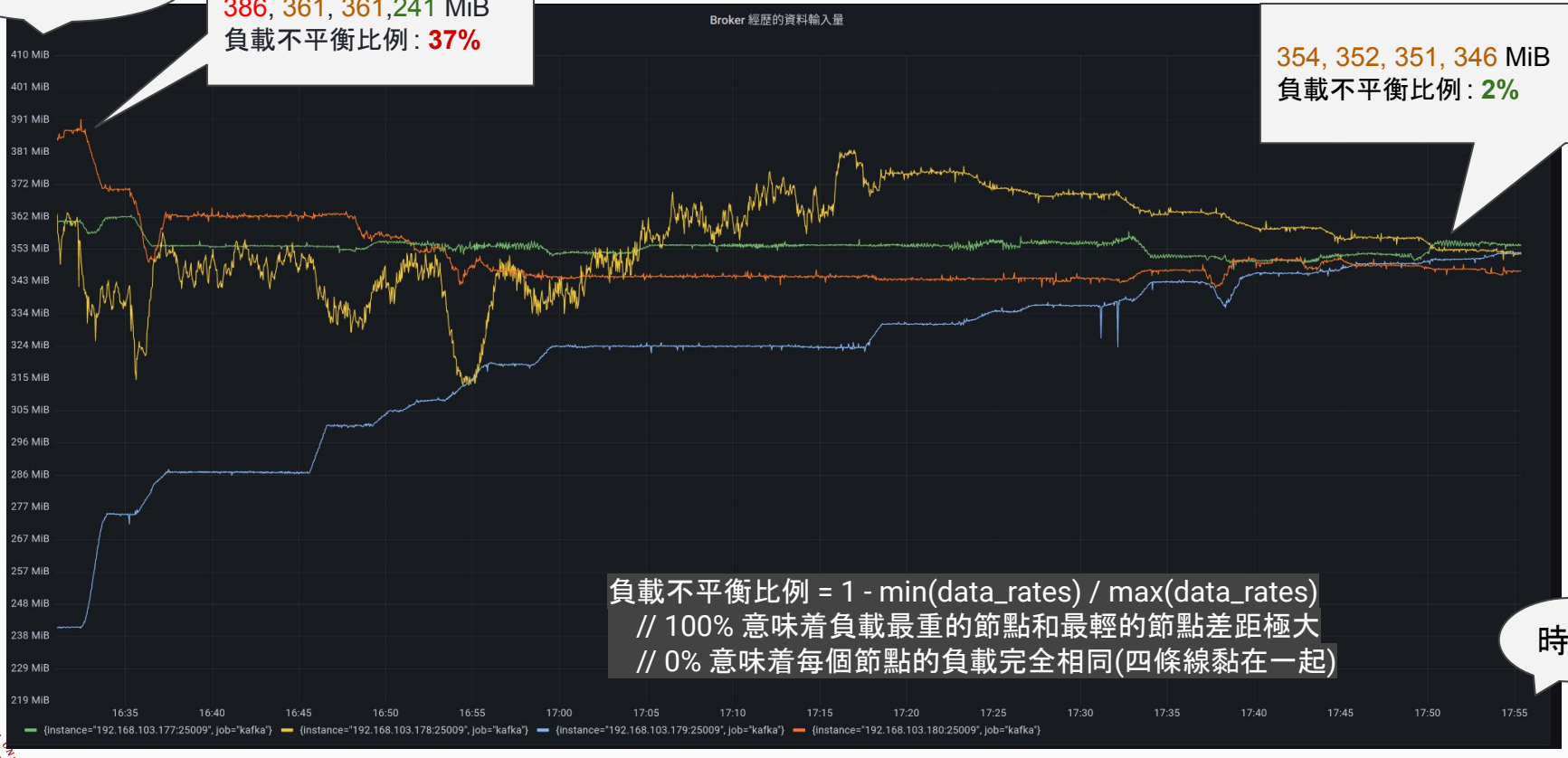

(圖表1) 節點負載

# 實驗結果

節點流量

386, 361, 361,241 MiB
負載不平衡比例 : **37%**

354, 352, 351, 346 MiB
負載不平衡比例 : **2%**




負載不平衡比例 = 1 - min(data_rates) / max(data_rates)
// 100% 意味着負載最重的節點和最輕的節點差距極大
// 0% 意味着每個節點的負載完全相同(四條線黏在一起)

時間

# Q&A Session

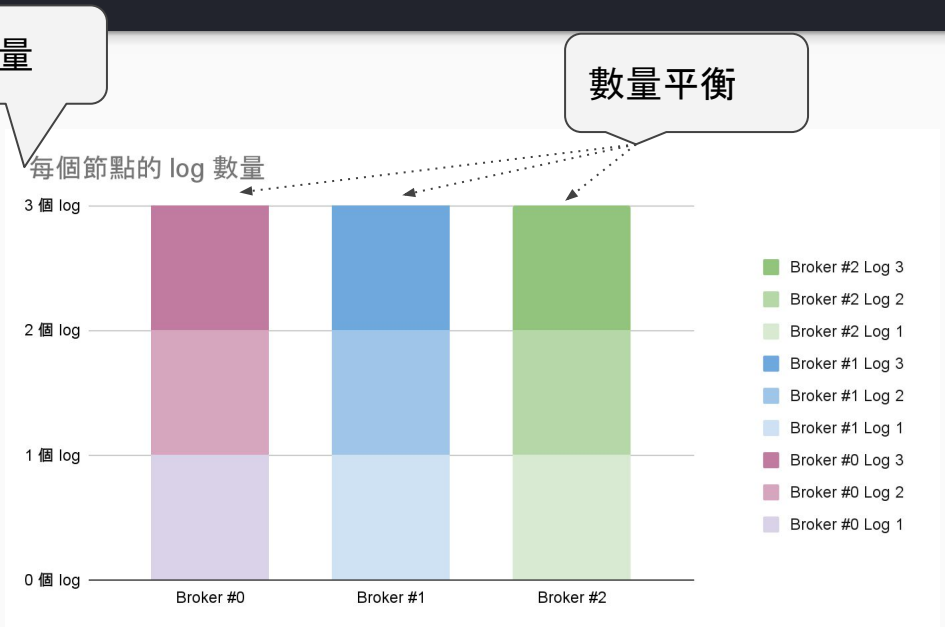
Link to the Astraea Project

https://github.com/skiptests/astraea

# Ignored Content

# 源頭 #1: 各 Partition 的負載差異 (1/2)

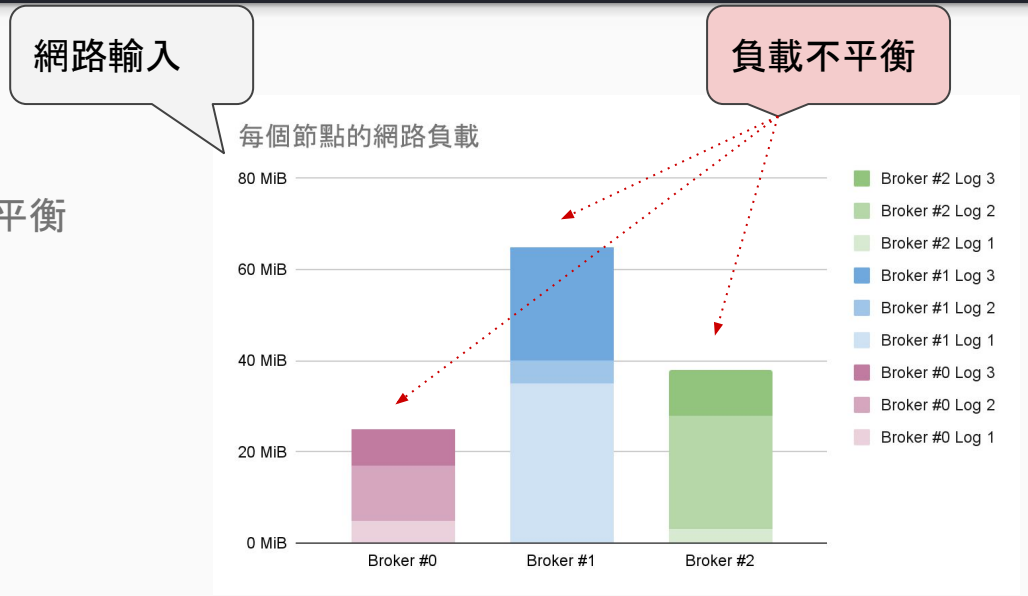
- Kafka 預設的負載平衡手法
  - 嘗試平衡每個節點的負載數量
- 這個手法存在盲點




(fig) Kafka 預設實作嘗試平衡每個節點的 log 數量

# 源頭 #1: 各 Partition 的負載差異 (2/2)

- 考量 Log 實際負載
  - 每個 Log 的負載可能不一
  - 在簡單的數量策略下可能不平衡




(fig) 但每個 Log 的負載量可能不太一樣

# 過於真實的鬼故事

純靠北工程師. 2022. *#純靠北工程師4rx | 現在正在經歷的鬼故事我這裡有台C....* [online] Available at: <https://init.engineer/cards/show/6189> [Accessed 22 July 2022].

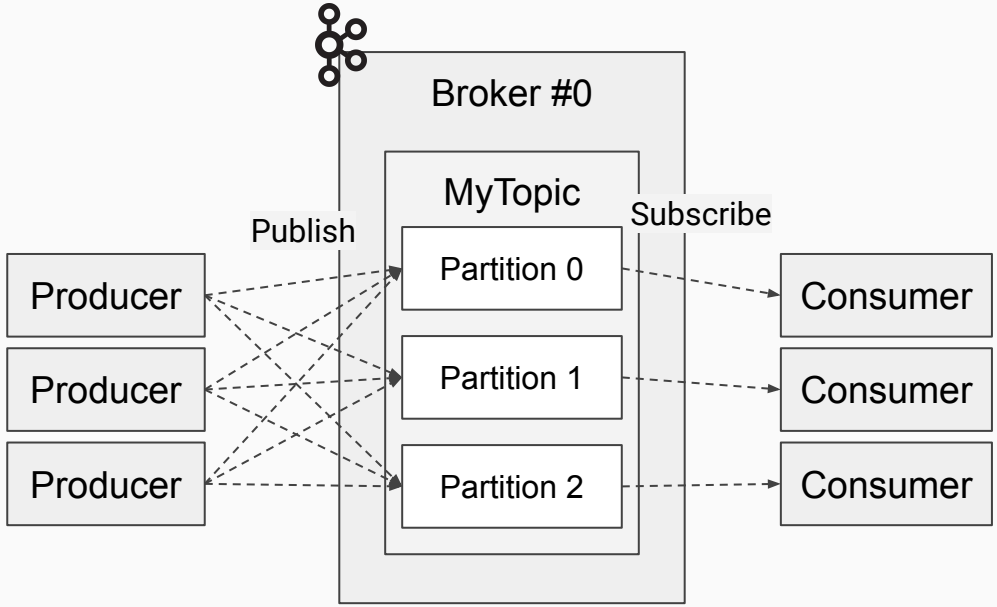

附註：這個故事的問題應該是Switch 為 IO 密集設備。

# Apache Kafka 的負載來源 (1/4)

- 專注在一個節點
- Network IO 密集操作

| TopicPartition | Partition Assignment |
|---|---|
| MyTopic-0 | [0] |
| MyTopic-1 | [0] |
| MyTopic-2 | [0] |

(table) 每個 Topic/Partition 的分區分配清單



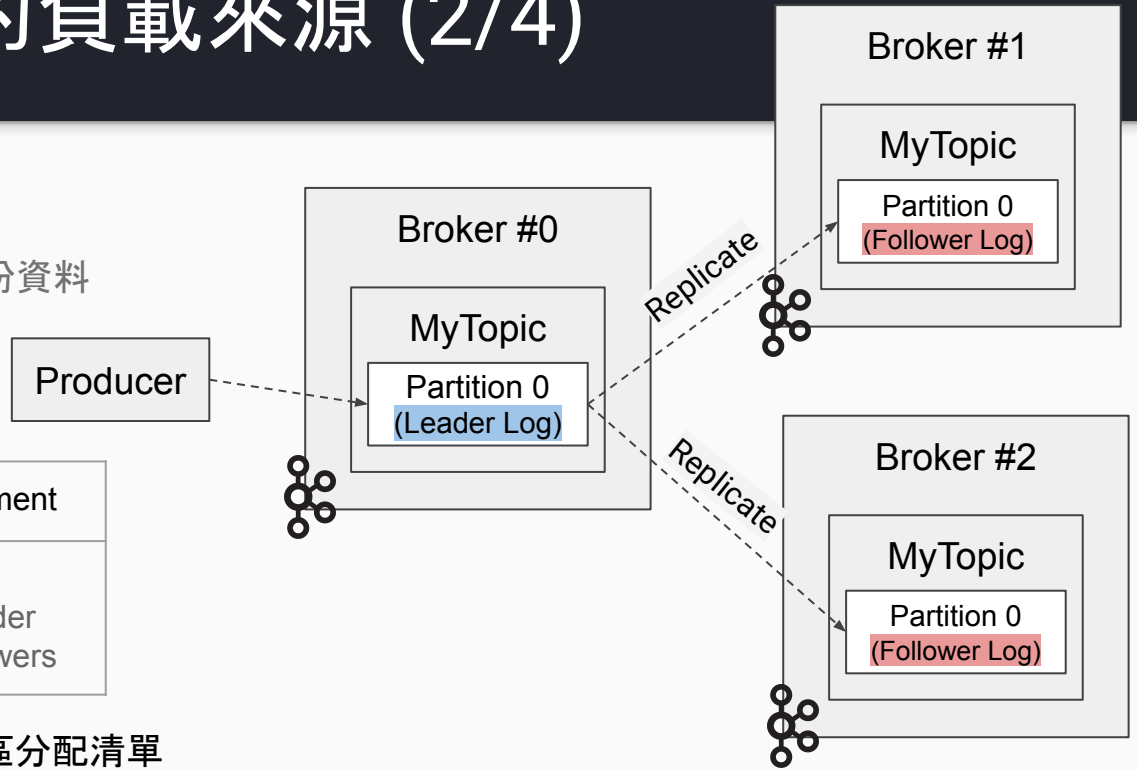

(fig) 於常見設定下的應用資料負載範例 .

# Apache Kafka 的負載來源 (2/4)

- 存在副本設定?
  - Partition Leader 需要備份資料至其他節點
- IO 密集操作

| TopicPartition | Partition Assignment |
| --- | --- |
| MyTopic-0 | [0, 1, 2]<br>// where 0 is leader<br>// 1 & 2 are followers |

(table) 每個 Topic/Partition 的分區分配清單




(fig) 由於副本所造成的資源開銷

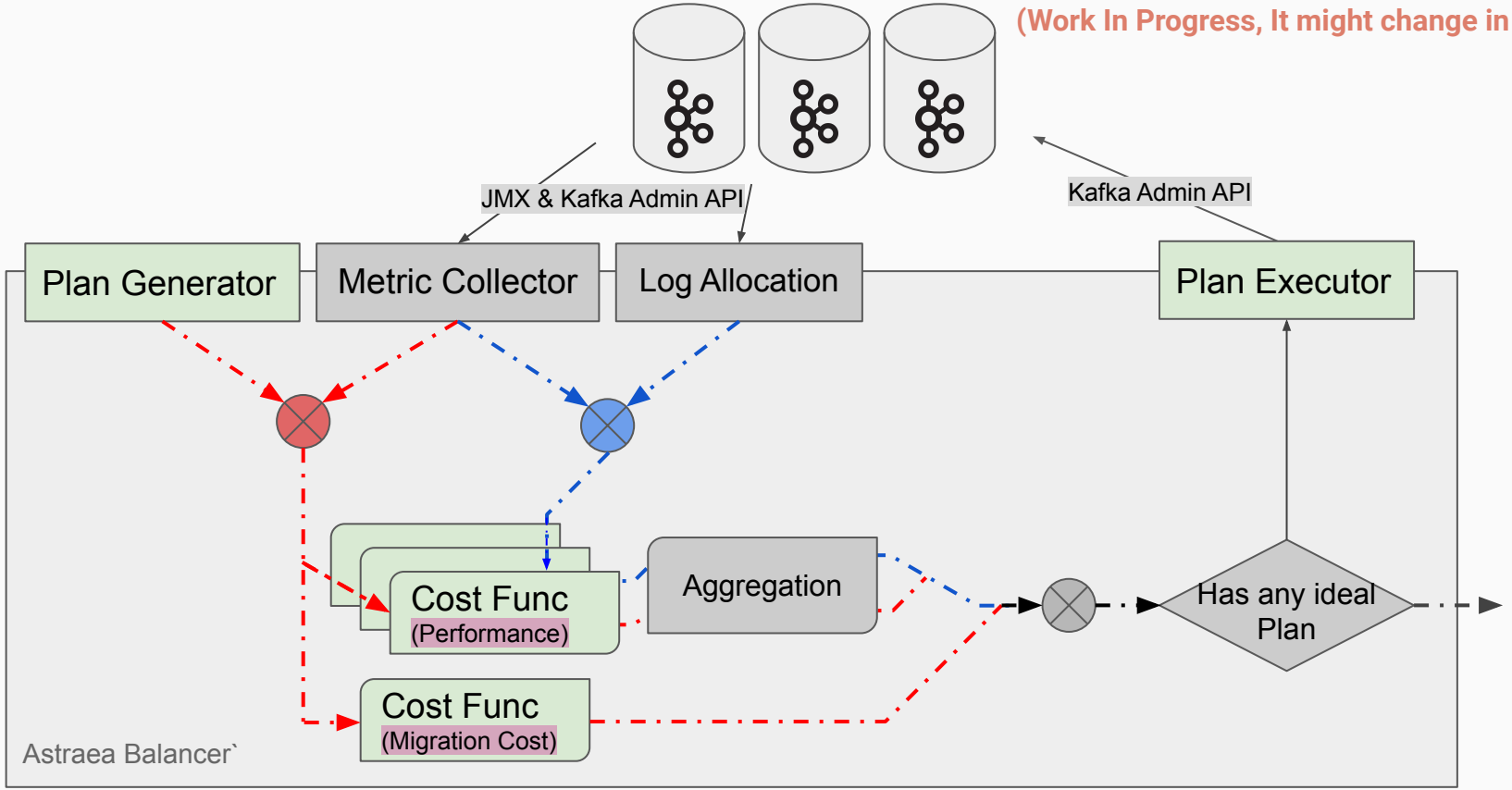
# 難題 #3: 不同的設定可能顯著地影響流量走向

- 某些叢集用法會大幅度地改變流量方向
  - Broker 插多張網卡
  - 網路環境不單純
  - Consumer 從 Follower 消費資料(需要 Rack Awareness) see KIP-392
    - Consume From Leader
      - 1000 Consumer, Log Increase Rate: 100 KB/s
      - 100 MB/s outgoing.
    - Consume From Follower (assume consumer spread evenly)
      - replica factor = 5
      - 20 MB/s per log

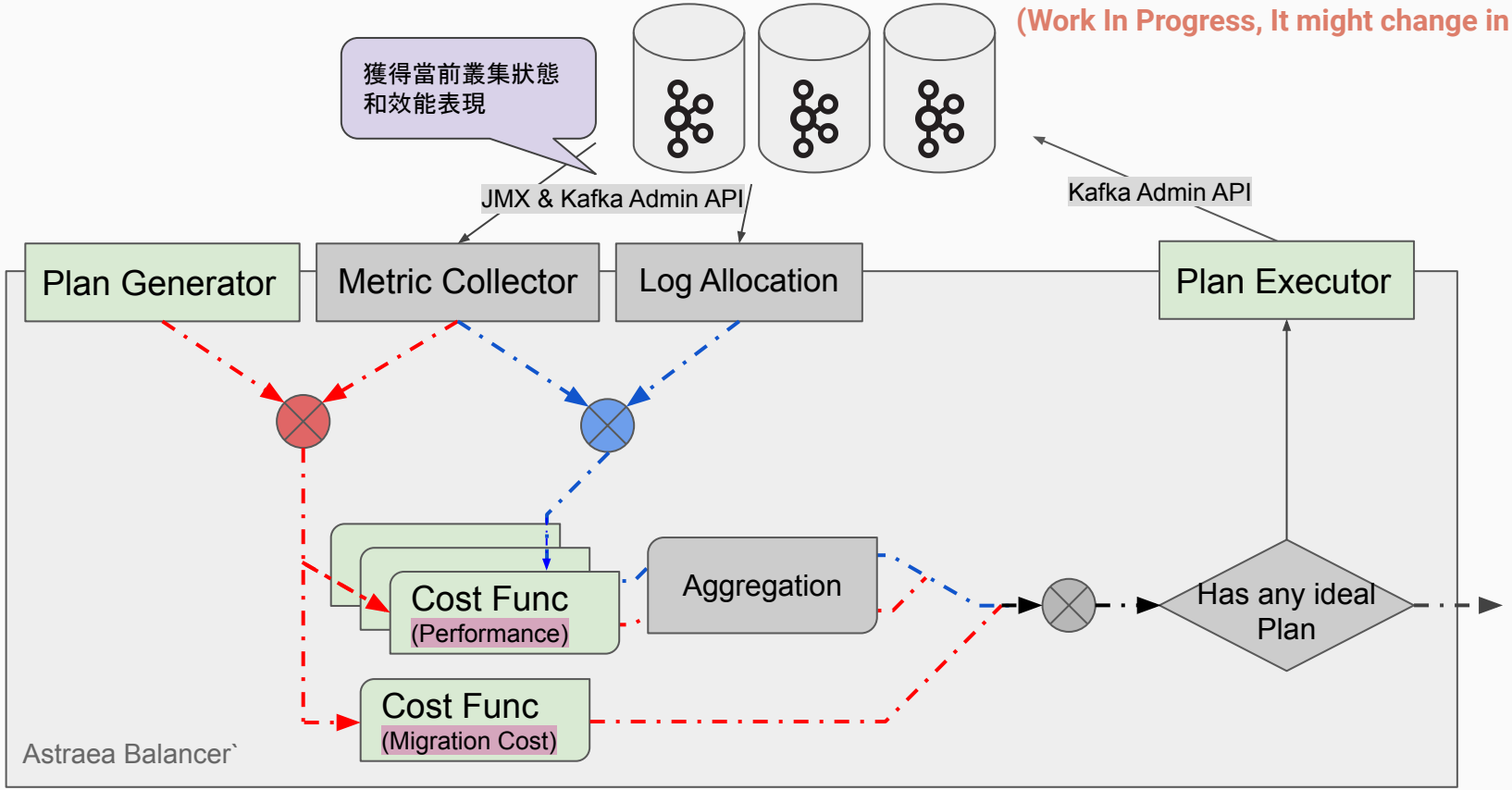
# 難題 #4: 選擇基於過去資訊，未來充滿不確定性

- 負載平衡的決策多半基於過去的服務表現
  - Log 資料流入/輸出量
- 負載平衡決策充滿不確定性
  - 現在的決策在短時間的未來會不會不適合？
  - Stochastic Programming?

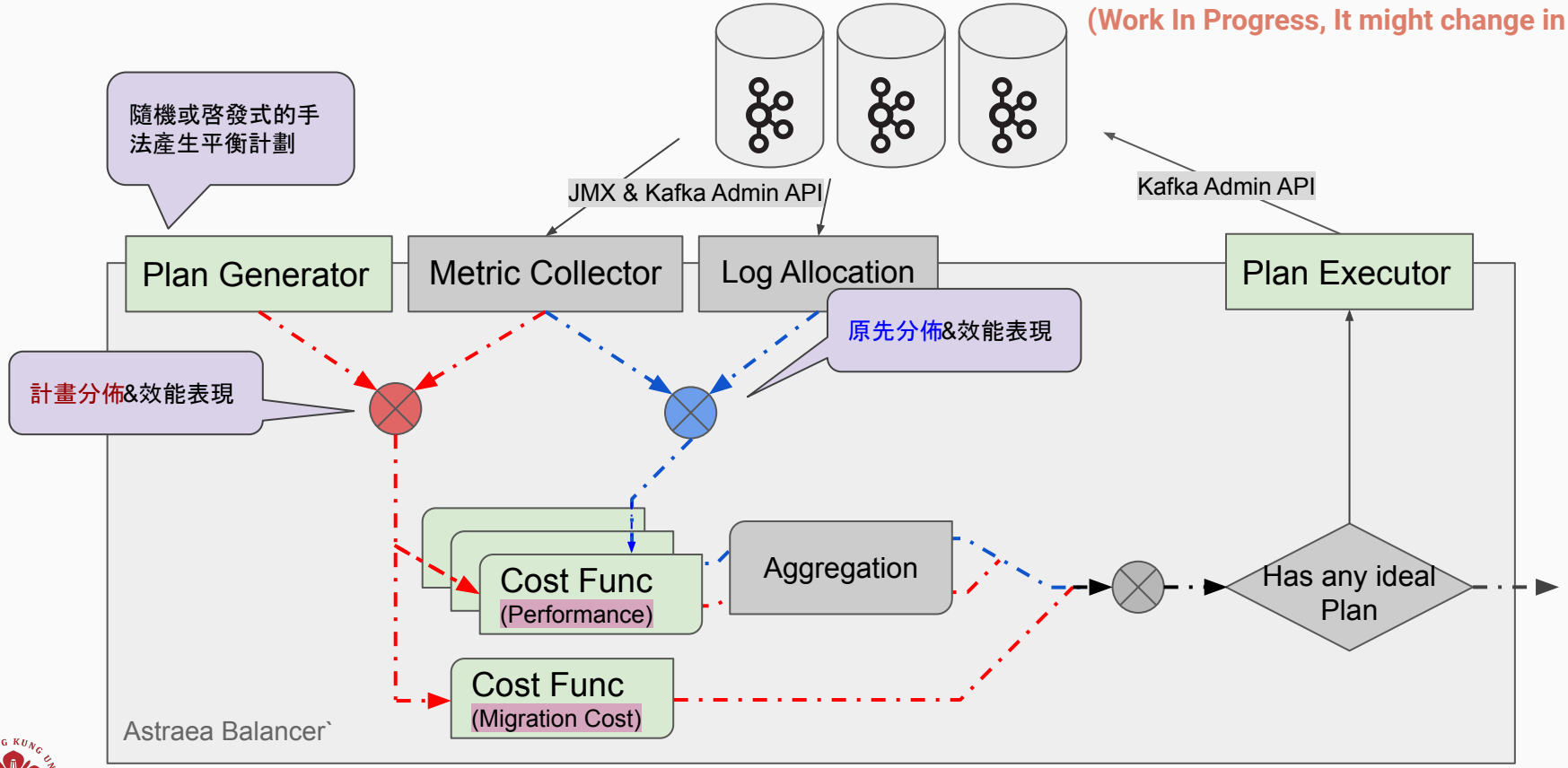
# 簡化版 Astraea Balancer 運作流程



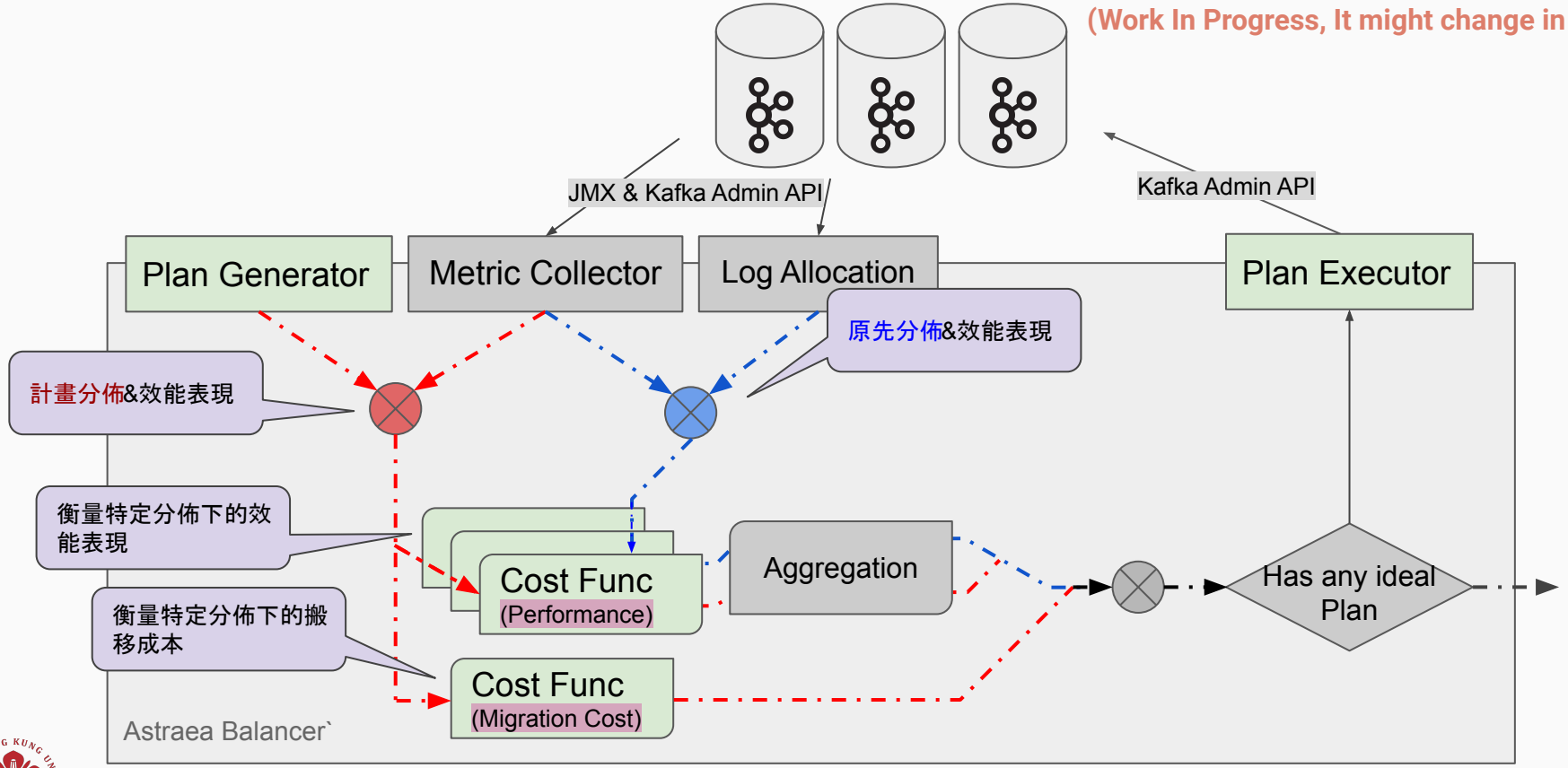
# 負載平衡計畫的生成


(Work In Progress, It might change in future)
獲得當前叢集狀態
和效能表現
JMX & Kafka Admin API
Kafka Admin API
Plan Generator
Metric Collector
Log Allocation
Plan Executor
Cost Func
(Performance)
Aggregation
Has any ideal
Plan
Cost Func
(Migration Cost)
Astraea Balancer`

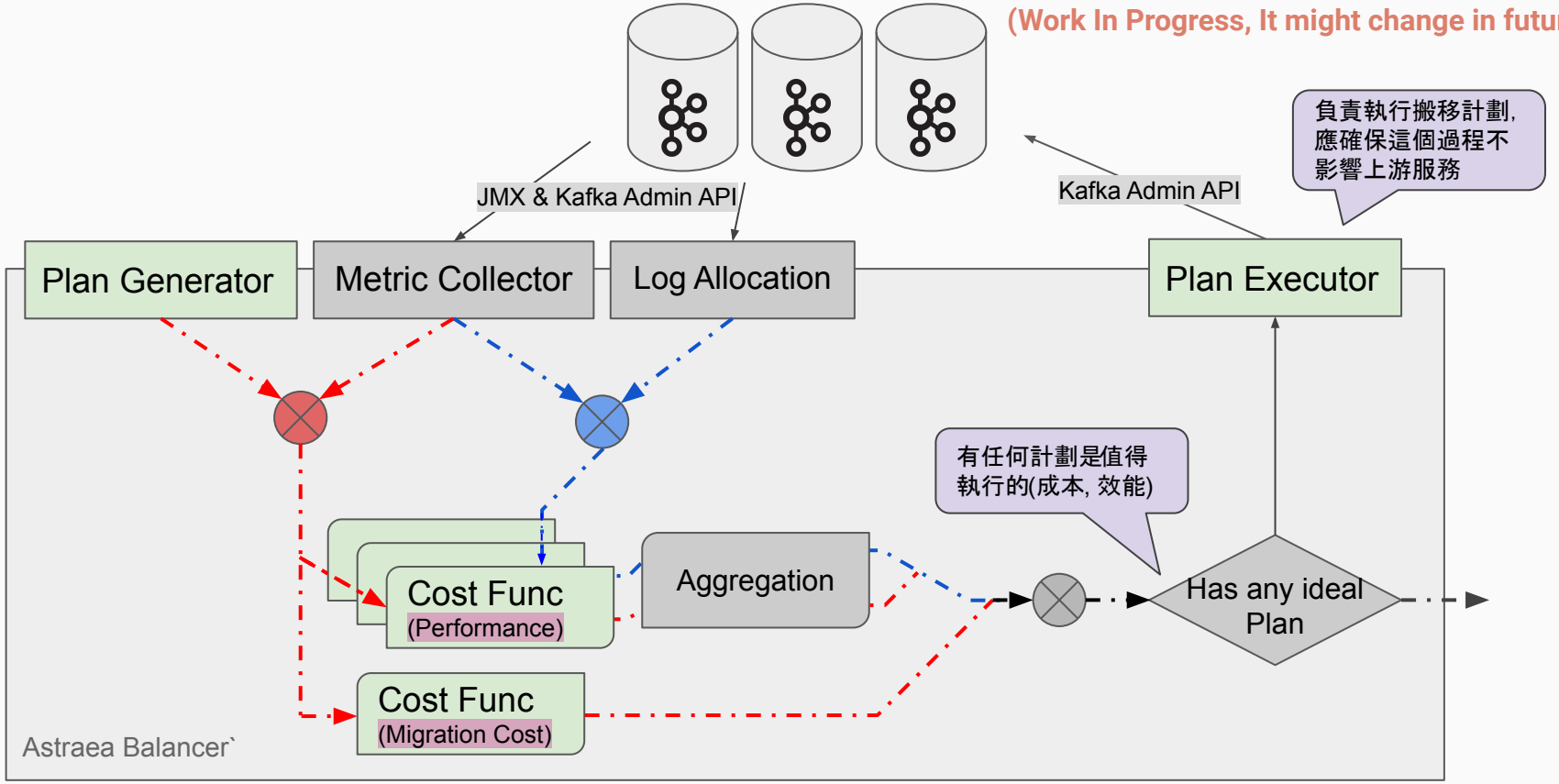
# 負載平衡計劃的衡量(平衡 & 成本)



# 負載平衡計劃的衡量(平衡 & 成本)



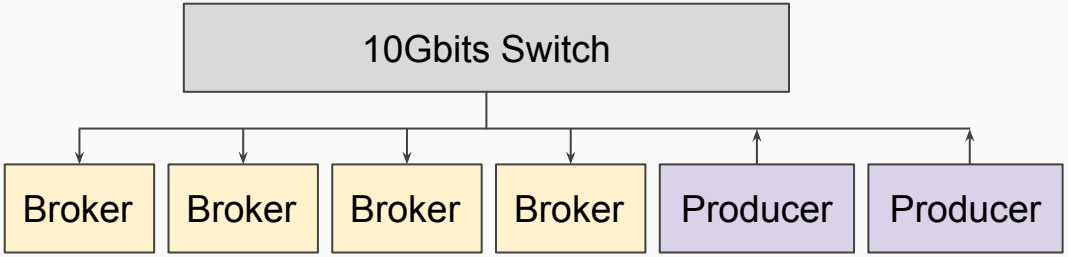
# 負載平衡計劃的執行


(Work In Progress, It might change in future)
JMX & Kafka Admin API
Kafka Admin API
負責執行搬移計劃, 應確保這個過程不影響上游服務
Plan Generator
Metric Collector
Log Allocation
Plan Executor
有任何計劃是值得執行的(成本, 效能)
Cost Func
(Performance)
Aggregation
Has any ideal Plan
Cost Func
(Migration Cost)
Astraea Balancer`

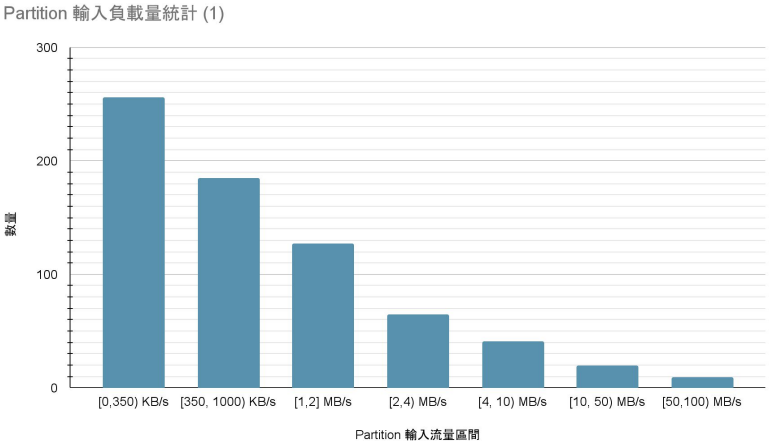
# 實驗設計

- 平衡目標
  - Produce 流量
- 硬體環境
  - 網路環境見圖片0
  - 硬體資訊見 https://github.com/skiptests/astraea/issues/130
- 情境

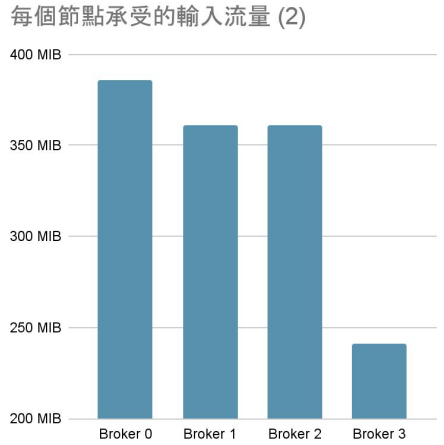
| | |
|---|---|
| Topics | 95 |
| Partitions | 803 |
| 分區負載量 | (見圖表1) |
| 節點輸入負載 | (見圖表2) |




(圖片0) 網路環境




(圖表1) Partiiton 輸入負載流量分佈




(圖表2) 節點負載