Kafka 優化的最後一哩路

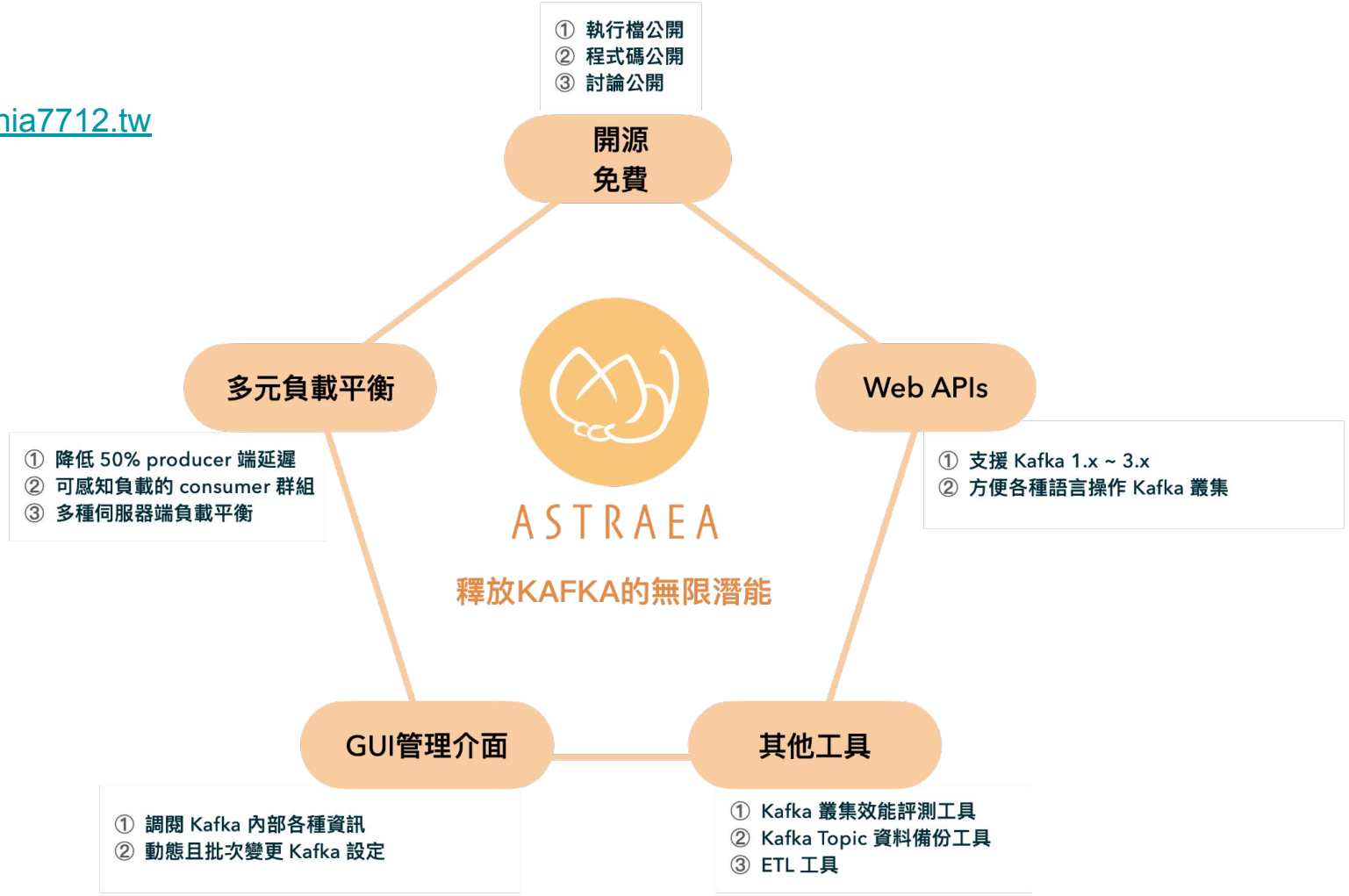
演講者 - 蔡嘉平, 李政憲, 方崢泓

Nov 2022 @ DataCon.TW

Astraea 專案 Github

簡報連結

www.chia7712.tw



介紹

- 講者
  - 李政憲, 方竫泓
  - 成功大學資訊工程所
    分散式系統實驗室的研究生
  - 研究 Kafka 負載平衡議題
- 內容
  - 前半段：Kafka 節點端優化
  - 後半段：Kafka 發送端優化

- Astraea 核心開發者：(Sorted by unicode order)
  孫祥鈞, 方竫泓, 李兆恆, 李宜桓,
  李政憲, 王懿宸, 蔡嘉平, 蕭宏章,
  鄧智懋, 陳嘉晟, 魏連興

Kafka 優化的最後一哩路

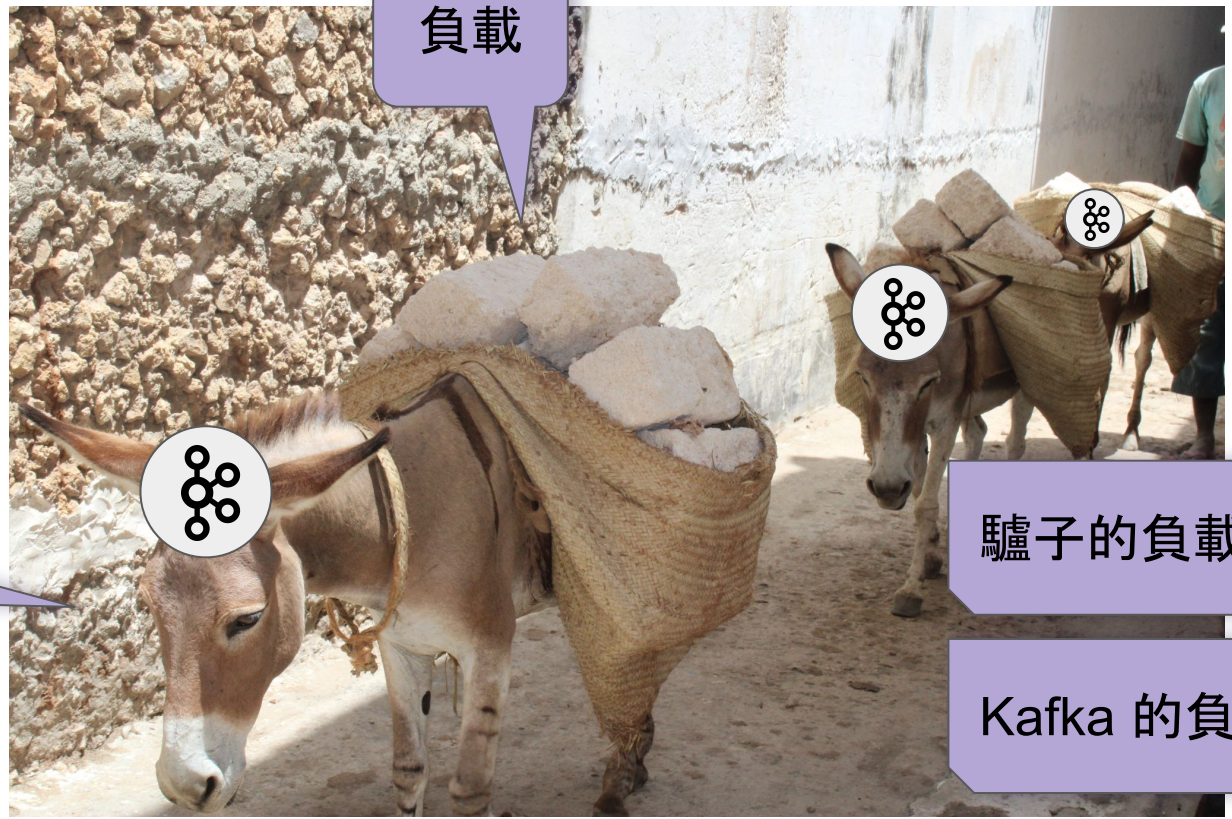
節點端負載優化

- 負載優化問題？
- 我們的解法 Astraea Balancer

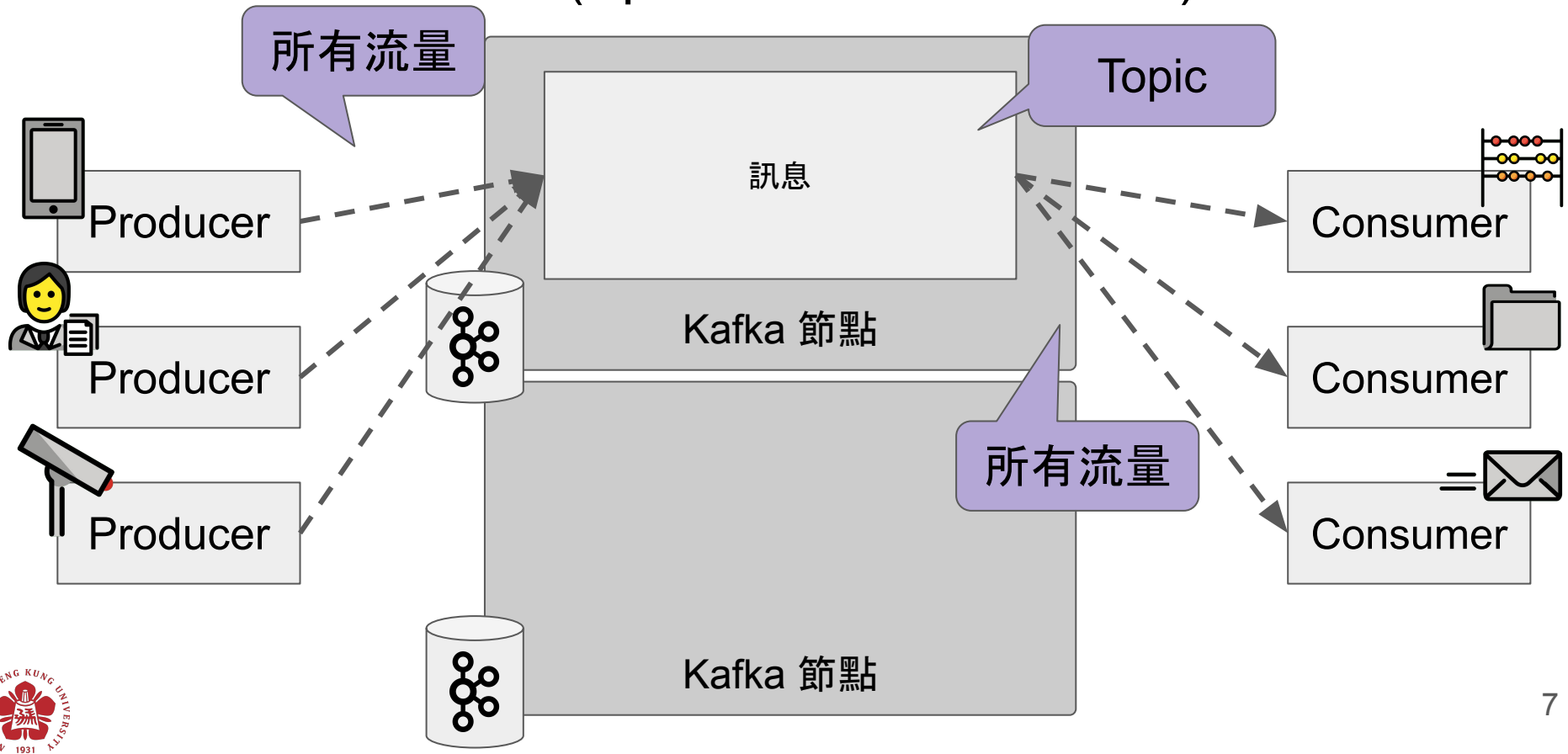
對於 Kafka 優化，我們能做得更好

- 優化
  - 維護者角度：資源使用最大化，減少不必要的設備成本。
  - 使用者角度：服務運作穩定。
- 優化手法
  - 程式實作優化
  - 參數優化
  - **負載優化**
- Kafka 內建的負載分配機制較為簡單
  - 這邊有很大的優化空間！

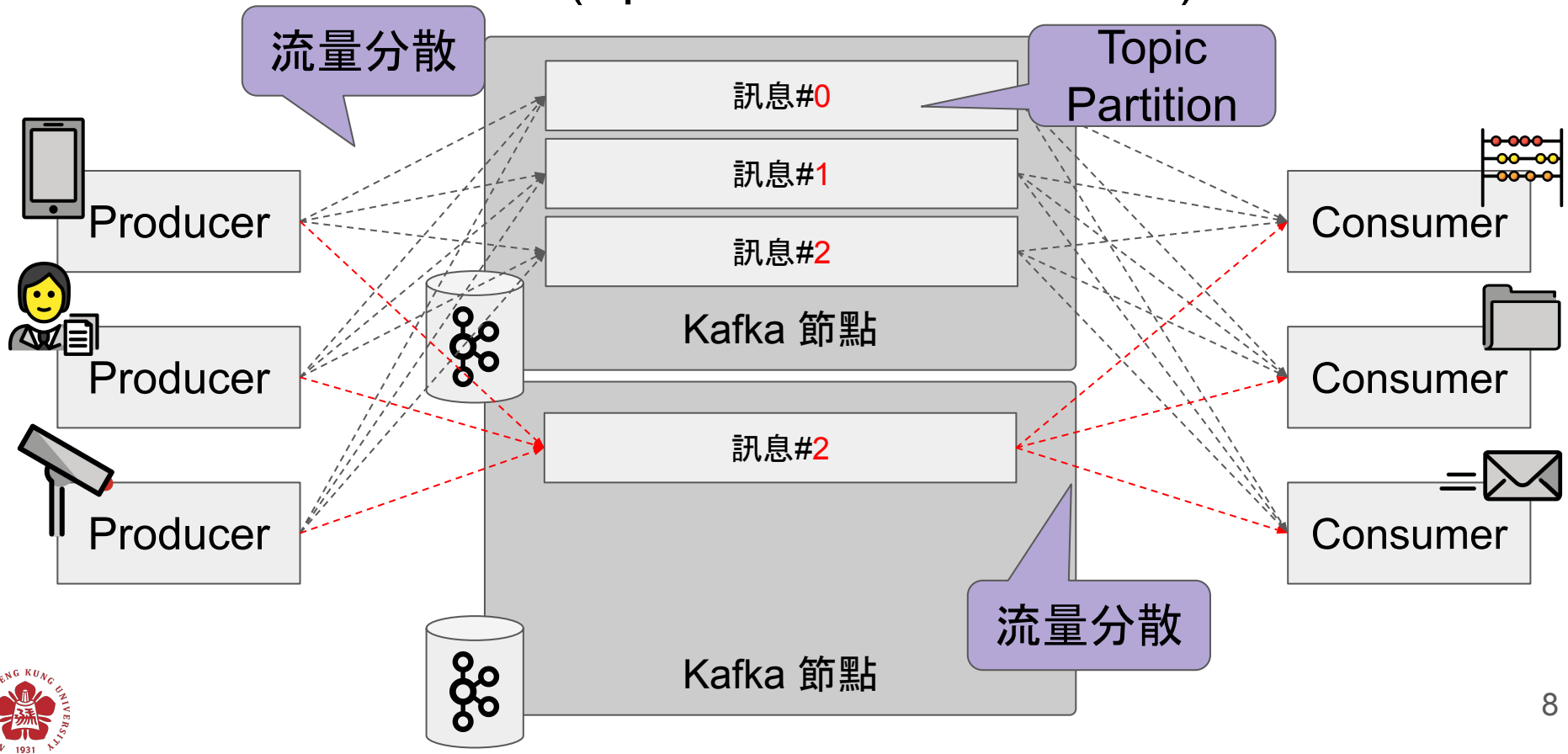
所以負載是什麼



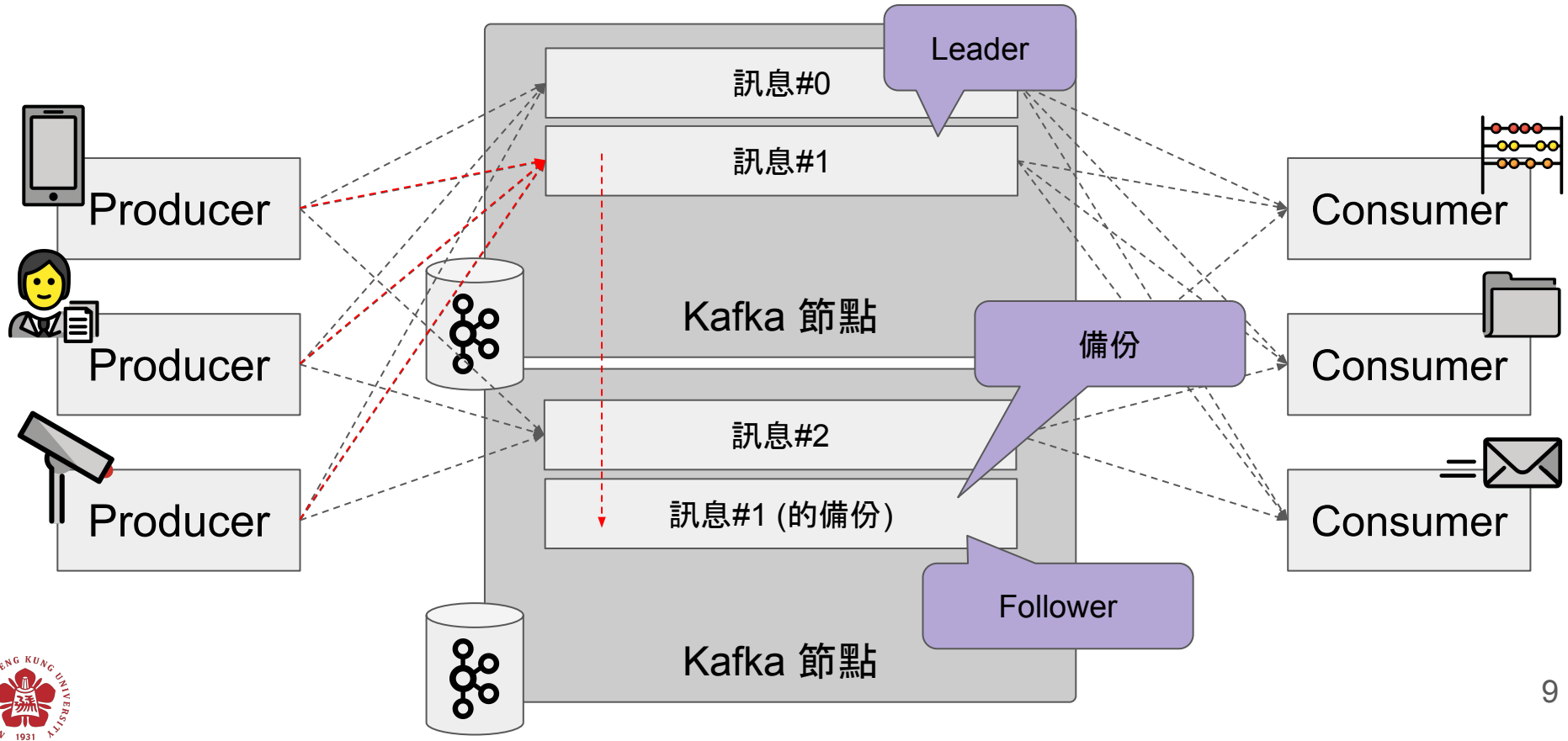
Kafka 的負載是什麼 (Apache Kafka 系統模型)



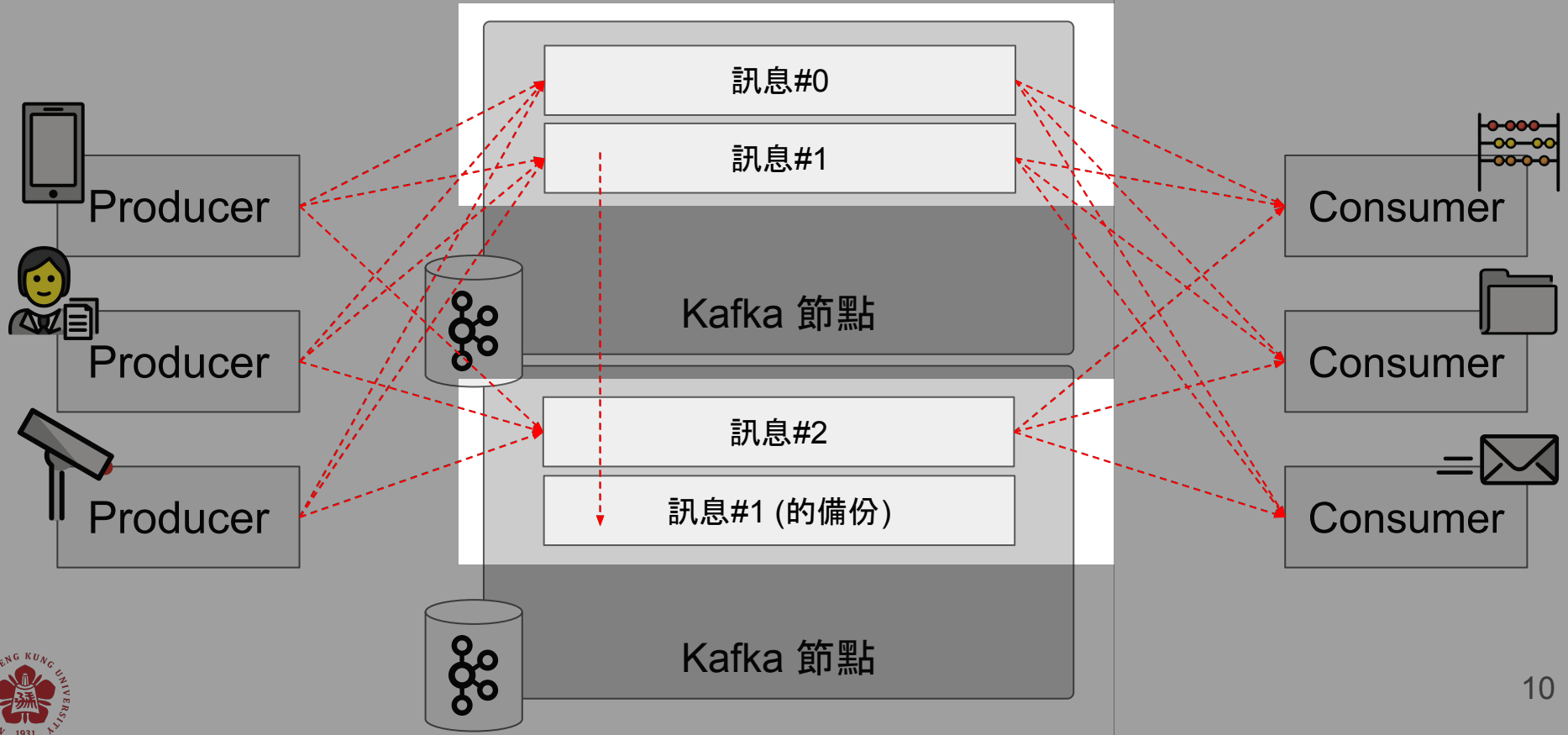
Kafka 的負載是什麼 (Apache Kafka 系統模型)



Kafka 的負載是什麼 (Apache Kafka 系統模型)



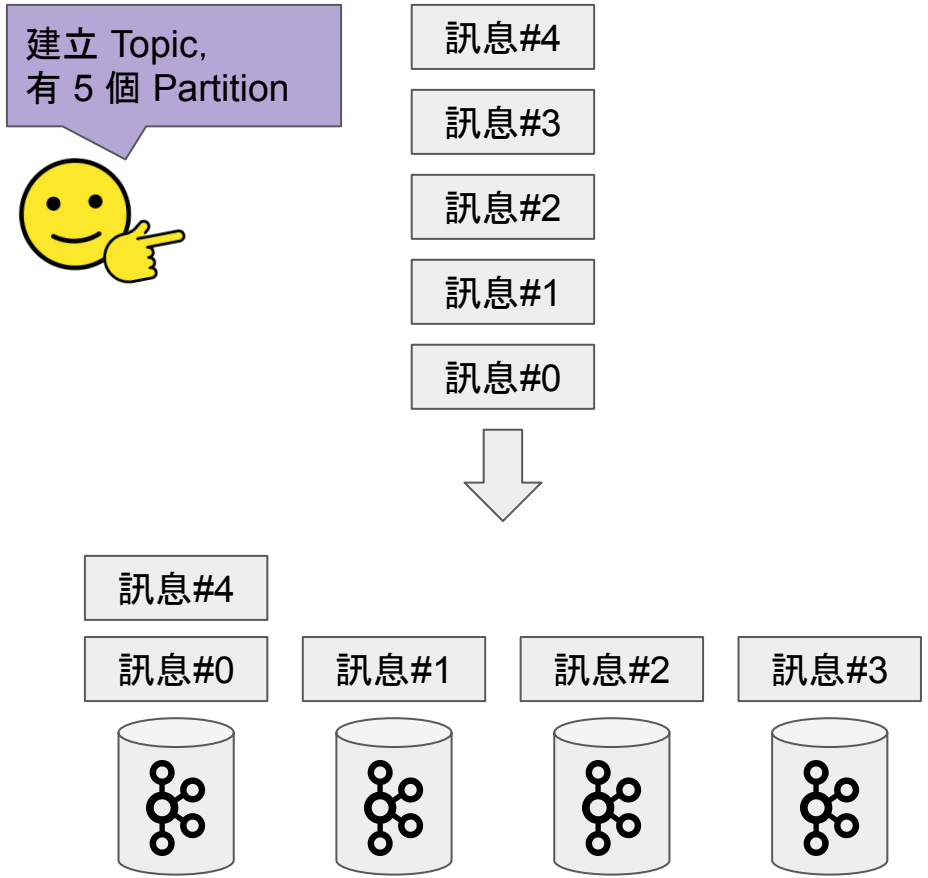
所以 Kafka 的負載是...



Kafka 的負載是 Partition

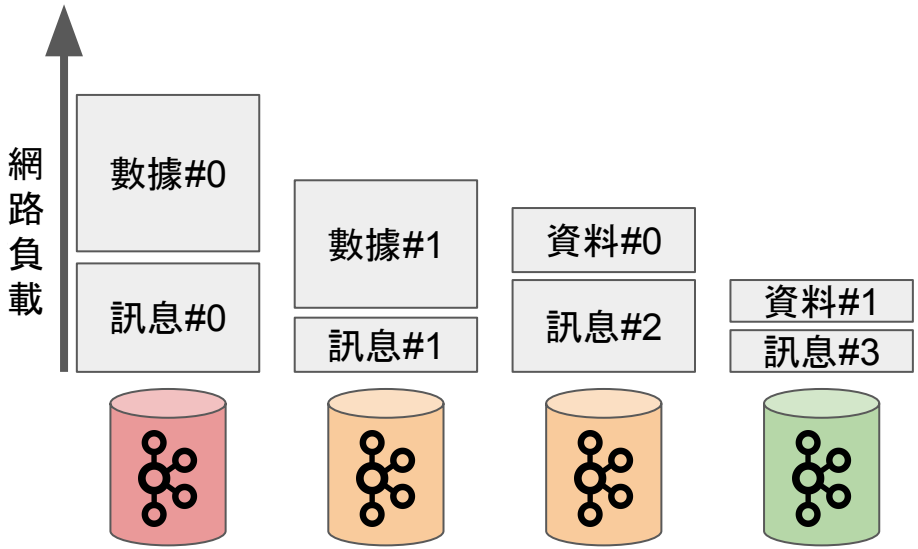
天然的 Kafka 負載分配機制

- Kafka 簡單負載分配機制
  - 有很大的優化空間！
- 負載位置的決定
  - 哪個 Partition 給哪個節點管理
- 負載的擺放方法
  - Round-Robin
- Round-Robin 背後的目的
  - 嘗試讓節點的 Partition 數量平均
  - 什麼情況下這個手法會不可靠？



自然造成的負載不平衡 - Skewed Load

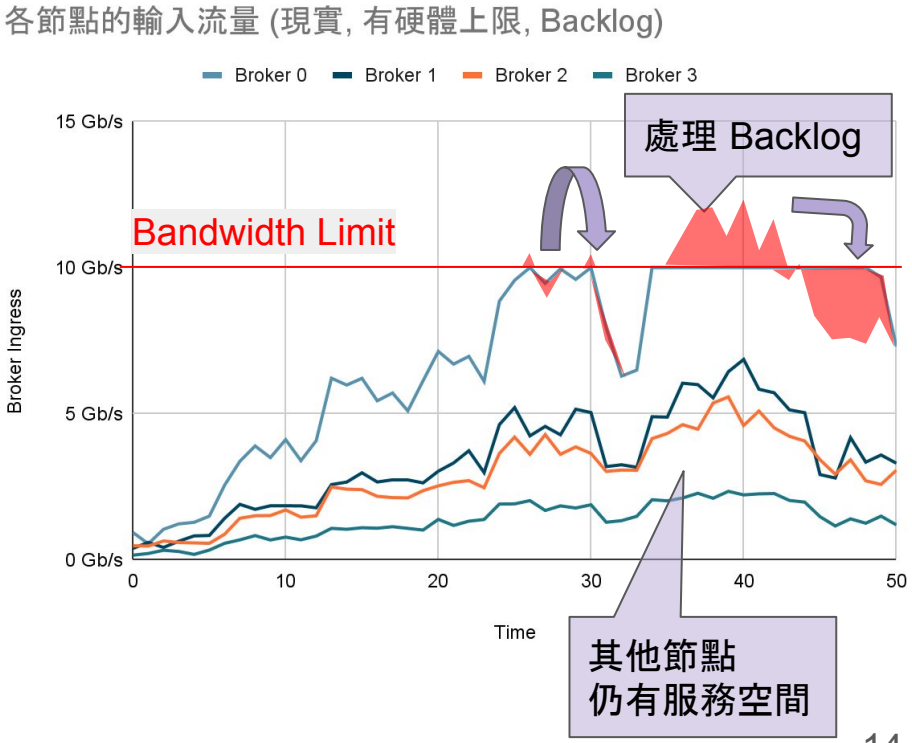
- Round-Robin 負載分配方法
  - 平均分配的直覺何時不可靠
- 如果 Partition 間有 Skewed Load…
  - Partition 負載大小不一
  - 負載不平衡



Partition 負載大小不一

自然造成的負載不平衡 - Skewed Load

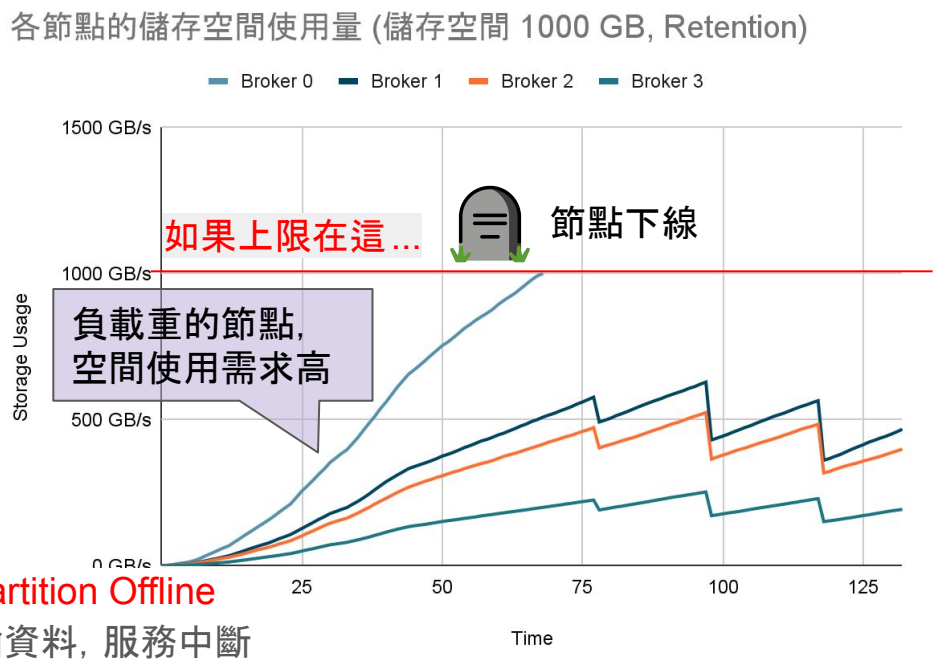
- Round-Robin 負載分配方法
  - 平均分配的直覺何時不可靠
- 如果 Partition 間有 Skewed Load…
  - Partition 負載大小不一
  - 負載不平衡
- 後果
  - 尖峰時倒霉的節點會過載
    - 請求要排隊，Client 應用延遲變高
    - 資源競爭，Client 資料傳輸瓶頸



自然造成的負載不平衡 - Skewed Load

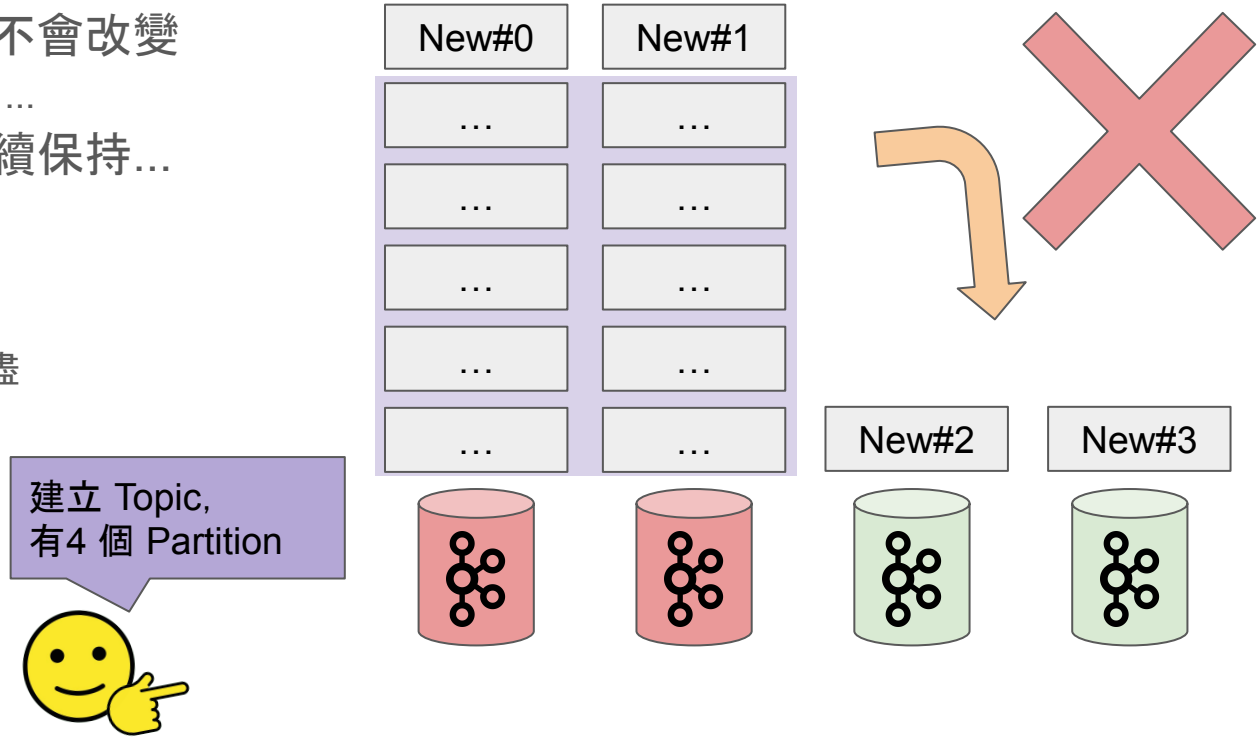
Space
5000 GB -> 1000 GB

- Round-Robin 負載分配方法
  - 平均分配的直覺何時不可靠
- 如果 Partition 間有 Skewed Load…
  - Partition 負載大小不一
  - 負載不平衡
- 後果
  - 尖峰時倒霉的節點會過載
    - 請求要排隊，Client 應用延遲變高
    - 資源競爭，Client 資料傳輸瓶頸
  - 倒霉節點的儲存空間消耗速度快
    - 節點儲存空間用盡 = 節點下線 or Partition Offline
    - (如無其他在線備份) Client 無法傳輸資料，服務中斷



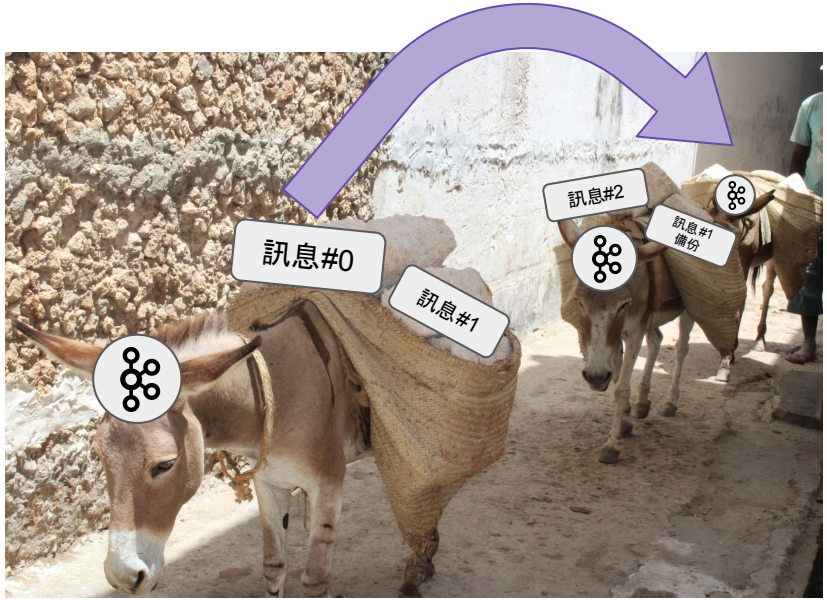
自然造成的負載不平衡 - 節點擴充後

- 負載的位置決定後不會改變
  - 擴充節點時要注意 ...
- 舊節點的負載會繼續保持...
  - 過載問題會衍生
    - 延遲提高
    - 傳輸瓶頸
    - 儲存空間用盡



負載不平衡，如何優化？

- Kafka 的負載其實能夠被轉移
  - Java Admin#alterPartitionReassignments
  - Shell Script bin/kafka-reassign-partitions.sh
- 嘗試用負載轉移來解決問題!
  - 負載平衡：每個節點分工平均
  - 意外避免：避免底下儲存空間用盡
  - 異質系統：性能好的節點負擔較多負載
  - QoS：重要的 Topics 放在 SSD 的儲存空間



優化問題的定義

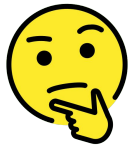
- 我們要來解決 Kafka 負載優化問題
- 開始解問題前需要定義問題

找到 f(x) 的最小值

$$f(\vec{x}) = An + \sum_{i=1}^{n}[x_i^2 - Acos(2\pi x_i)]$$

限制

$$A = 10$$
$$-5.12 \leq x_i \leq 5.12$$



描述問題(1/2) - 如何描述負載狀態

- 如何描述負載優化問題的**輸入**：負載的狀態

Topic/Partition	備份 Leader	備份 Follower
訊息#0	(節點0)	無
訊息#1	(節點0)	(節點1)
訊息#2	(節點1)	無

描述問題(1/2) - 如何描述負載狀態

- 如何描述負載優化問題的**輸入**：負載的狀態

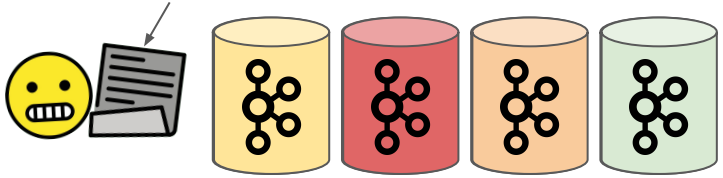
Just a Bunch of Disk (JBOD)

Topic/Partition	備份 Leader	備份 Follower
訊息#0	(節點0, /ssd1)	無
訊息#1	(節點0, /ssd2)	(節點1, /hdd1)
訊息#2	(節點1, /ssd1)	無

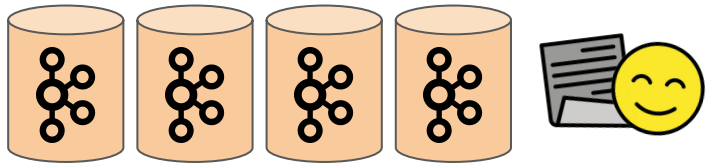
描述問題(2/2) - 負載優化

Before

欲優化的目標



After

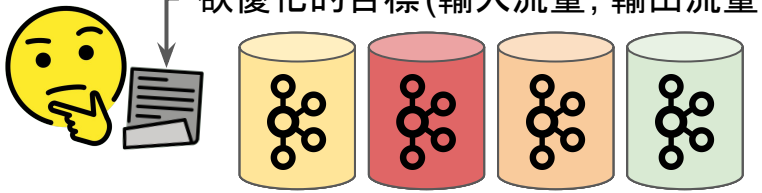


Topic/Partition	備份 Leader	備份 Follower
…	(ax0, ay0)	(ai0, aj0)
…	(bx0, by0)	(bi0, bj0)
…	(cx0, cy0)	(ci0, cj0)
…	(dx0, dy0)	(di0, dj0)

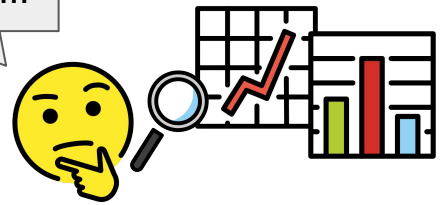
Topic/Partition	備份 Leader	備份 Follower
…	(ax1, ay1)	(ai1, aj1)
…	(bx1, by1)	(bi1, bj1)
…	(cx1, cy1)	(ci1, cj1)
…	(dx1, dy1)	(di1, dj1)

工人智慧: 手動轉移負載

欲優化的目標(輸入流量, 輸出流量, 儲存空間足夠)



過去的叢集效能 ...



要移動到哪個節點

(JBOD Enabled)
要放到哪個資料夾

哪個 Partition 要移動...

Topic/Partition	備份 Leader	備份 Follower	
訊息#0	(broker0, /ssd2)	(broker1, /ssd1),	(broker3, /ssd2)
訊息#1	(broker1, /ssd1)	(broker2, /ssd1),	(broker0, /ssd1)
數據#0	(broker2, /ssd1)	(broker3, /ssd1)	(broker1, /ssd2)
	(	r1, /ssd2),	(broker2, /ssd1)

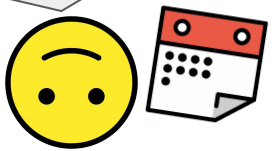
叢集隨著使用負載會劣化
某天這整個流程要重複一次

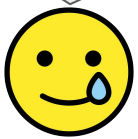
重複這個過程 n 次

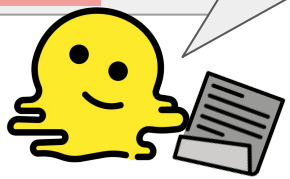
輸入 輸出

輸入 輸出

這些變更會不會影響其他我想優化的指標



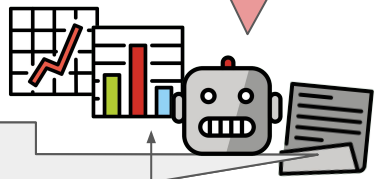


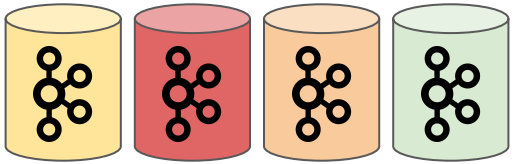


軟體自動化負載優化: Astraea Balancer

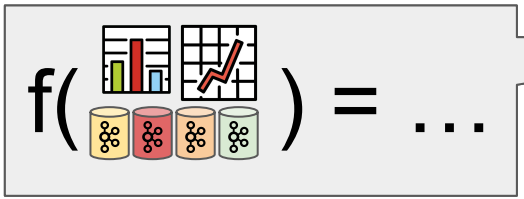
我想要...
1. 所有節點輸入負載平均
2. 所有節點輸出負載平均
3. 每個節點的 Leader 數平均
4. 重要的 Topic 放在 SSD 上
5. 儲存空間不會塞爆

50 分





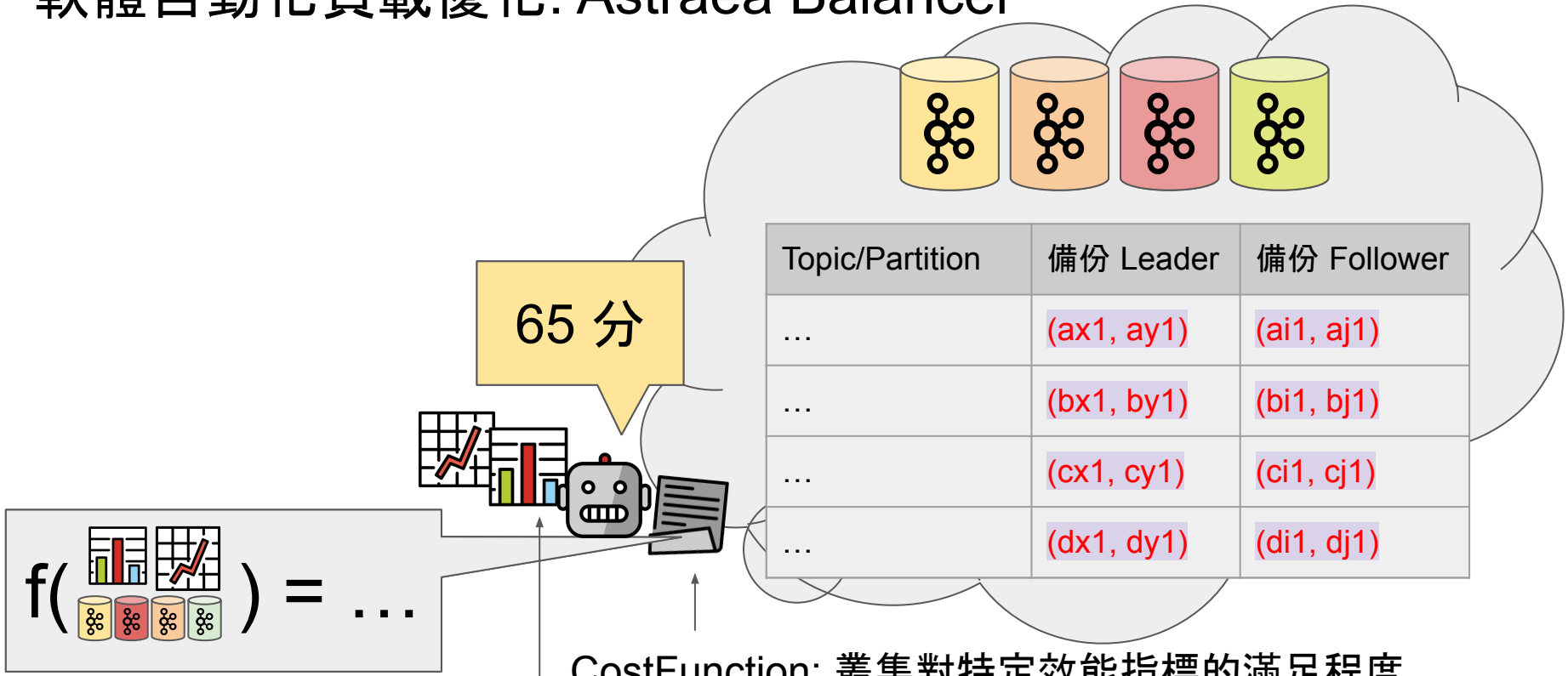
Topic/Partition	備份 Leader	備份 Follower
…	(ax0, ay0)	(ai0, aj0)
…	(bx0, by0)	(bi0, bj0)
…	(cx0, cy0)	(ci0, cj0)
…	(dx0, dy0)	(di0, dj0)



CostFunction: 叢集對特定效能指標的滿足程度

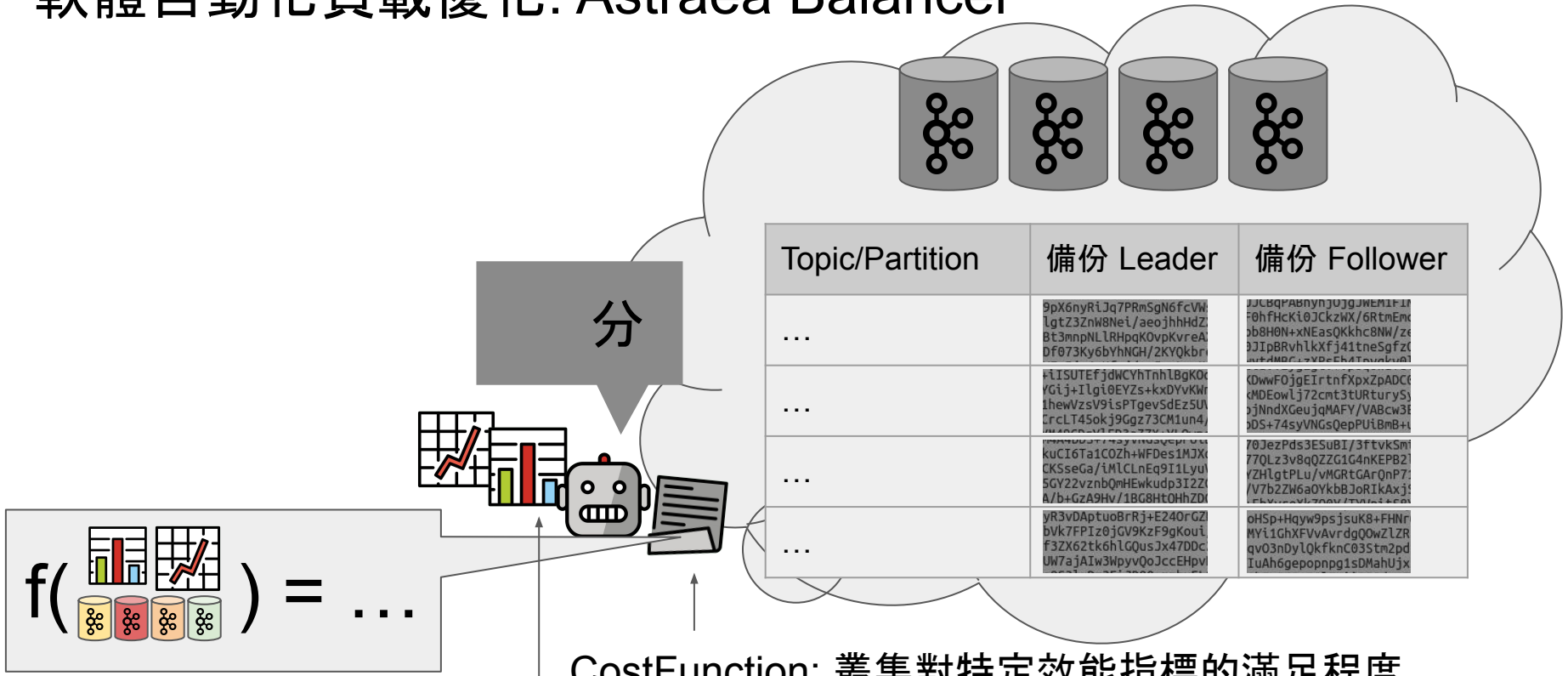
叢集過去的效能指標

軟體自動化負載優化: Astraea Balancer



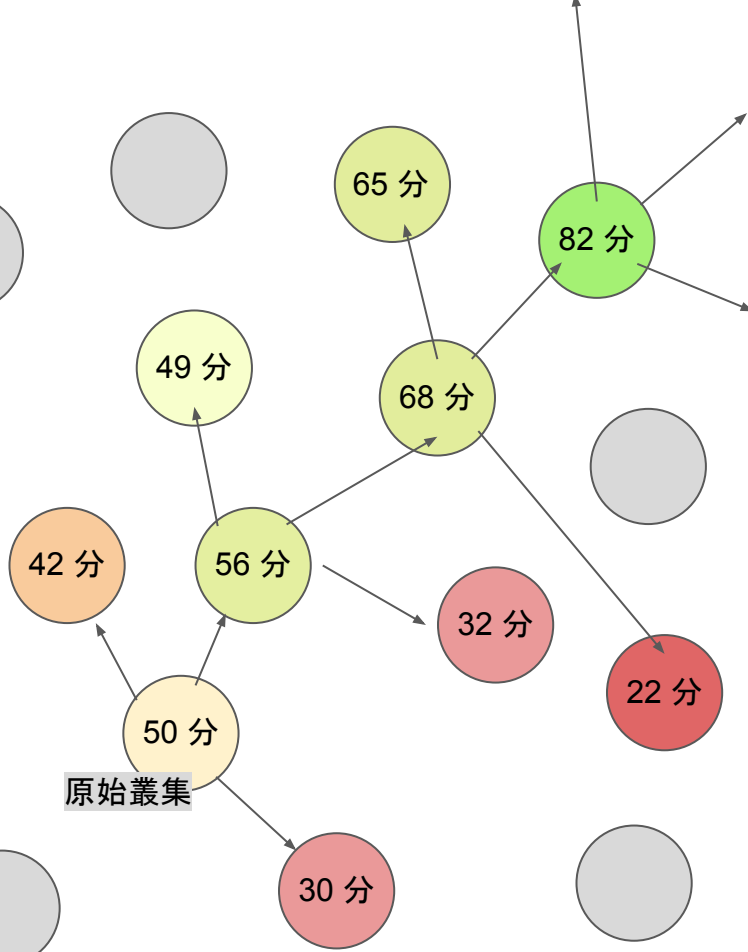
Topic/Partition	備份 Leader	備份 Follower
…	(ax1, ay1)	(ai1, aj1)
…	(bx1, by1)	(bi1, bj1)
…	(cx1, cy1)	(ci1, cj1)
…	(dx1, dy1)	(di1, dj1)

軟體自動化負載優化: Astraea Balancer



如何提出新負載分佈

- 尋找更好的叢集負載分配
  - 在解空間中搜尋
- 可能的解決方法
  - Random Search
  - Hill Climbing
  - Simulated Annealing
  - Evolutionary Algorithm
  - Swarm Optimization



```
curl -X POST http://localhost:8001/balancer \
     -H "Content-Type: application/json" \
     -d '{
   "timeout": "10s" ,
   "balancer": "org.astraea.common.balancer.algorithms.GreedyBalancer",
   "balancer-config": {
     "shuffle.plan.generator.min.step": "1",
     "shuffle.plan.generator.max.step": "5"
   },
   "costWeights": [ { "cost": "org.astraea.common.cost.ReplicaLeaderCost", "weight":  1} ],
   "max-migrated-leader": "10"
}'
```

計劃生成時間

演算法實作

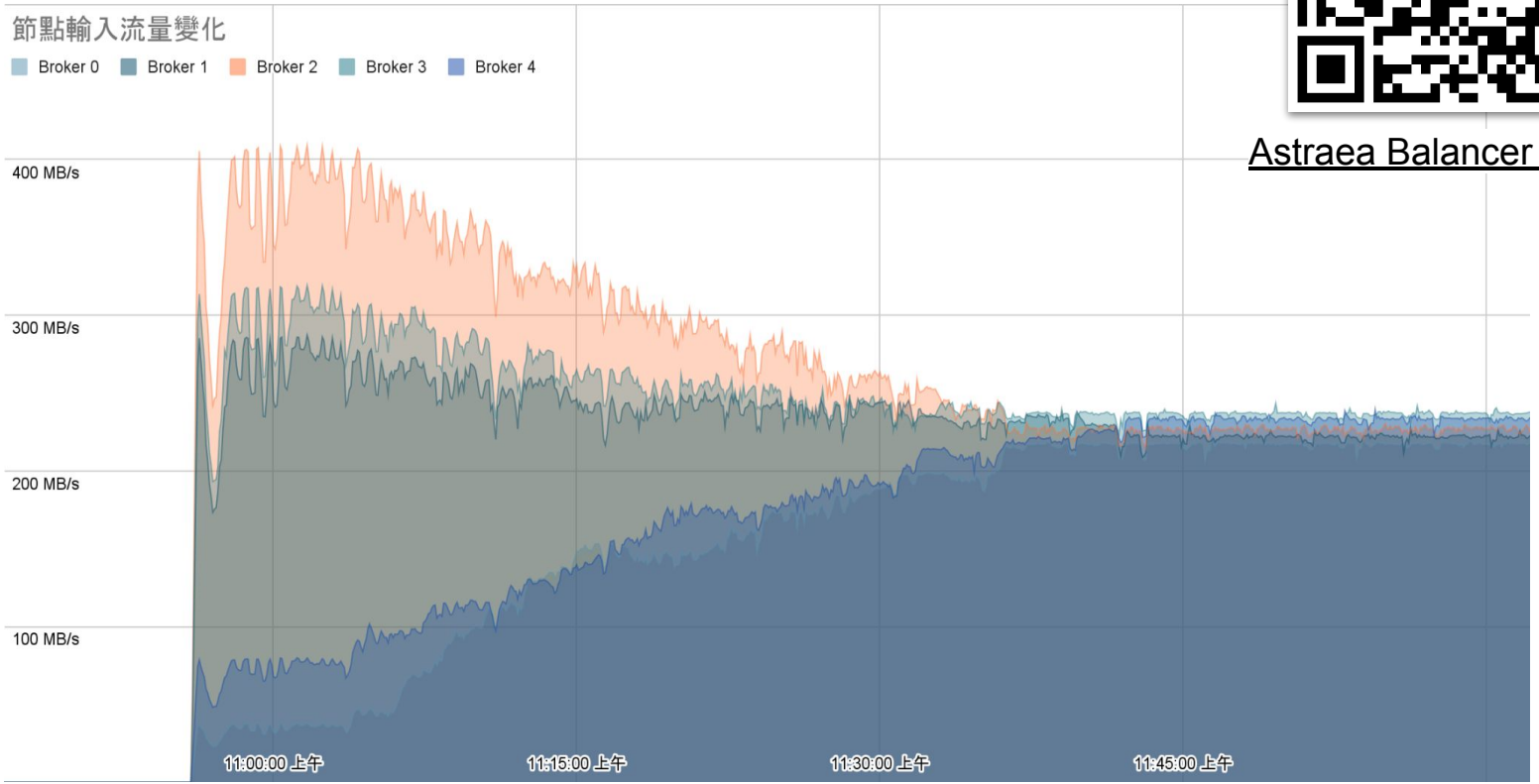
演算法參數

使用的 Cost Function

```
      after: [ { brokerId: 3, directory: /tmp/log-folder-2 } ]
    },
+----- 17 lines: {
+----- 17 lines: {
+----- 17 lines: {
+----- 17 lines: {
+----- 17 lines: {
+----- 17 lines: {
+----- 17 lines: {
+----- 17 lines: {
  ],
+---- 83 lines: "migrationCosts": [
}
}
```

計劃的細部改變

Astraea Balancer 實驗#2



Astraea Balancer 實驗#2

Summary

- Kafka 內建的負載分配機制較為簡單
  - Round-Robin
  - 負載擺放後不會移動
- 負載不平衡的危險
  - 節點過載
  - 資料儲存問題
- Astraea Balancer 框架
  - 客製化負載優化目標
  - 撈取 & 分析效能指標
  - 自動化優化計劃生成

Astraea Balancer 實驗#2

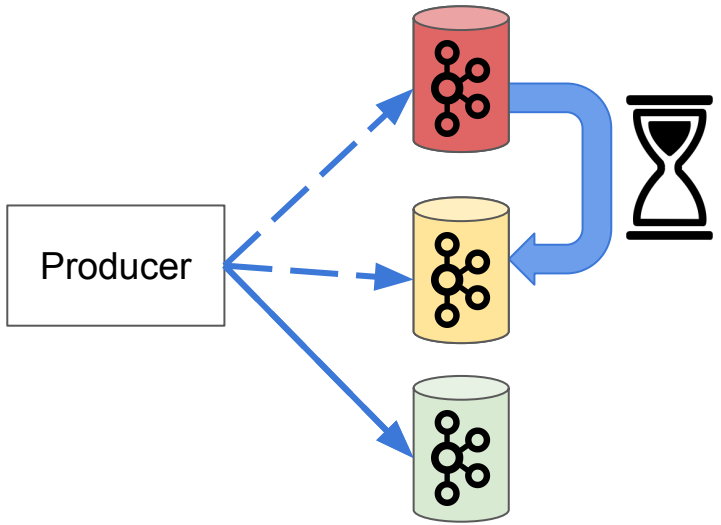
Kafka 優化的最後一哩路

Producer 端優化

取得叢集效能 -> 選擇發送目的

為何需要 Producer 端優化？

- Balancer 優點
  - **不需要** 動到應用端
- Balancer 無可避免
  - 搬移有成本：過去的訊息都要搬動
  - 搬移要時間：無法 **即時改善** 現況
- Producer 端優化
  - 訊息一次到位，盡量避免未來再次搬移
  - 即時反應，將訊息導向狀況較好的節點
- Producer 端優化限制
  - 需要應用端配合

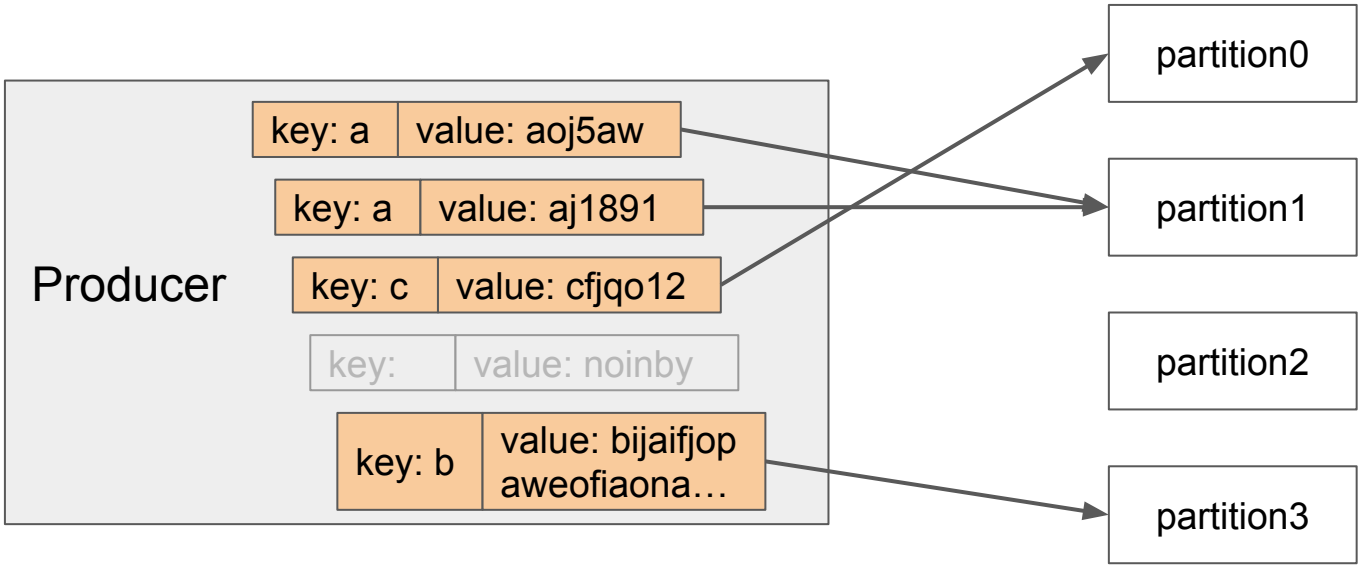


Partitioner

- 是一個選擇 partition 的邏輯
- ***Producer 不指定 partition 時, 才會呼叫 Partitioner***
- 是 Kafka 開出來的介面，供人實作並抽換

Kafka Partitioner 預設 (有 key)

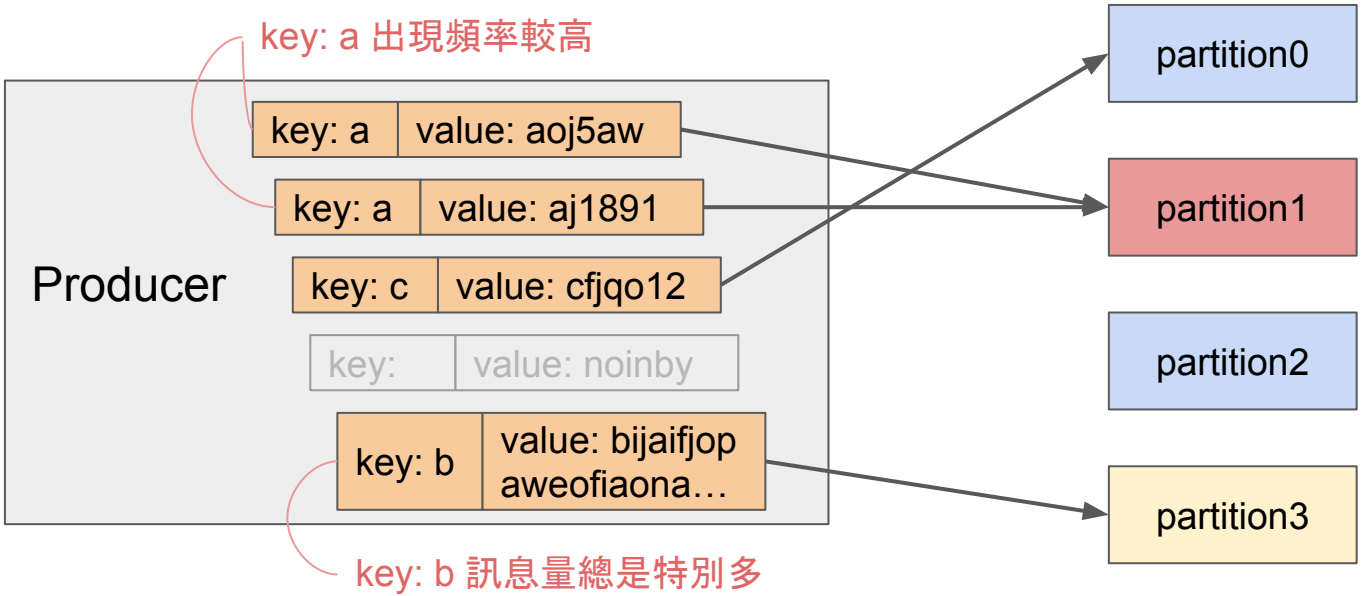
- 相同 key，到相同 partition



Kafka Partitioner 預設 (有 key)

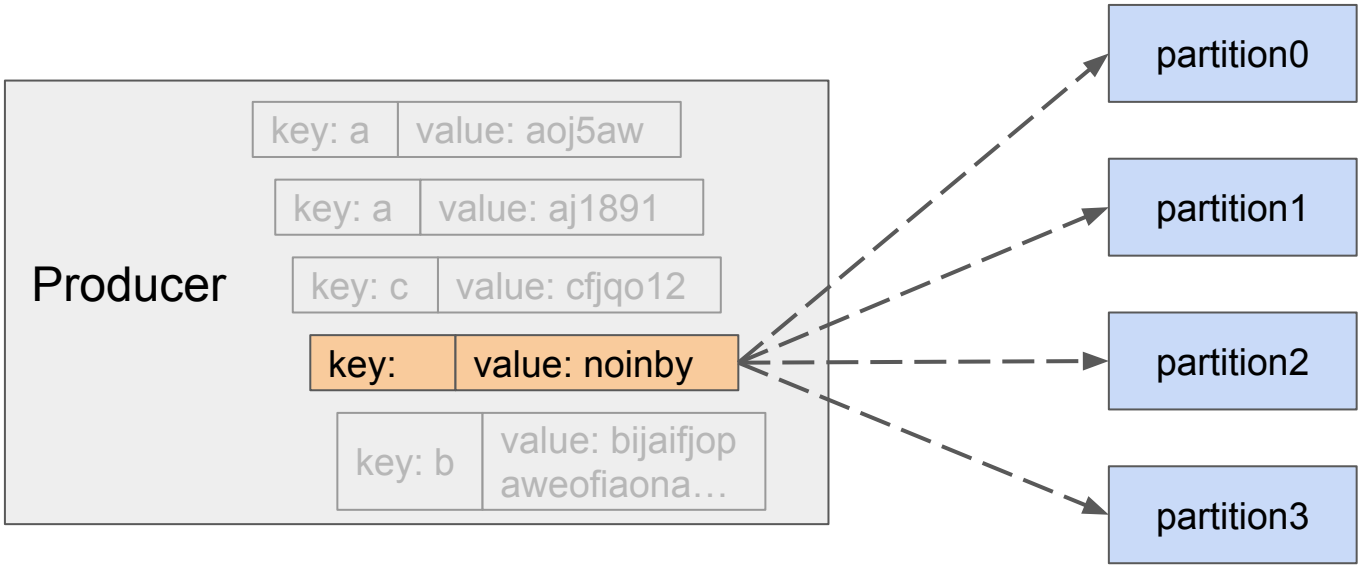
- 相同 key，到相同 partition

Skewed Partition Load!!



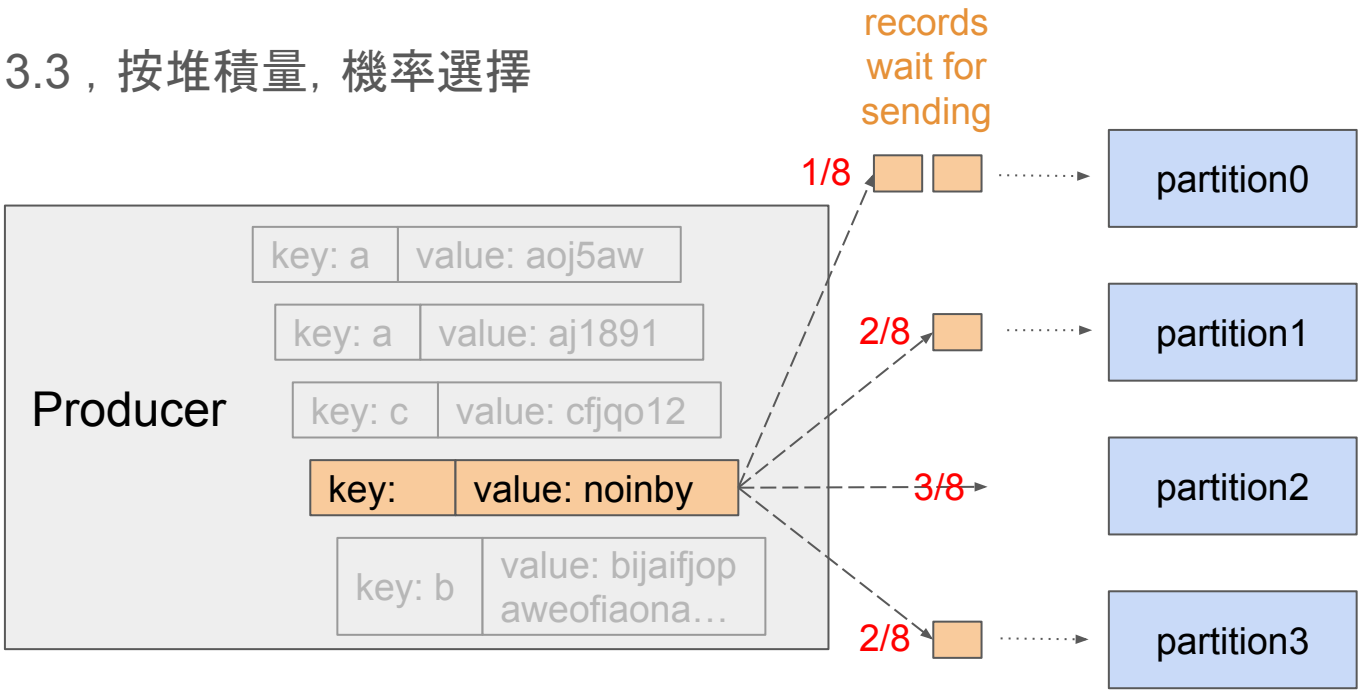
Kafka Partitioner 預設 (沒有 key)

- Kafka 3.2 前，平均選擇



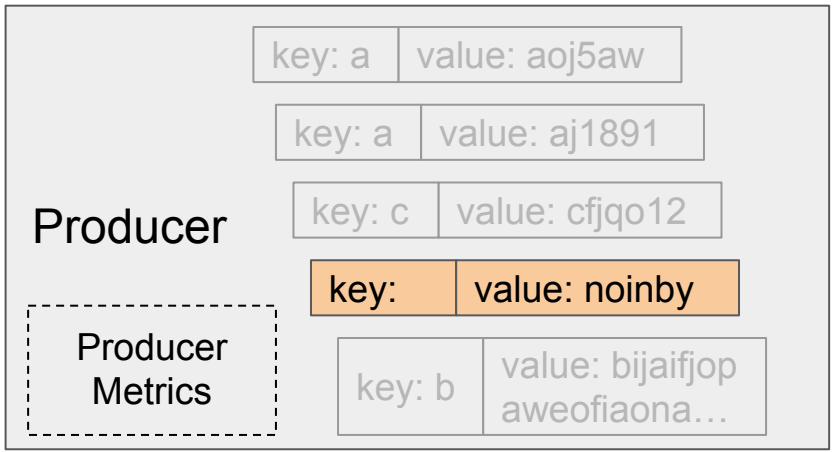
Kafka Partitioner 預設 (沒有 key)

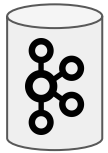
- Kafka 3.3，按堆積量，機率選擇



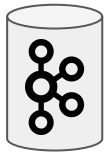
Astraea Dispatcher

- 抓取效能數據
- 效能數據代入 Cost Function
- 選擇 partition

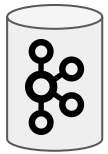




Network recv: 800MiB/sec
latency: 40ms
…



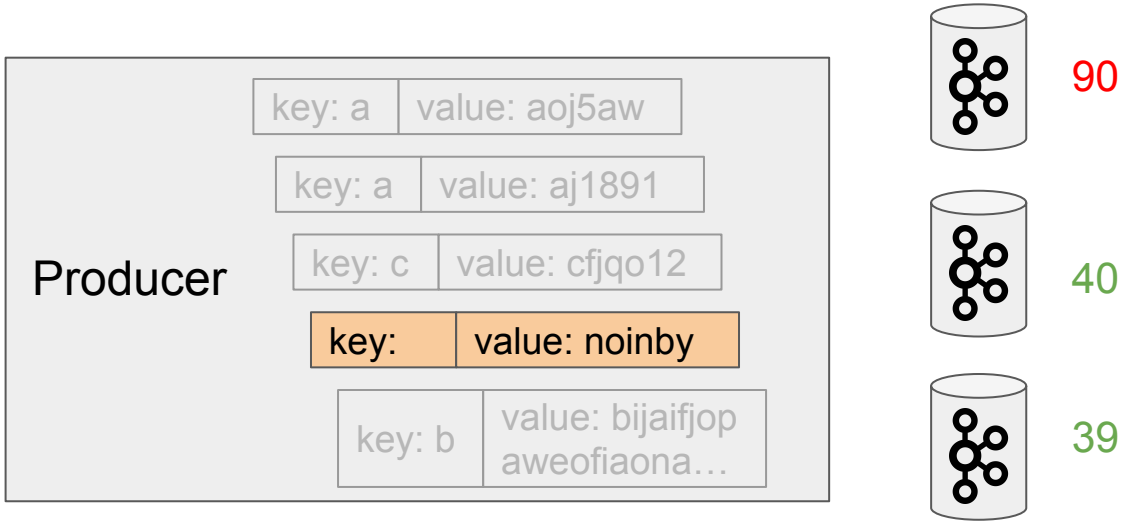
Network recv: 500MiB/sec
latency: 5ms
…



Network recv: 500MiB/sec
latency: 4ms
…

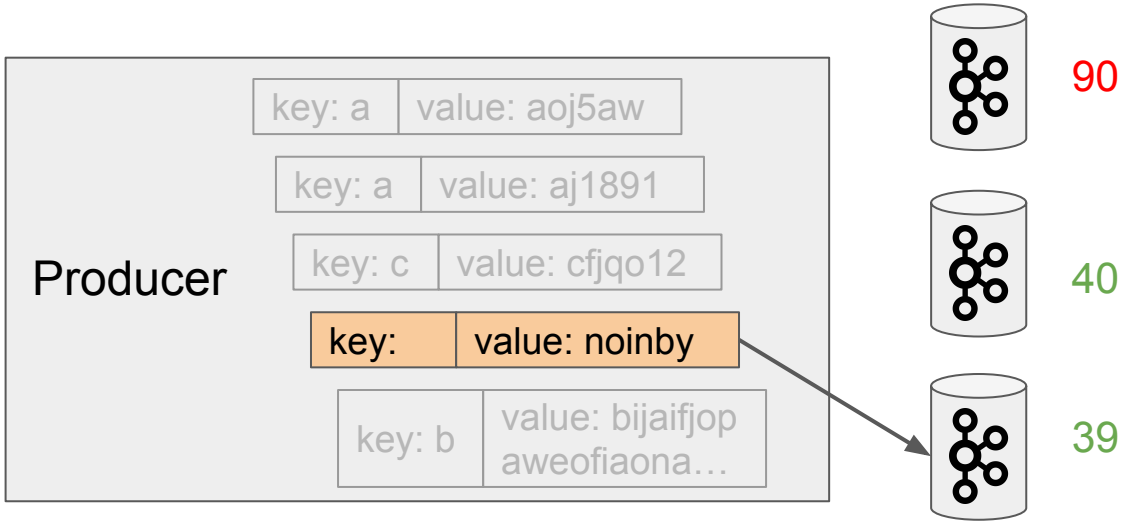
Astraea Dispatcher

- 抓取效能數據
- 效能數據代入 Cost Function
- 選擇 partition

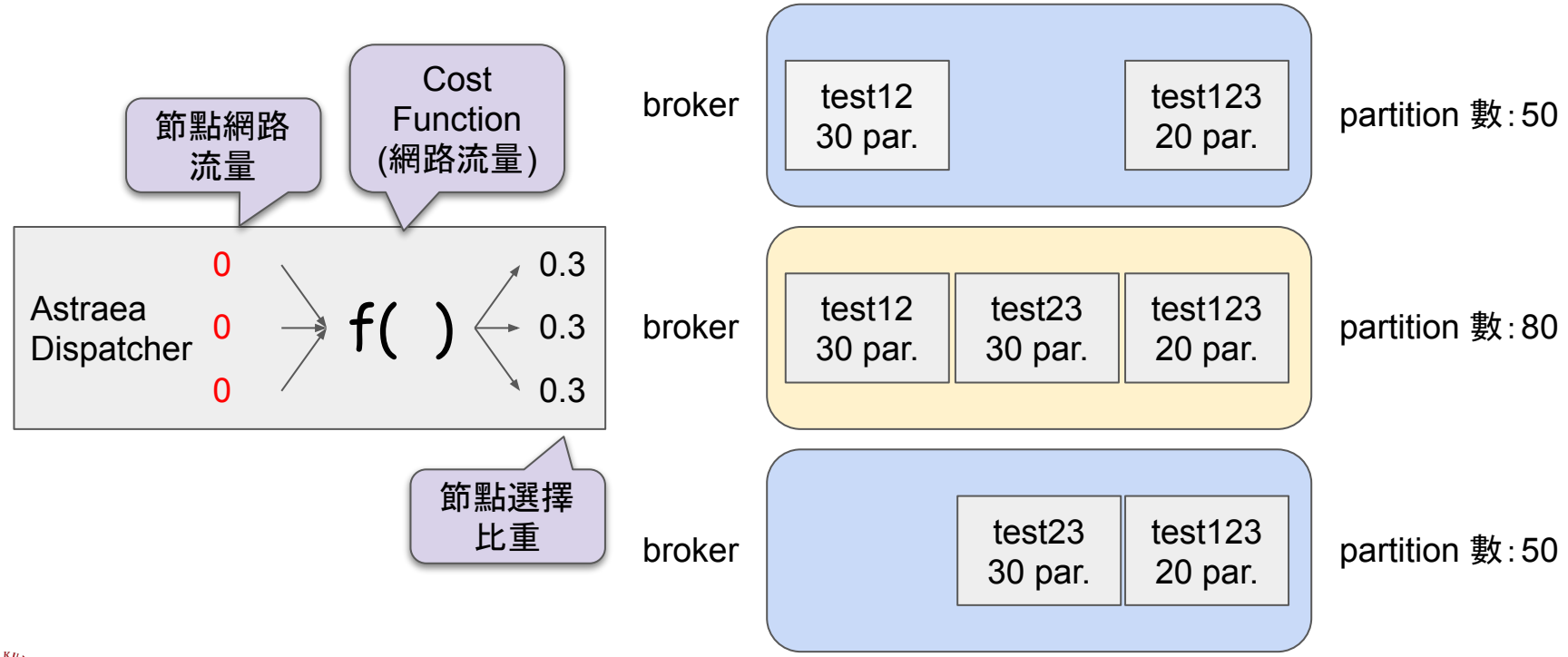


Astraea Dispatcher

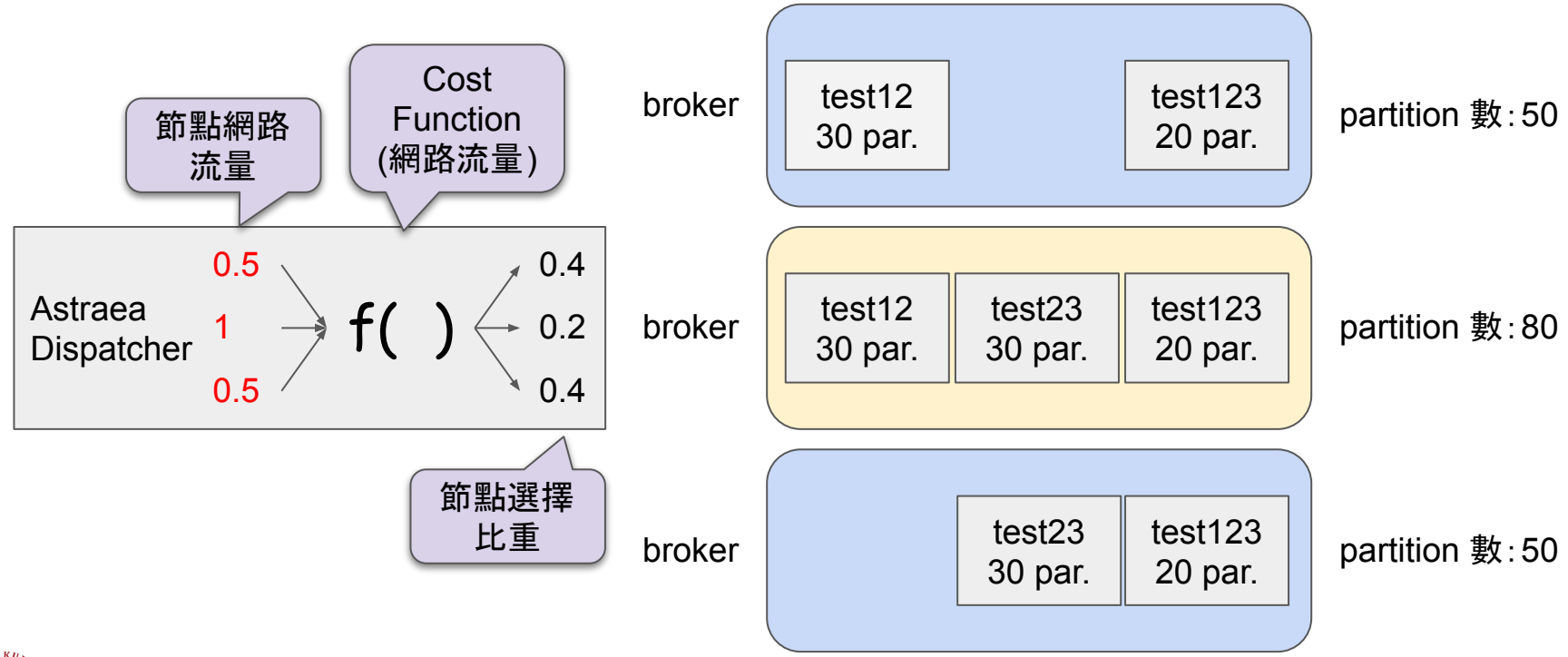
- 抓取效能數據
- 效能數據代入 Cost Function
- 選擇 partition



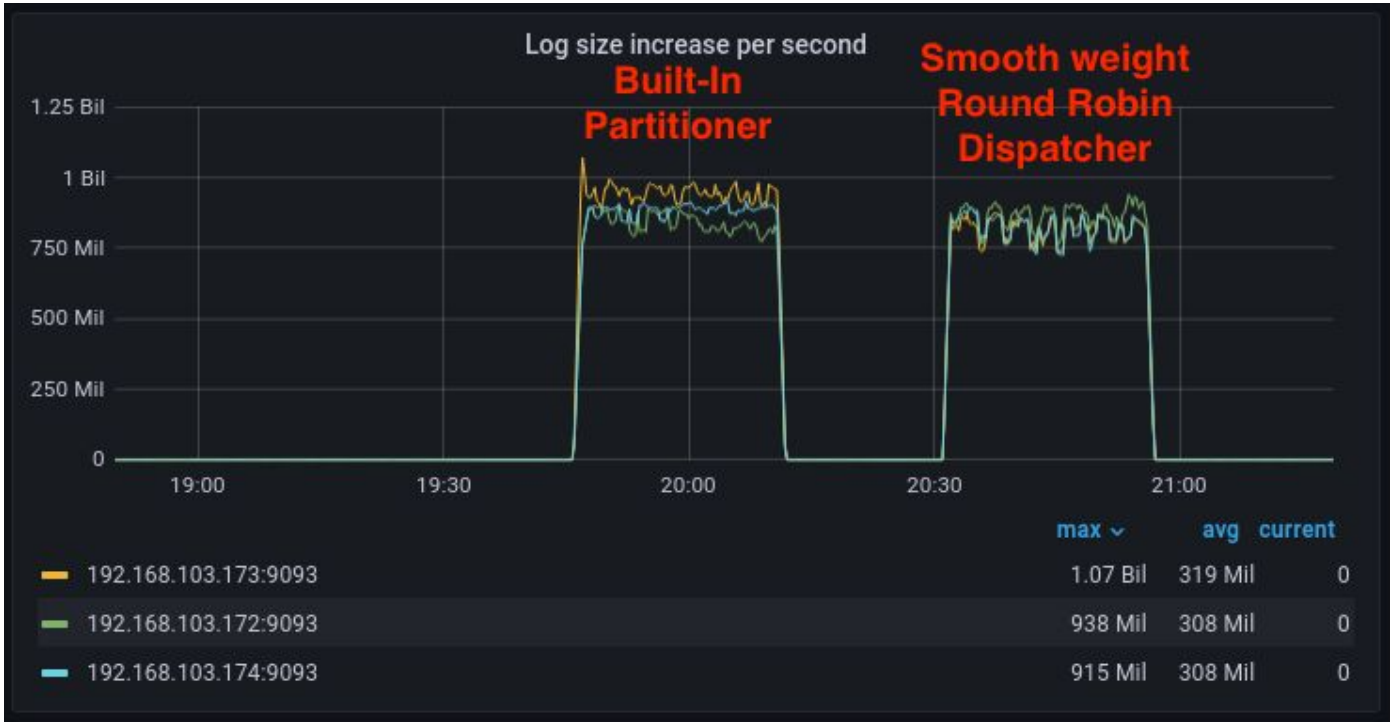
Astraea Dispatcher 例子



Astraea Dispatcher 例子



Astraea Dispatcher Experiment



實驗討論串

比較

	Sticky Partitioner	Built-In Partitioner	Astraea Partitioner
發送策略	平均選擇	根據發送延遲	根據叢集負載(可調整)
優勢	• 叢集 partition 分佈平衡時	• 考量發送延遲, 避開反應較慢的節點 • 可決定是否 ”同 key, 同 partition”	• 兼容 Kafka 版本 • 負載定義有彈性

Q&A

簡報連結

Astraea 專案 Github

Astraea Balancer 實驗#2

Kafka 優化的最後一哩路

講者：方竫泓、李政憲

簡報連結

講者介紹

講者：方竫泓、李政憲

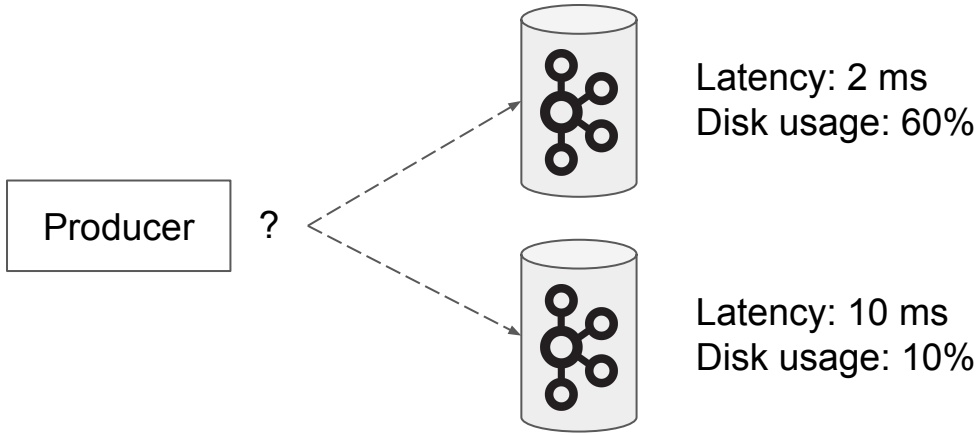
目前是成功大學資訊工程所分散式系統實驗室的研究生，正在做 Kafka 負載平衡議題，於 GitHub 開源這些工具 — Astraea。

Astraea 核心開發者：孫祥鈞, 方竫泓, 李宜桓, 李政憲, 王懿宸, 蔡嘉平, 蕭宏章, 鄧智懋, 陳嘉晟, 魏連興, 李兆恆

Astraea 專案 Github連結

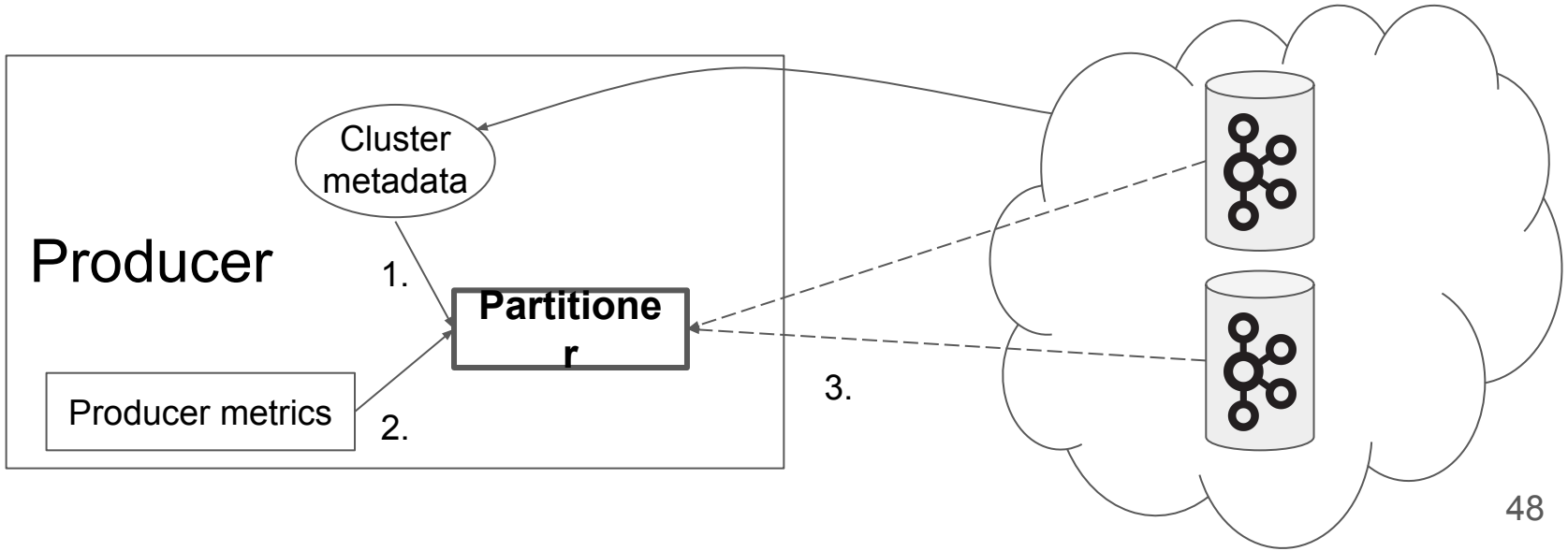
Astraea Partitioner 目標

- 根據 “叢集狀況” 來選擇 partition
- 叢集資源多樣, “負載” 需要定義



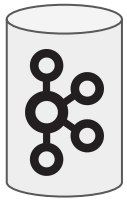
Astraea Partitioner 運作 (1/3): 數據取得

1. Cluster metadata
2. Producer metrics
3. remote JMX server



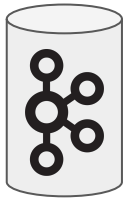
Astraea Partitioner 運作 (2/3): 負載定義

- 各種數據統一成 “節點負載”
  - 為何是以節點為單位？
  - 為何不直接拿數據來選擇 partition？



Latency: 2 ms
Disk usage: 70%

➡ **0.54**

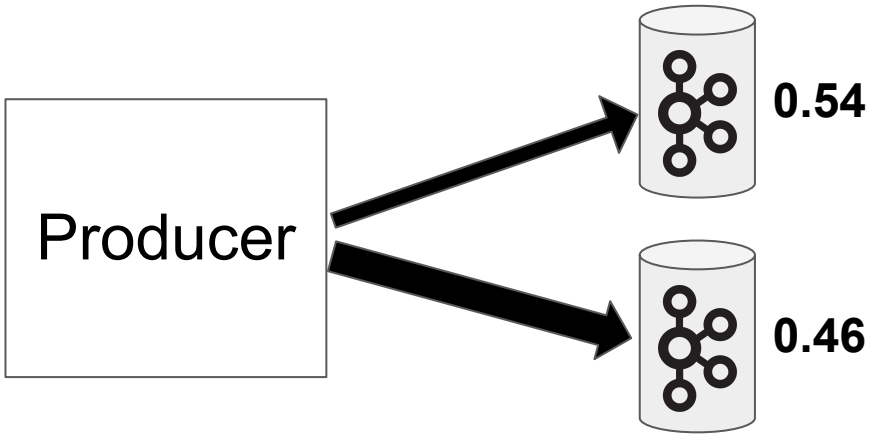


Latency: 8 ms
Disk usage: 10%

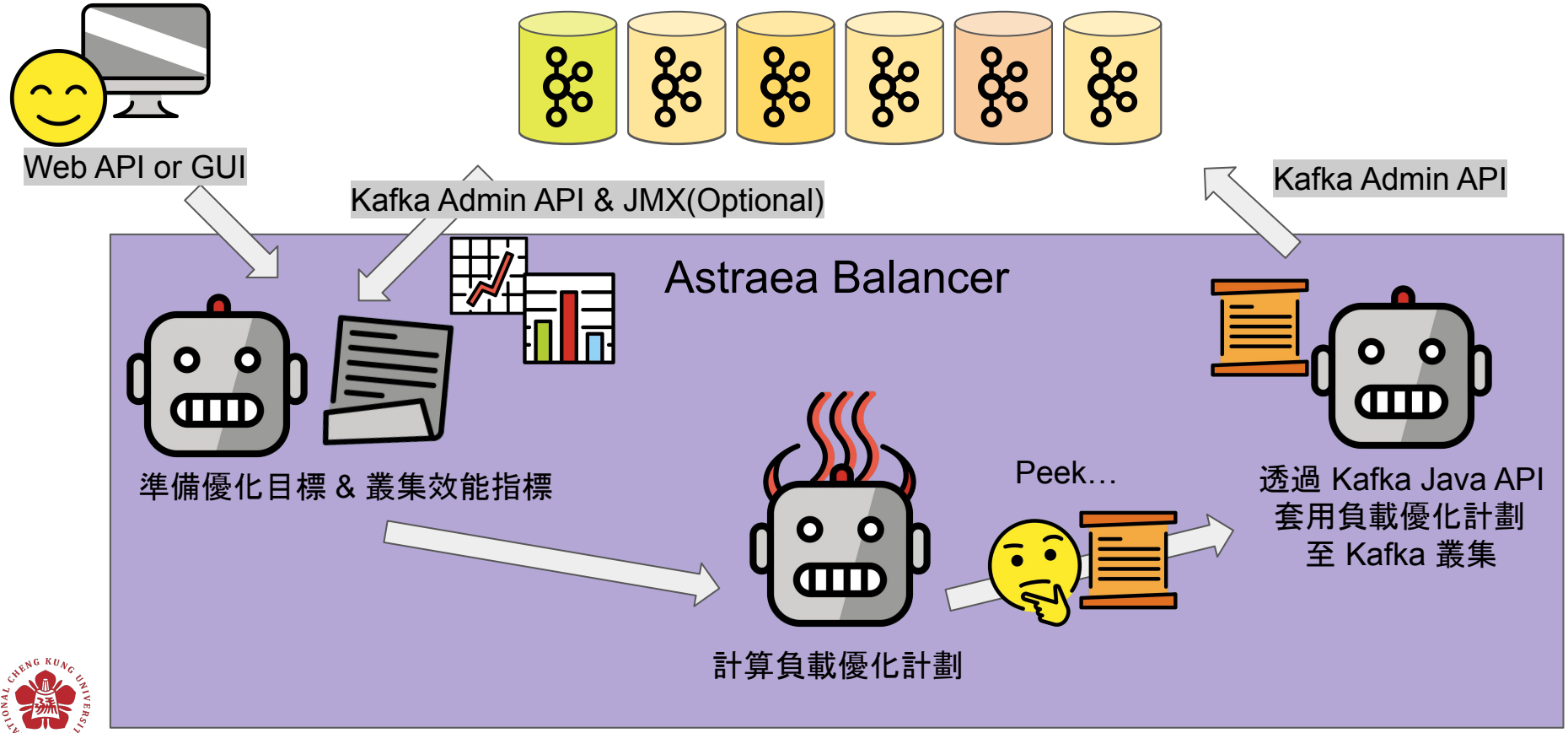
➡ **0.46**

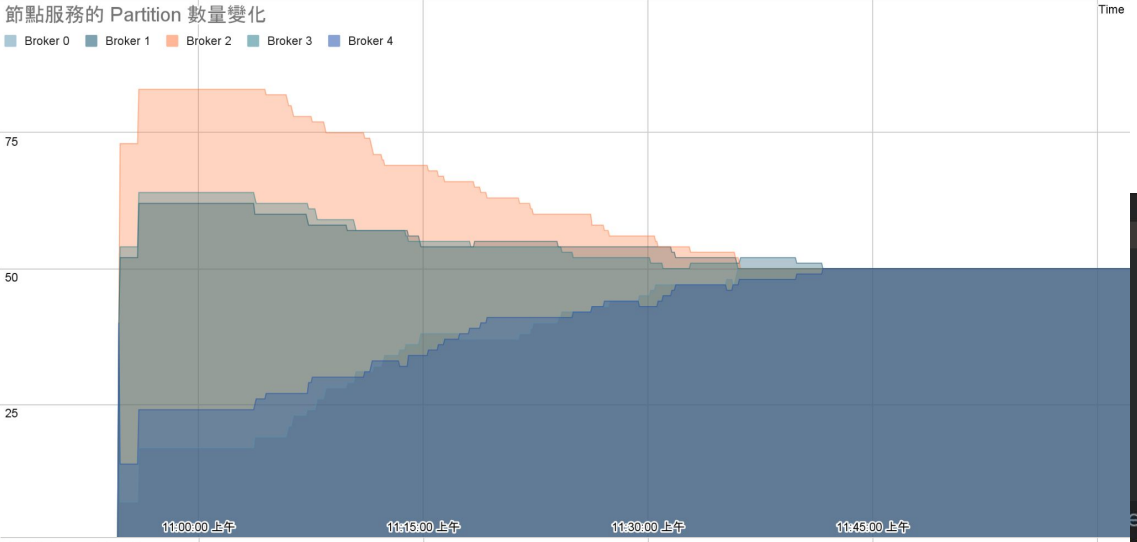
Astraea Partitioner 運作 (3/3)：選擇節點

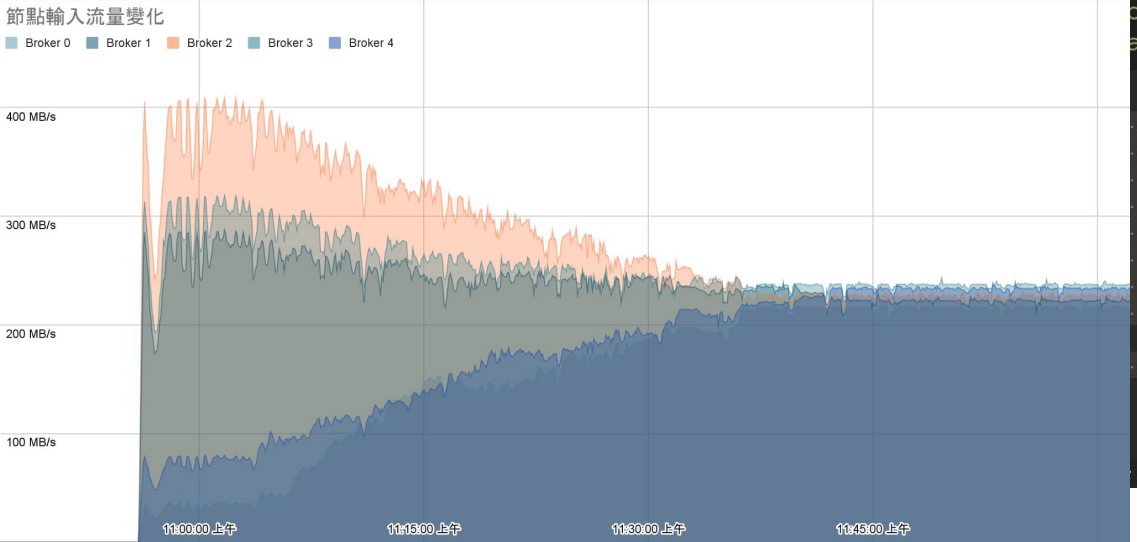
- 按 “負載” 決定節點出現頻率
- 避免連續選到相同節點

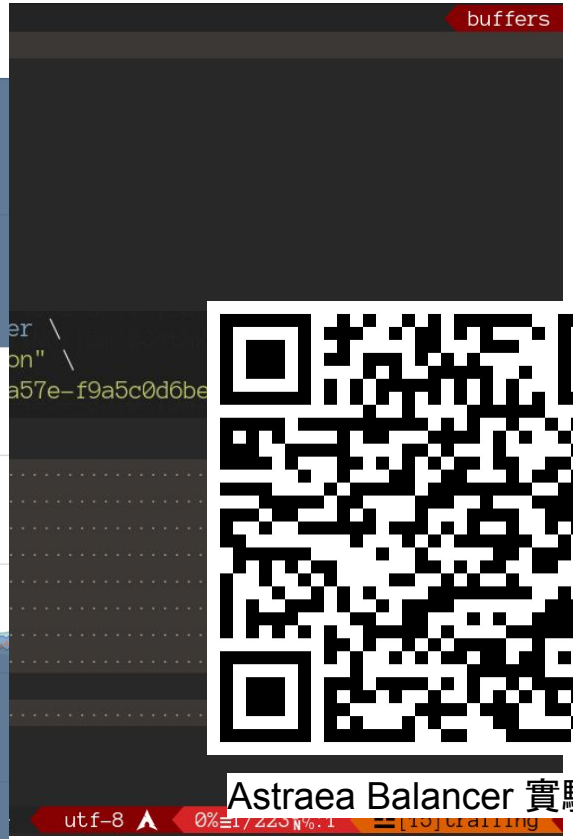


Astraea Balancer 負載優化整體流程









Astraea Balancer 實驗#1

工人智慧: 手動轉移負載

欲優化的目標

要轉移 Leader 或是 Follower

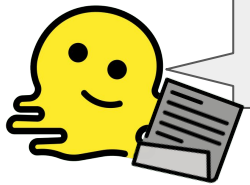
要移動到哪個節點

哪個 Partition 要移動...

(JBOD Enabled) 要放到哪個資料夾

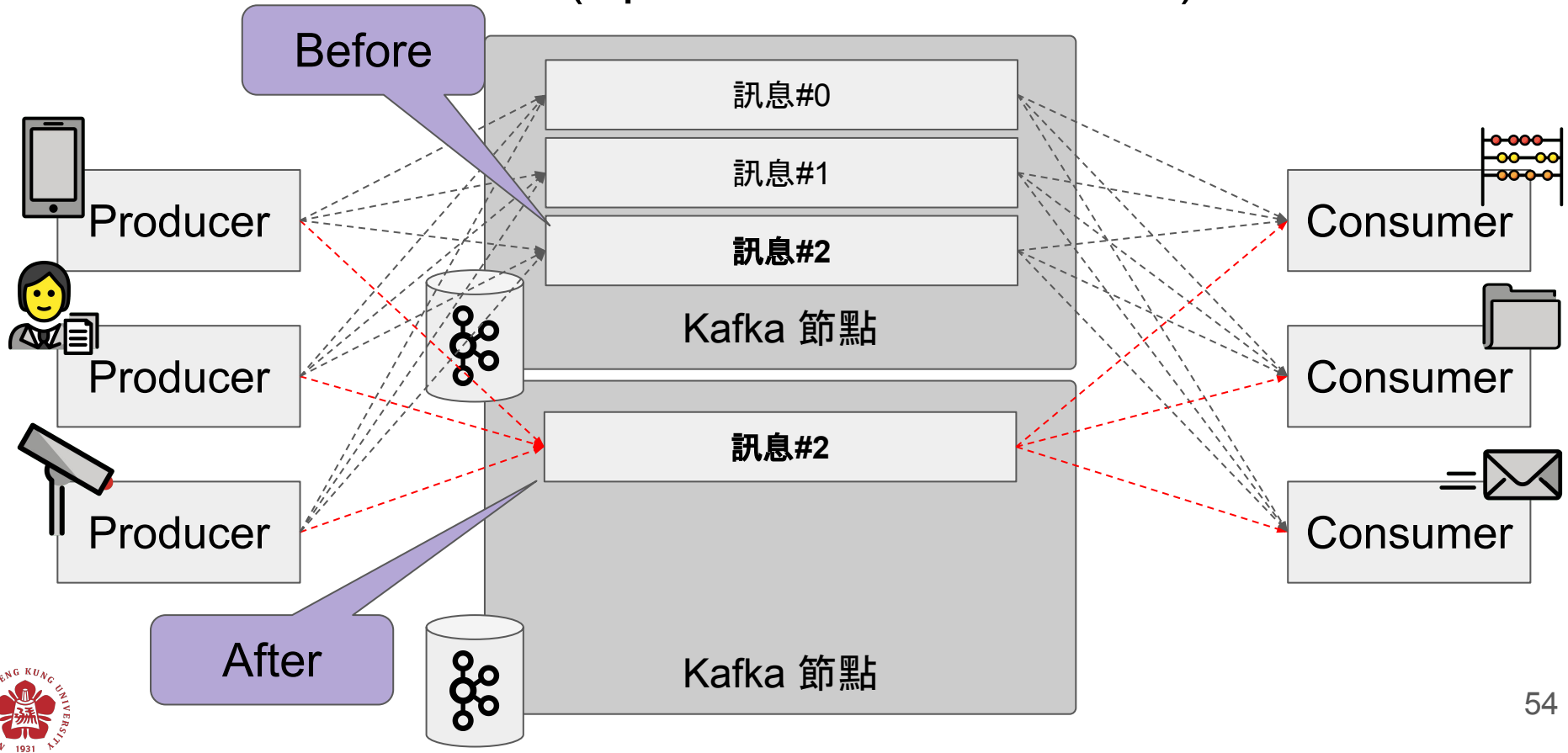
Topic/Partition	備份 Leader	備份 Follower
訊息#0	(broker0, /ssd1)	(broker1, /ssd1), (broker3, /ssd1)
訊息#1	(broker1, /ssd2)	(broker2, /ssd1), (broker0, /ssd1)
數據#0	(broker2, /ssd1)	(broker0, /ssd1), (broker1, /ssd2)
錯誤#0	(broker0, /ssd2)	(broker1, /ssd2), (broker2, /ssd1)

這個決策要做多少次

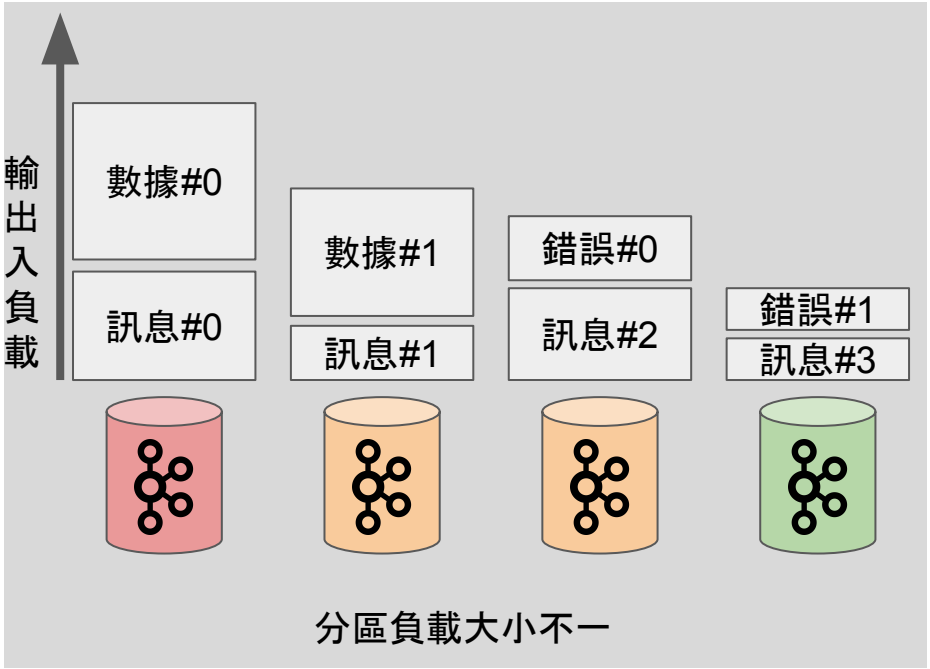


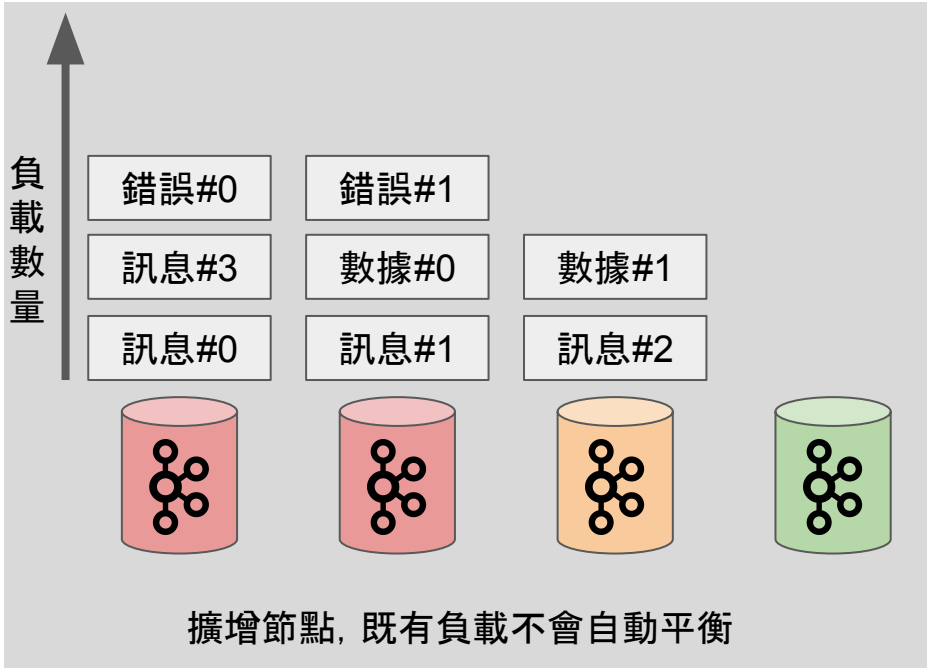
這些變更會不會影響其他我想優化的指標

Kafka 的負載是什麼 (Apache Kafka 系統模型)

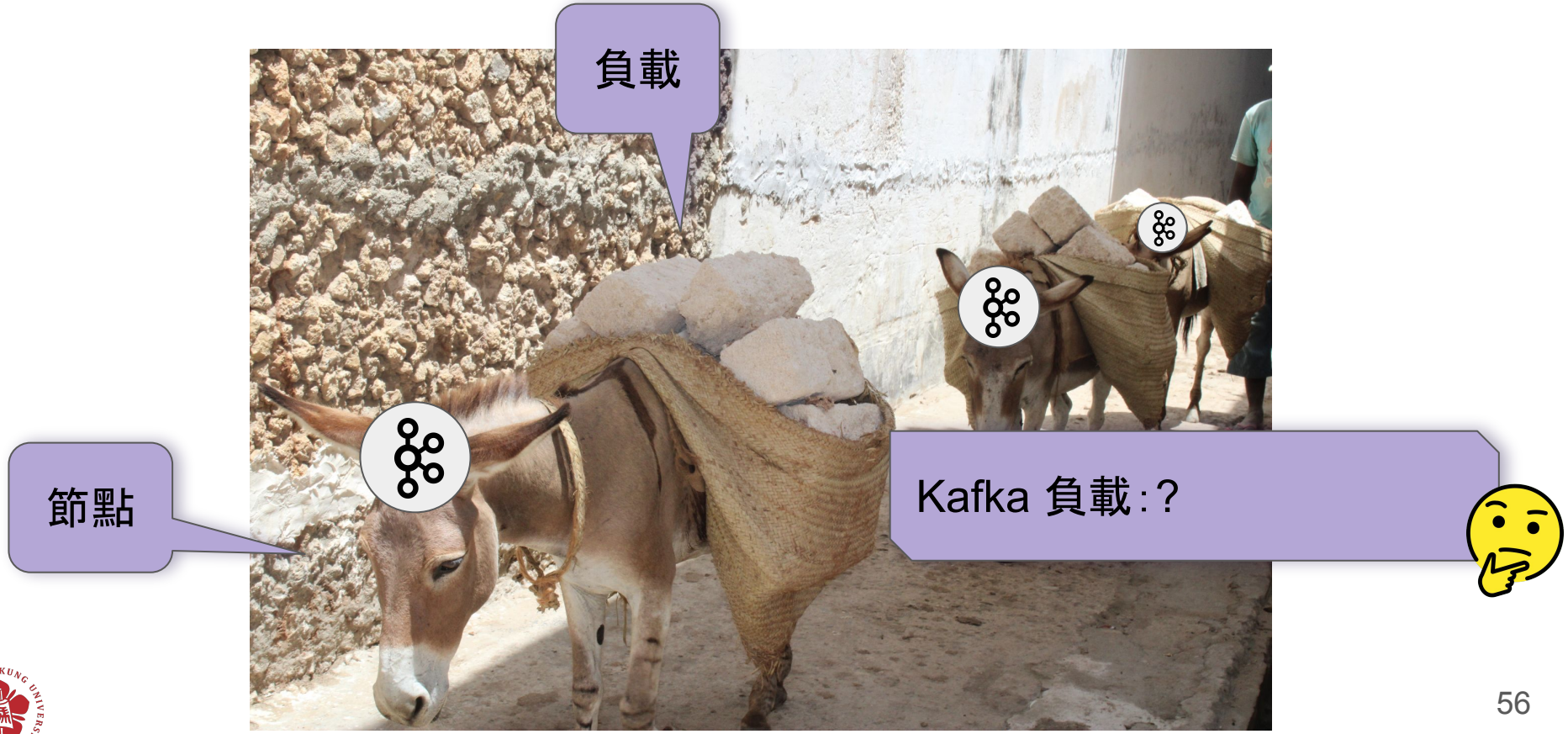


天然的 Kafka 負載不平衡現象





負載? 平衡?



描述問題(1/2) - 如何描述負載

- 如何描述負載的狀態

從節點描述

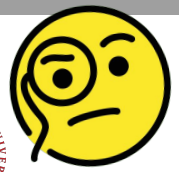
誰是備份

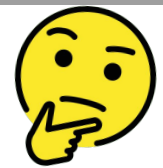
節點	負載
駱駝0	(訊息#0), (訊息#1)
駱駝1	(訊息#2), (訊息#1)
駱駝2	無

知道了負載，那平衡?

- 為什麼要平衡
  - 維護者角度：資源使用最大化，減少成本。
  - 使用者角度：服務運作穩定。
- 直覺上的平衡
  - 平衡 Kafka 節點的輸入/輸出負載
- 廣義上的平衡 = 優化
  - 異質系統：耐操的節點負責更多負載
  - QoS：特定 Topic 的資料放在 SSD 的儲存空間，剩餘的放 HDD

Apache Kafka 在節點端的負載平衡做了什麼措施?

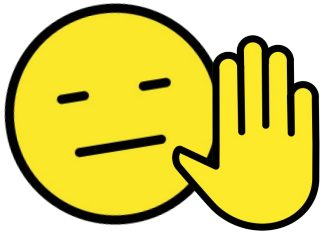


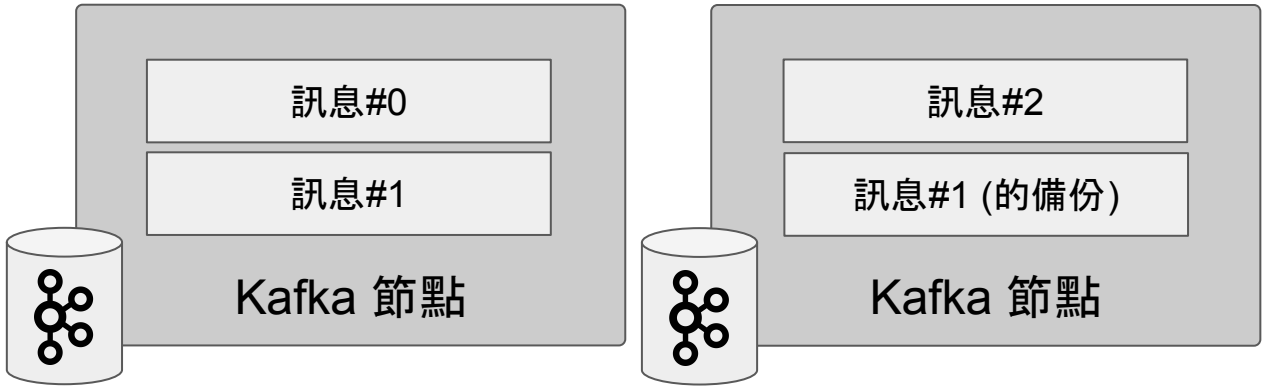


工人智慧: 手動轉移負載

- 示範手動搬移的過程

在此之前，需要更好且真實的負載表示方式



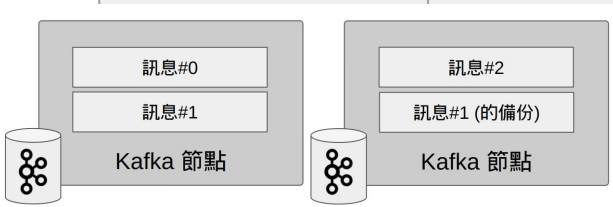


工人智慧: 手動轉移負載

- 示範手動搬移的過程

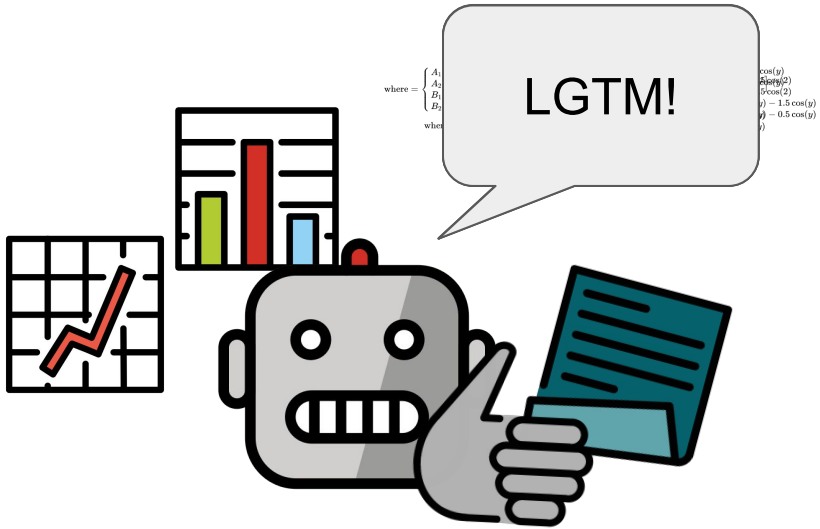
在此之前，需要更好且真實的負載表示方式

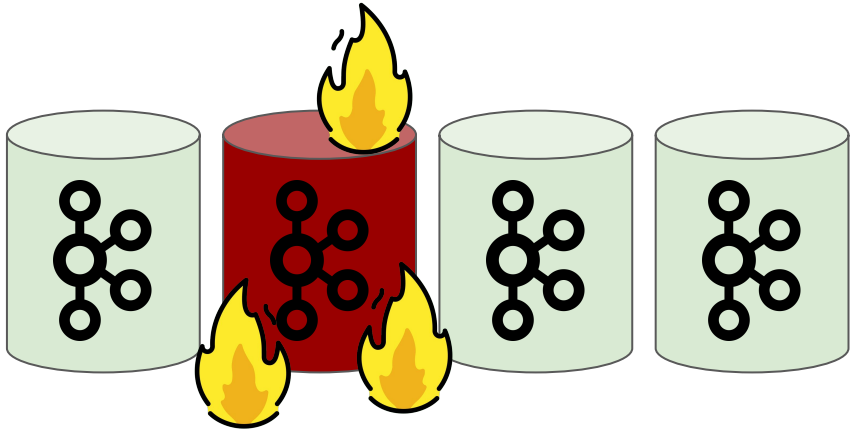
Topic/Partition	備份 Leader	備份 Follower
訊息#0	(broker0, /ssd1)	(broker1, /ssd1), (broker2, /ssd2)
訊息#1	(broker1, /ssd2)	(broker2, /ssd1), (broker0, /ssd1)
數據#0	(broker2, /ssd1)	(broker0, /ssd1), (broker1, /ssd2)
錯誤#0	(broker0, /ssd2)	(broker1, /ssd2), (broker2, /ssd1)



Astraea Balancer 的挑戰

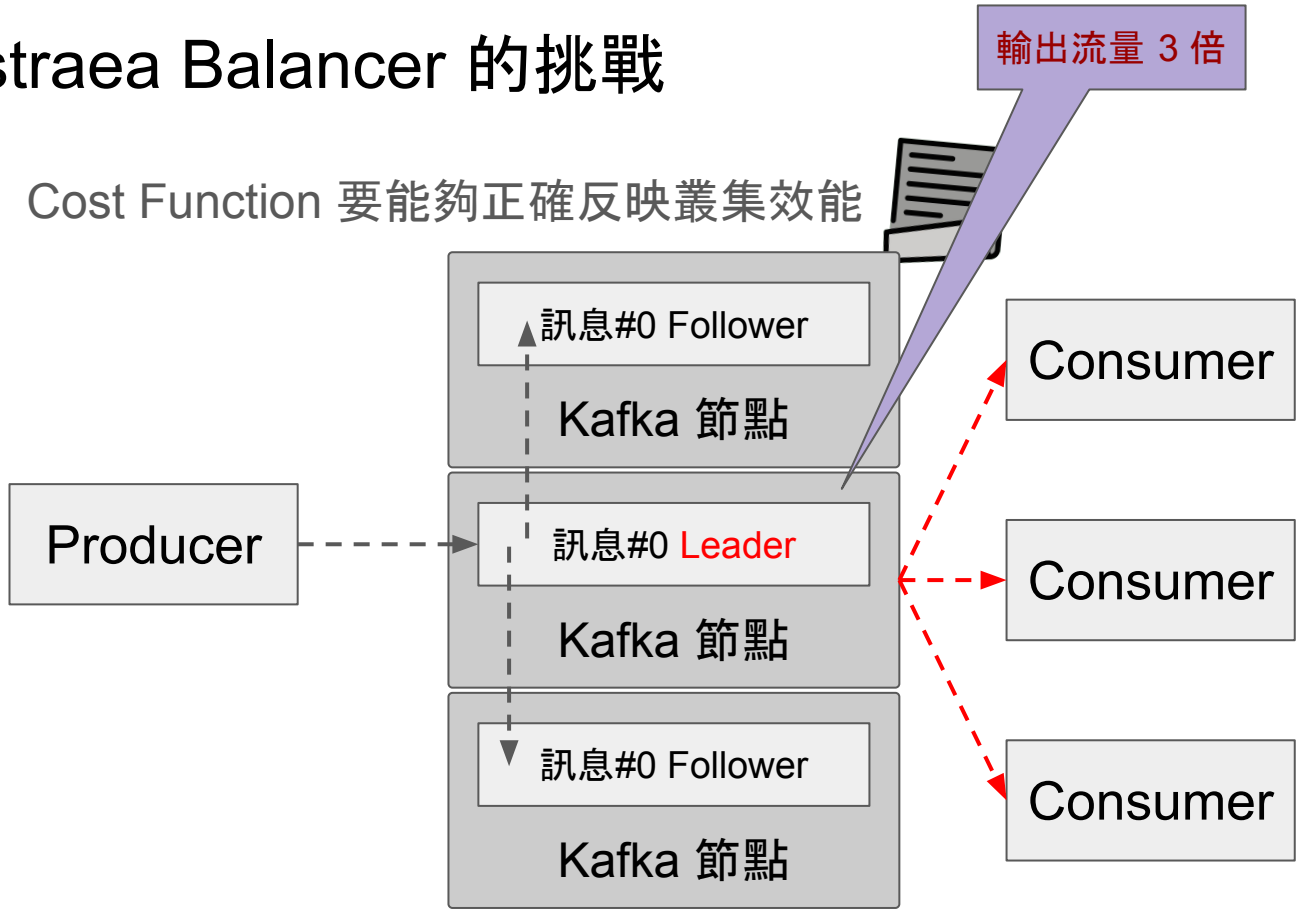
- Cost Function 要能夠正確反映叢集效能





Astraea Balancer 的挑戰

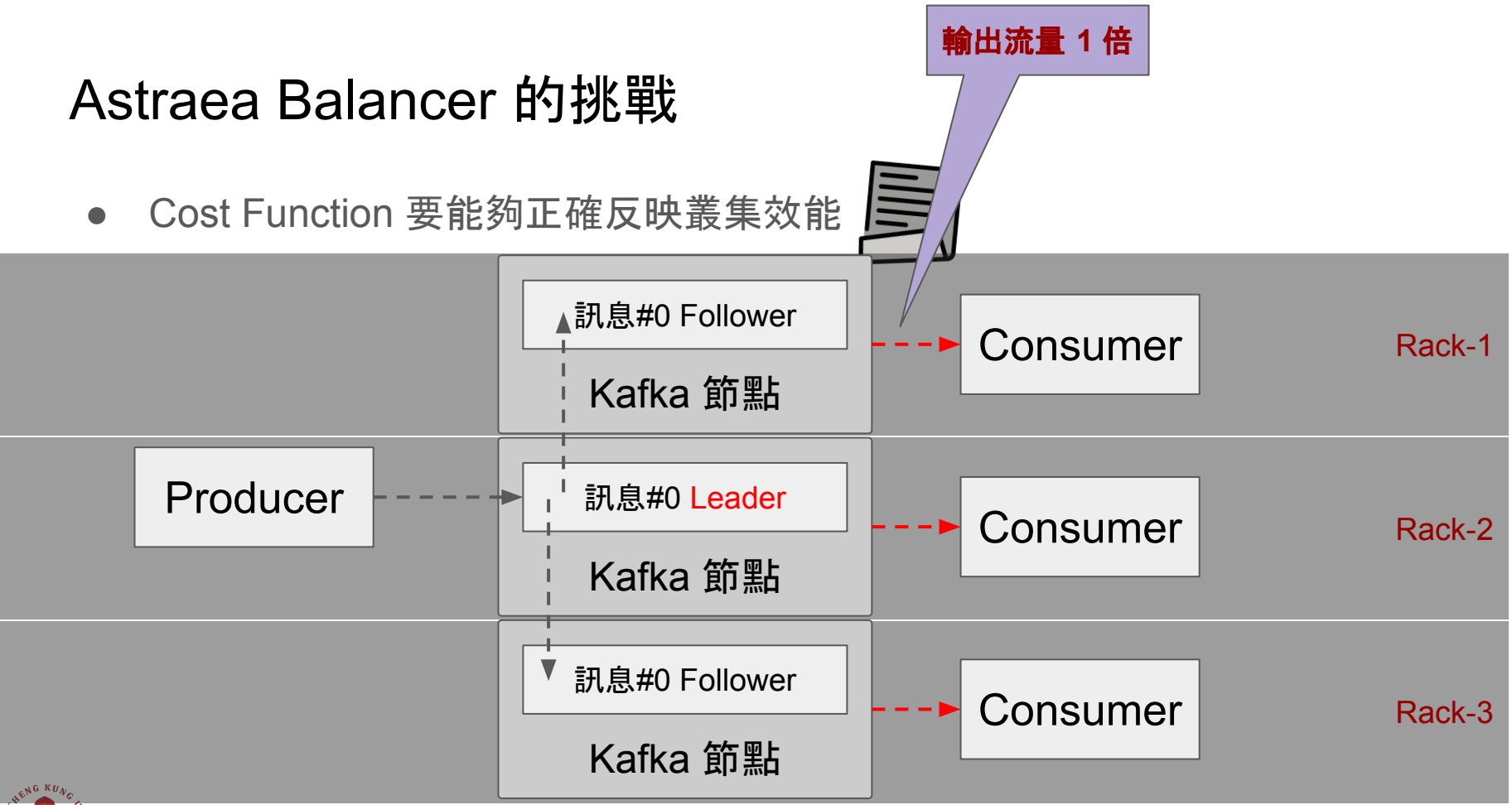
- Cost Function 要能夠正確反映叢集效能



無機架感知 (No Rack Awareness)

Astraea Balancer 的挑戰

- Cost Function 要能夠正確反映叢集效能



有機架感知 (With Rack Awareness)

Astraea Balancer 的挑戰

- 想優化的項目和願望一樣多
- 願望越多實現難度越高
- More Cost Function = Complex Optimization Problem
- 搜尋演算法需要能夠在複雜的情境下去搜尋可用的負載分佈