Kafka 的最後一哩路

講者 : 王懿宸

關於我

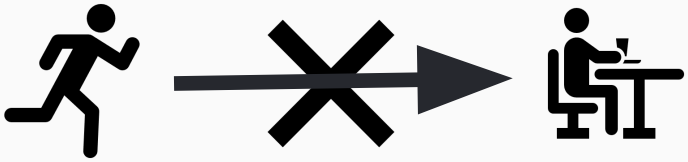
- 成功大學資工研究生
- 研究 Kafka 負載平衡議題
- https://hackmd.io/@warren215/wyc

- 演講者
  - 王懿宸
- 核心開發者
  - 方竫泓, 孫祥鈞, 李宜桓, 李政憲, 蔡嘉平, 蕭宏章, 鄧智懋, 魏連興
- 特別感謝
  - 成功大學
  - 原昌工業
  - 亦思科技
  - 來自其他工程師的貢獻

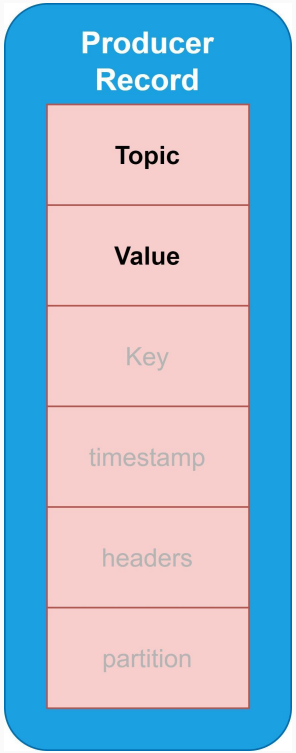
Kafka 介紹

Background : What's Kafka?

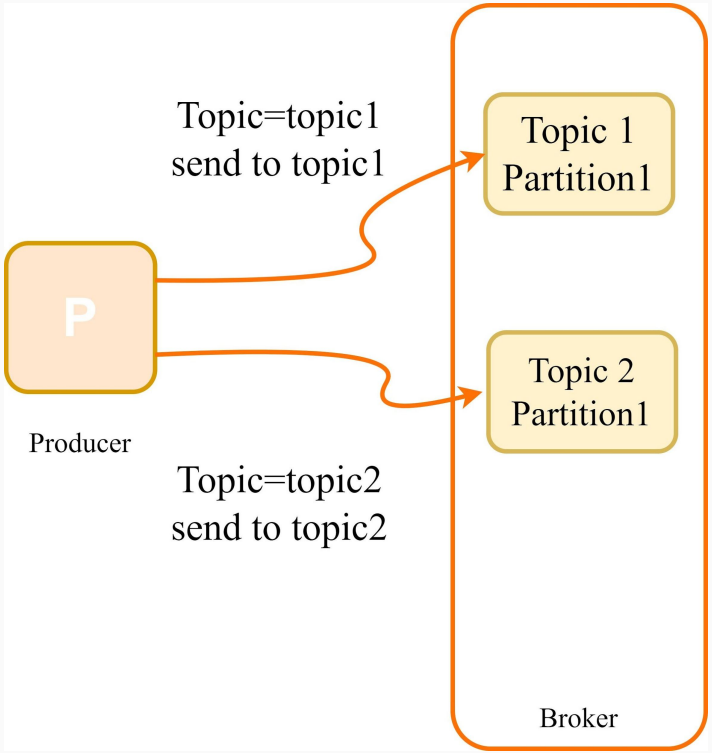
要想瞭解Kafka，首先需要知道什麼是消息系統。



Background : Kafka send record



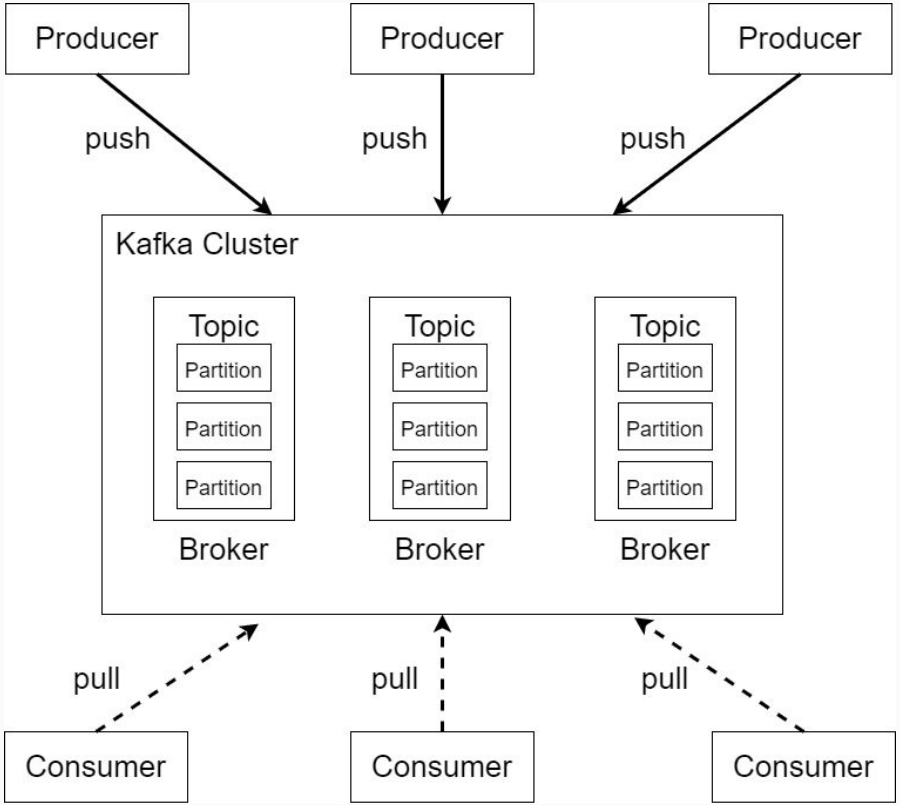
構建 Producer record 需要的屬性



Producer 將 record發送到其對應的topic中

Background : Kafka module

- Producer向目標Topic推送資料
- Consumer從目標Topic拉取資料
- 資料被存放在Partition中，一個Topic是由分佈在不同節點的Partition共同構成的。



Motivation and problems

Event Streaming

Transaction

Compression

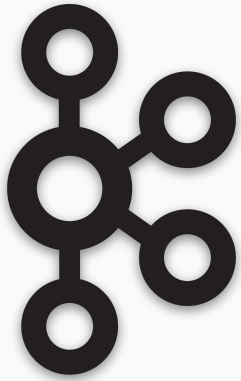
JBOD

Fault Tolerance

Encryption

Resource Quota

Replication



Authentication

At-most once delivery

Authorization

At-least once delivery

Exactly once delivery

Multi-Tenancy

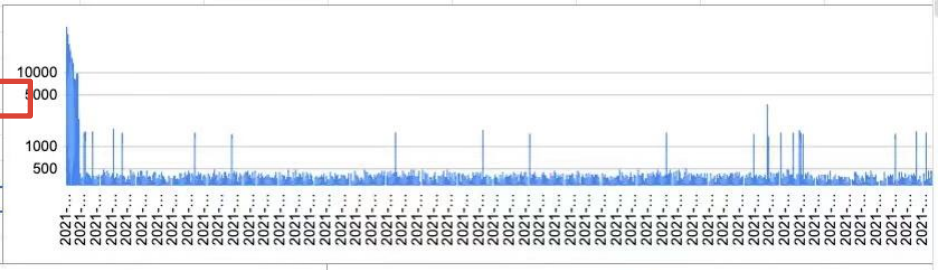
High Throughput

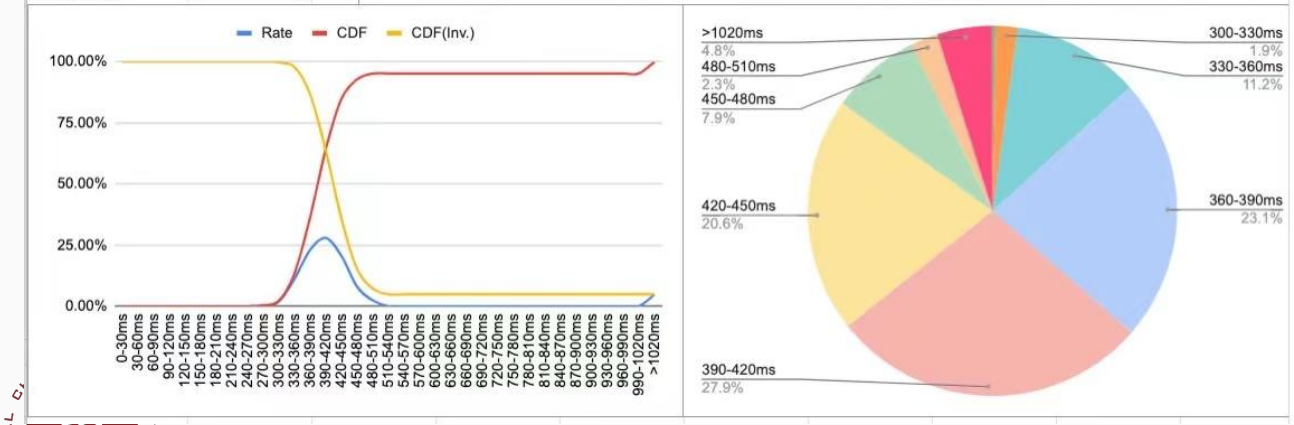
Astraea初衷的由來

Motivation and problems

你嘗試發送一些測試資料，結果發現每筆資料的延遲高達 1400ms。

Messages	681	
Message Size (Avg.)	361	Byte
Total Data Size	240.0546875	KB
average speed	1411.558003	ms
lowest speed	42891	ms
fastest speed	293	ms
MessageMaxBytes	1000000	
FetchMaxBytes	52428800	
Consumers	3	



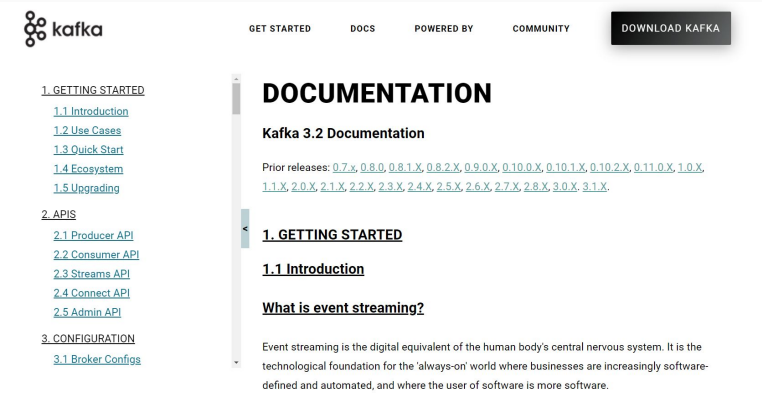


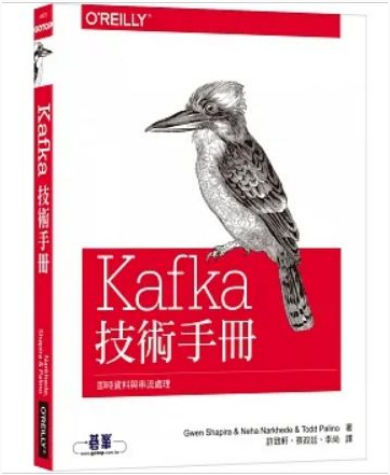
kafka

HIGH THROUGHPUT

Deliver messages at network limited throughput using a cluster of machines with latencies as low as 2ms.

Motivation and problems





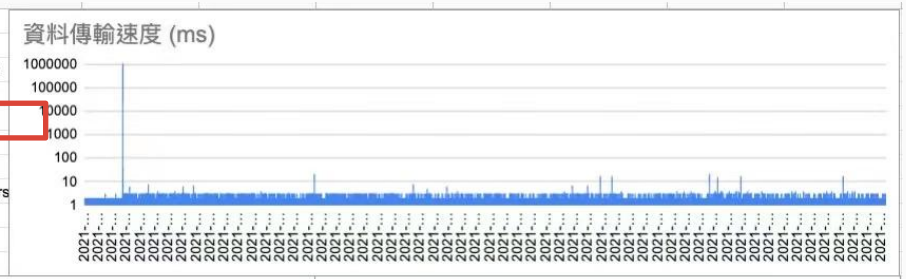
CONFLUENT

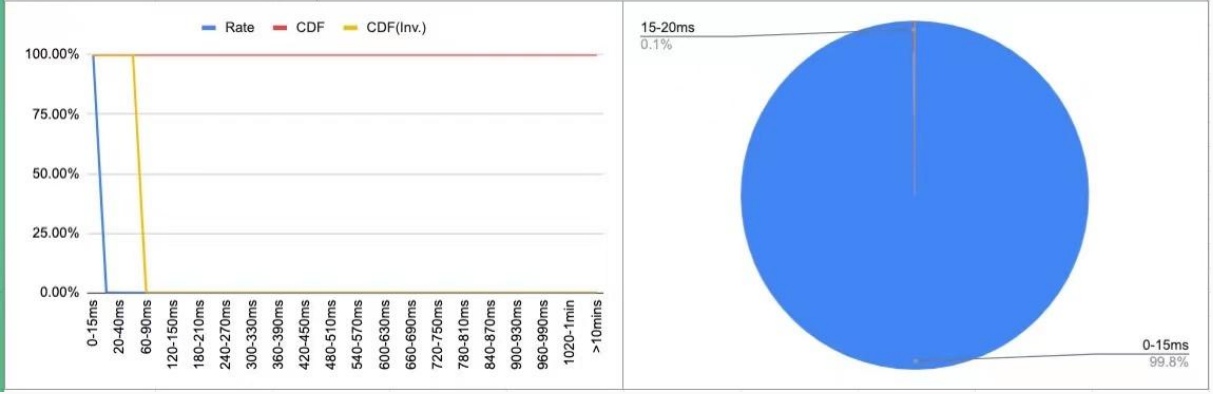
	Boyang Chen Committer /in/boyangc		Xi Hu Committer /in/xi-hu-892b5b84
	David Jacot Committer, and PMC member /in/davidjacot @davidjacot		Chia-Ping Tsai Committer, and PMC member /in/chia7712

Google

Motivation and problems

Messages	4760	
Message Size (Avg.)	106	Byte
Total Data Size	494.921875	KB
average speed	2.385084034	ms
lowest speed	1198962	ms
fastest speed	0	ms
Executing TIme	3.482391667	Hours
ack	0	
FetchMaxBytes	default	
Consumers	1	





Kafka叢集是一個動態的環境

- 隨著用戶對處理資料能力需求的增加，加入的新節點
- 叢集同時需要處理的服務變化
- 基於不同目的與需求，導致的不同Topic創建策略
- 節點下綫導致原先為replica的partition變為leader

效能議題成为了
使用Kafka的最後一哩路

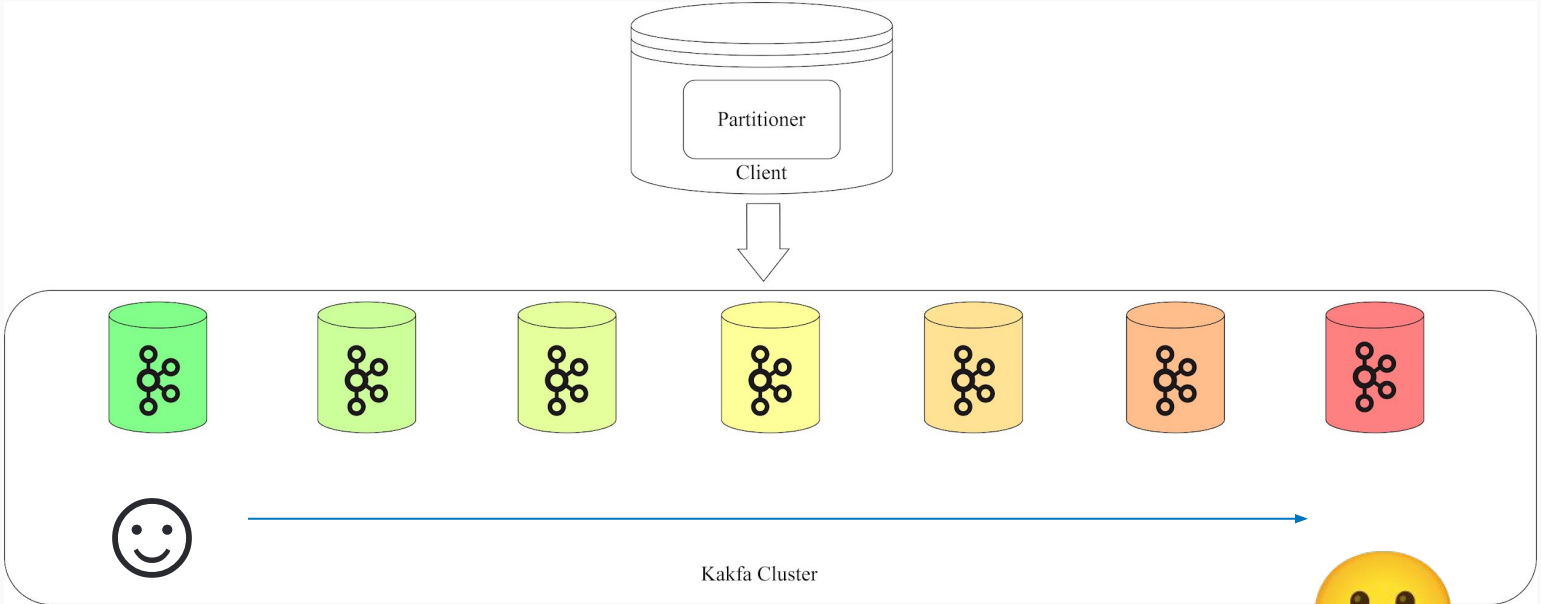
Astraea Dispatcher
幫助用戶輕鬆走完最後一哩路

Kafka Partitioner

- Partitioner決定了每筆資料會發往哪一個Partition
- Kafka 允許用戶根據自身需求自定義Partitioner
- Astraea Dispatcher：將每一筆資料按照**節點負載狀況**進行分配達到叢集負載平衡
  1. 評估節點狀態
  2. 以節點分數為標準派送資料

Node Load

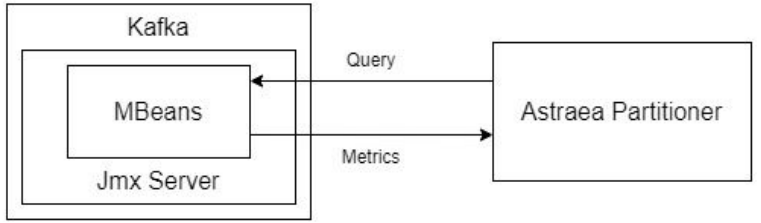
- 叢集中機器的負載狀況各異，
- 作為用戶一定希望狀況最好的機器獲得資料。



Kafka metrics

Kafka 本身提供了眾多metrics,

能夠反應叢集的各種狀況。



ResponseQueueTimeMs

RequestQueueTimeMs

UnderReplicatedPartitions

BytesOutPerSec

LocalTimeMs

TotalTimeMs

BytesInPerSec

ResponseSendTimeMs

DiskUsage

RemoteTimeMs

Kafka Cluster State

request-latency-avg

CPUusage

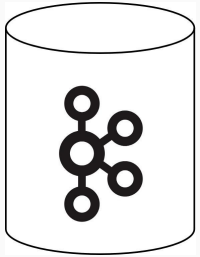
batch-size-avg

RequestsPerSec

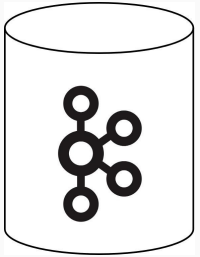
Node score from single metric

通過對節點間狀況的比較，得出各節點的分數。

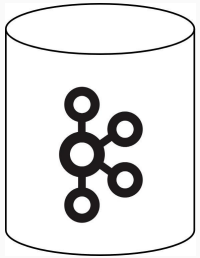
BytesInPerSec



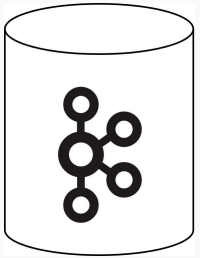
600MB/s
Score:48分



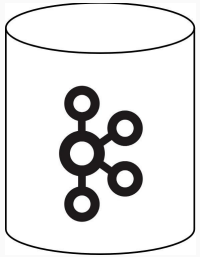
400MB/s
Score:40分



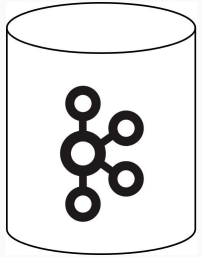
500MB/s
Score:60分



200MB/s
Score:80分

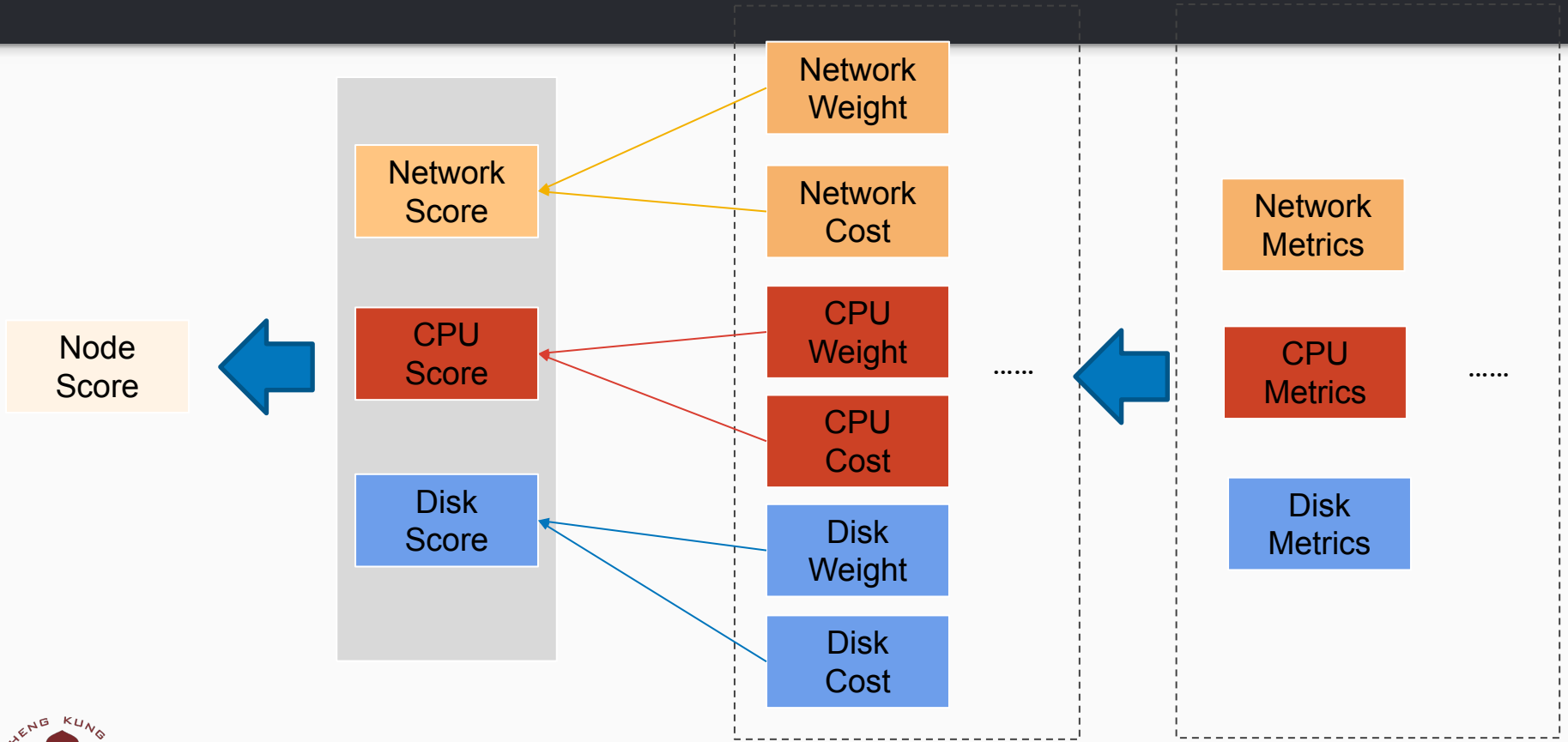


400MB/s
Score:72分



200MB/s
Score:80分

Node score from metrics



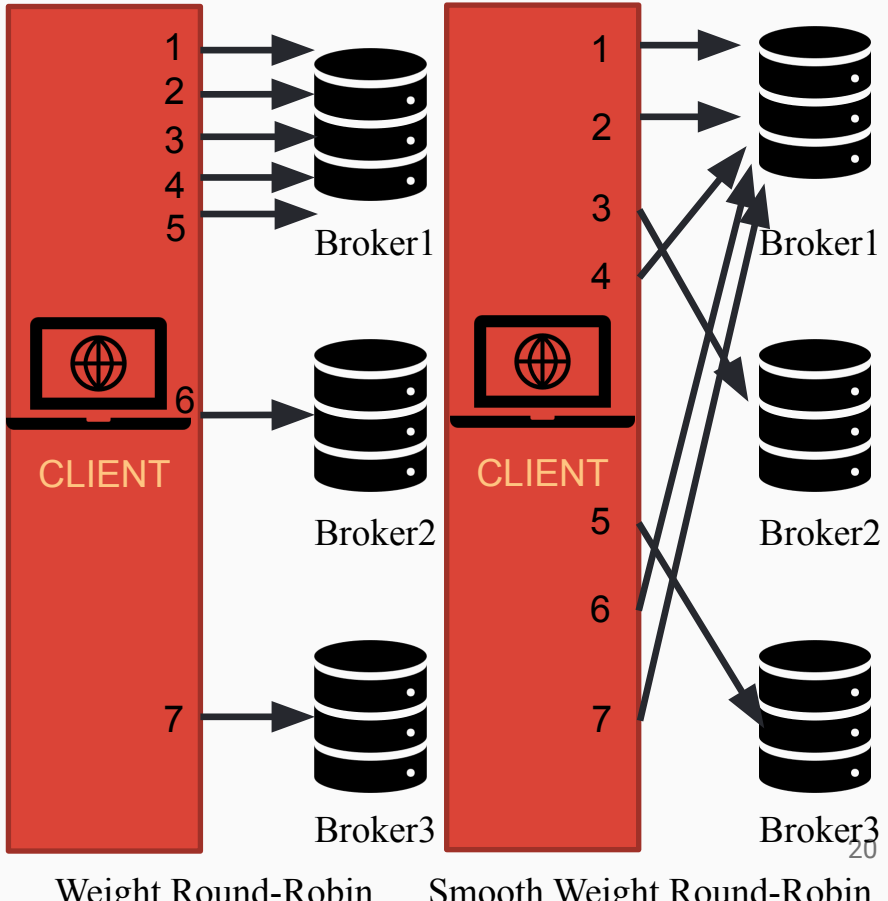
Node Score = Network Score + CPU Score + Disk Score

Smooth Weighted Round-Robin

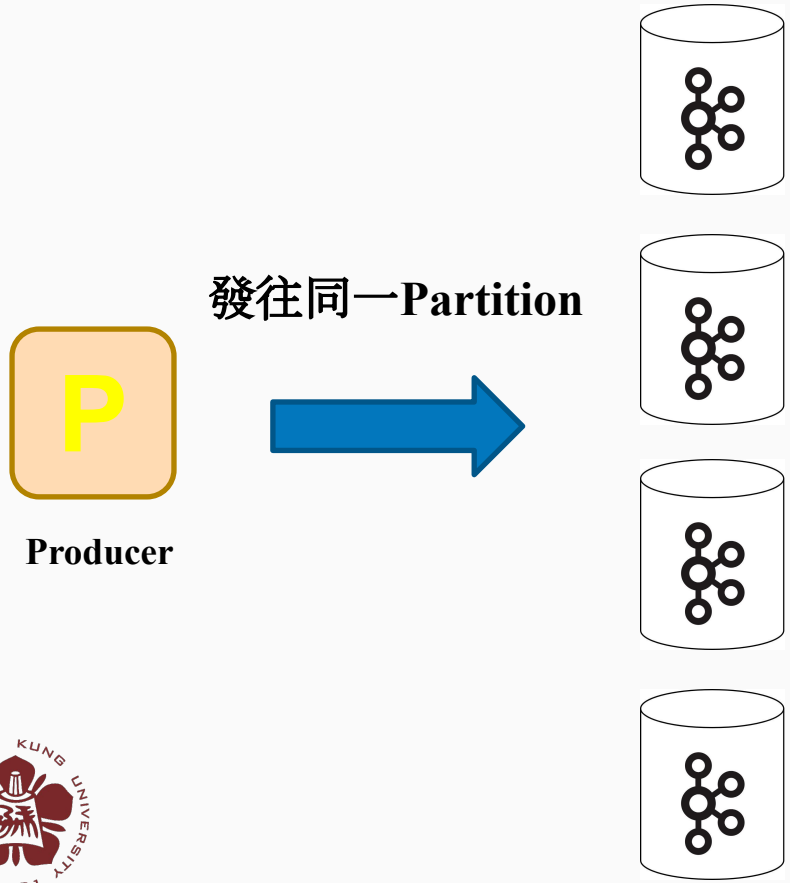
- 一種加權輪詢算法，能夠根據節點的分數狀況進行資料分配。
- 其特點是平滑，避免低權重的節點長時間處於空閒狀態。

假設Client能夠同時發送五筆資料
單臺Broker只能同時處理三筆資料

叢集機器	Effective weight
Broker1	5
Broker2	1
Broker3	1



Astraea Dispatcher : Interdependent Dispatcher



Interdependent Dispatcher

- 被搭建在所有Dispatcher之上，通過簡單的呼叫方法控制功能的開關。
- 開啟後將挑選一個狀況最好的partition作為目標，所有使用該Dispatcher的資料都會發送到該partition。
- 搭建在界面中，任意Dispatcher的實作都能使用此功能。會利用該Dispatcher的特色來挑選

Astraea Dispatcher module

評估節點狀態

- 以Kafka metrics作為評估標準
- 根據節點間的比較評估每個節點的分數
- 根據不同的需求以不同的metrics進行判斷比較

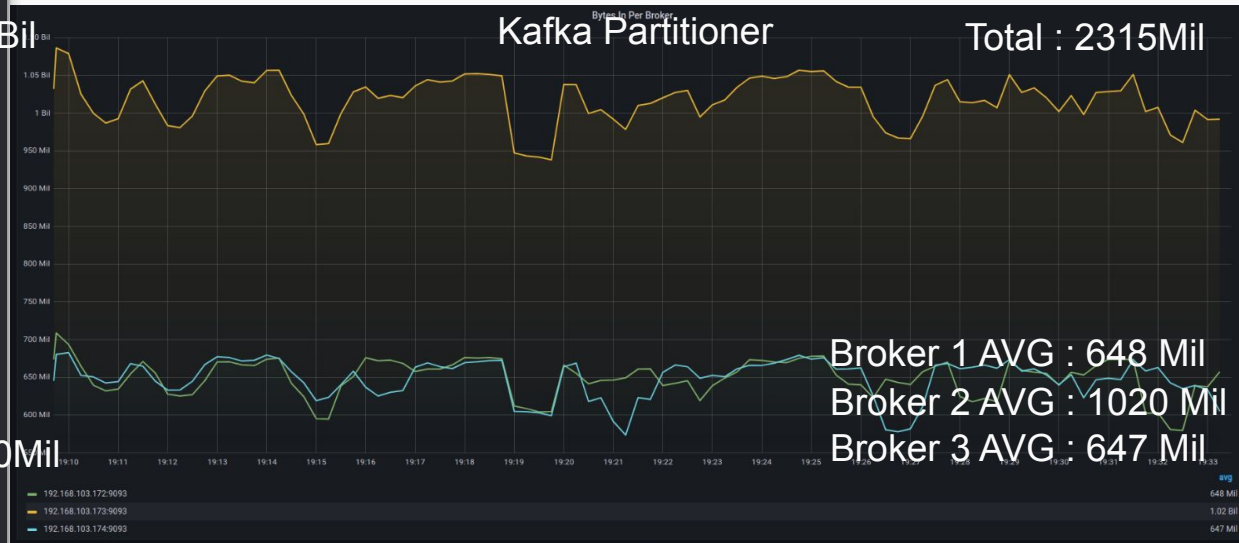
Astraea Dispatcher

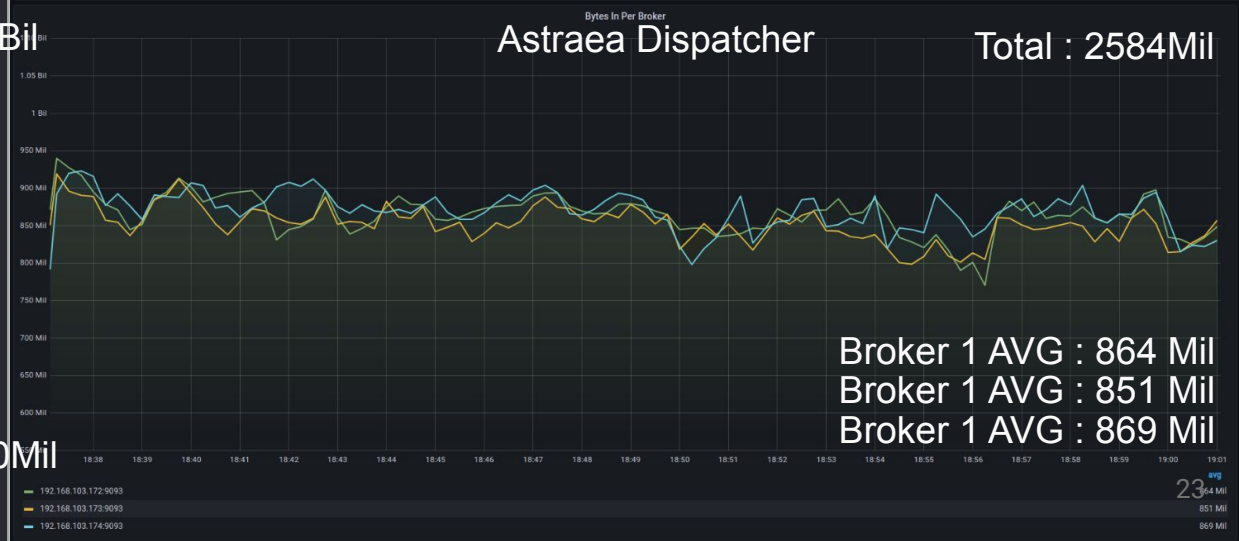
- Dispatcher 會以節點分數為標準派送資料
- 支持不同的資料派送模式

Dispatcher能根據需求使用不同的評分模組，以實現複用，幫助用戶針對環境變化進行調配。

實驗結果

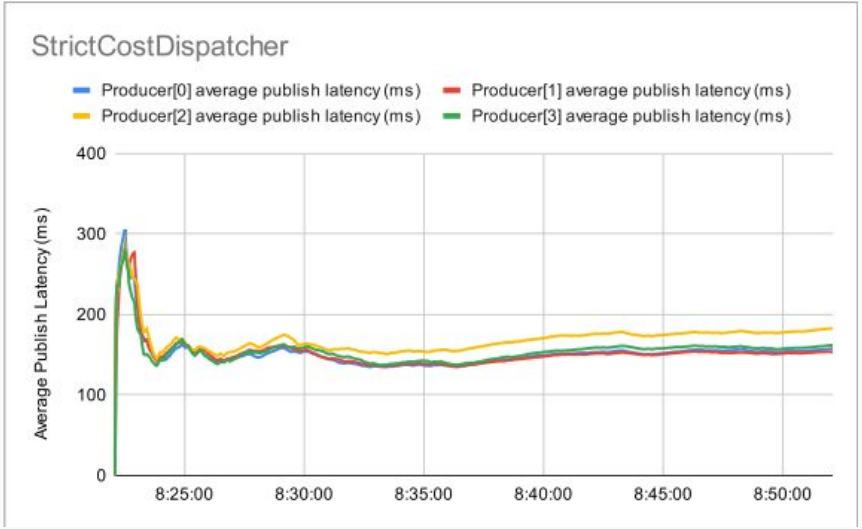
- 3 broker
  - 3 台網路吞吐 10Gbps
- 3 topic 分別 30 個 partition
  - 有一台機器擁有更多的 partition
- 3 台機器同時 produce
  - 分別發送到不同 topic
  - 全力發送
- 整體效能提升約12%
  - 對閒置節點的利用率提升在33%和 34%

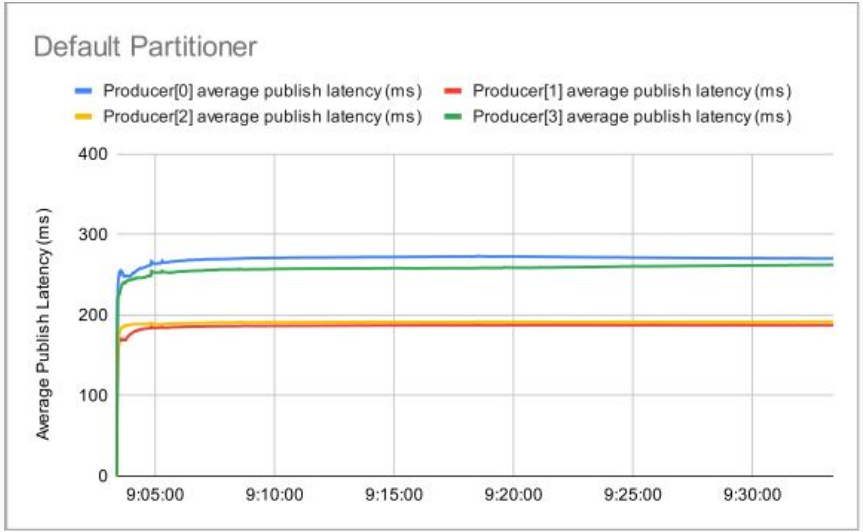




實驗結果

- 4 broker
  - 3 台在switch A
  - **1 台**在switch B
- 1 topic 分佈在4 broker上
- 1 台機器作為 produce
  - 發送資料到topic
  - 機器位於switch A
- 整體latency相比之前降低約30%





Conclusion

- 搭建在partitioner，只要導入專案任何Kafka使用者都能輕鬆使用
- 任何情況都能運行，運行後就能立即生效，改善叢集的整體效能
- 功能模組化，能夠調整對應的模組來實現對應的需求

Astraea Dispatcher	Astraea Balancer	Astraea Assignor
Producer 寫入端	Server 叢集	Consumer 讀取端

感謝聆聽
歡迎提問

Astraea 專案
Github連結