91APP

非同步系統的服務水準保證

Andrew Wu, 2020/11/12

淺談非同步系統的 SLO 設計

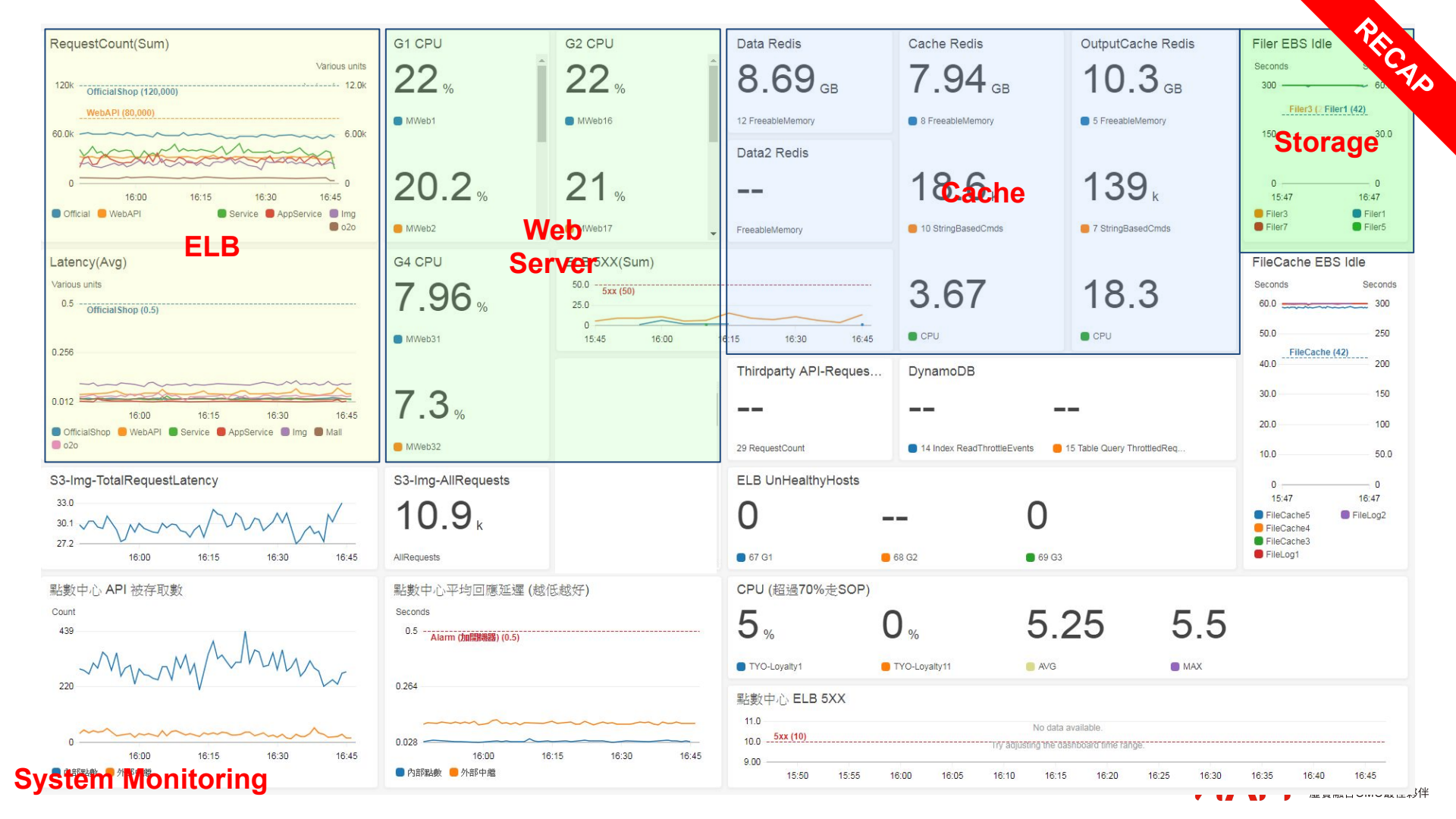
AGENDA

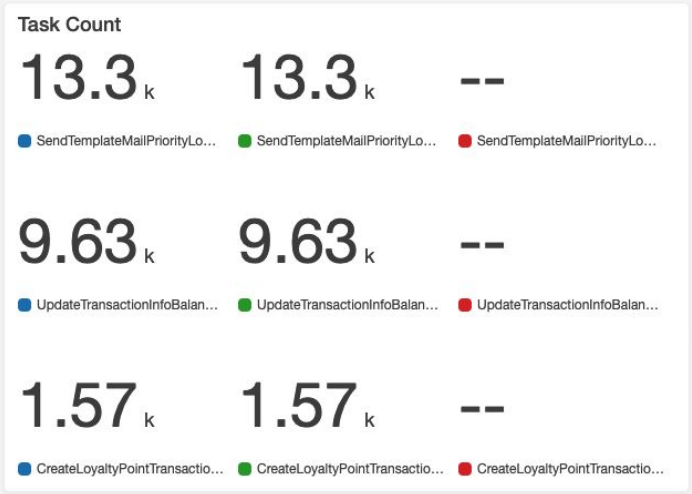
服務水準的概觀: SLA, SLO, SLI ?

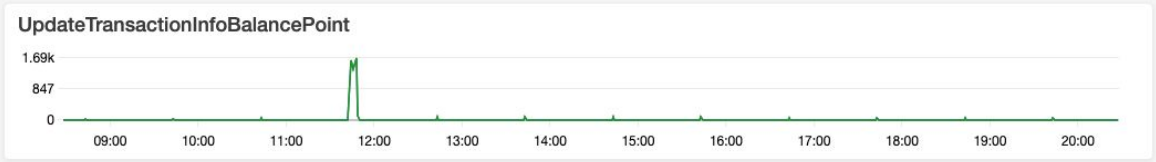
RECAP

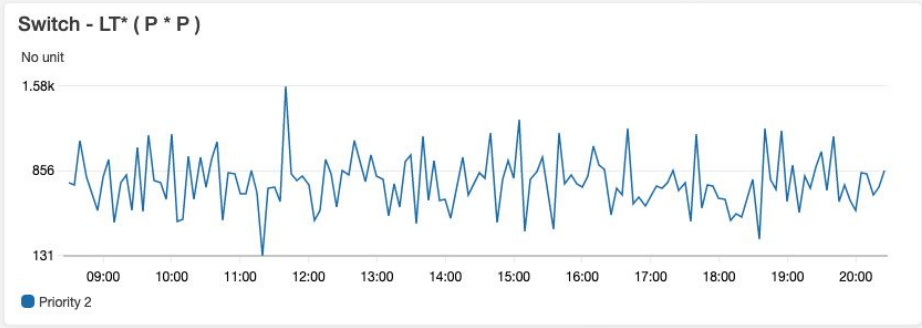
維運管理重點

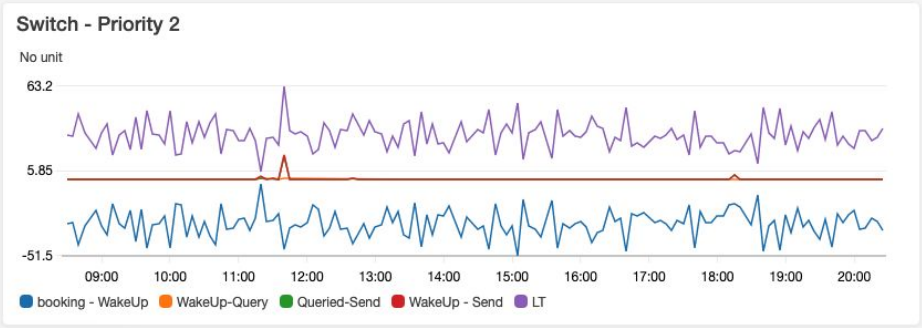
- 系統監控
  - 「能被量測的系統，才能被控制」
- 預防型維運管理
  - 設定目標
  - 量測指標
  - 提前改善


RECAP
RequestCount(Sum)
Various units
120k
12.0k
OfficialShop (120,000)
WebAPI (80,000)
60.0k
6.00k
0
0
16:00
16:15
16:30
16:45
Official
WebAPI
Service
AppService
Img
o2o
ELB
Latency(Avg)
Various units
0.5
OfficialShop (0.5)
0.256
0.012
16:00
16:15
16:30
16:45
OfficialShop
WebAPI
Service
AppService
Img
Mall
o2o
G1 CPU
22 %
MWeb1
20.2 %
MWeb2
G2 CPU
22 %
MWeb16
21 %
MWeb17
Web Server
G4 CPU
7.96 %
MWeb31
7.3 %
MWeb32
ELB 5XX(Sum)
50.0
25.0
0
5xx (50)
15:45
16:00
16:15
16:30
16:45
Data Redis
8.69 GB
12 FreeableMemory
Data2 Redis
--
FreeableMemory
Cache Redis
7.94 GB
8 FreeableMemory
18.6
10 StringBasedCmds
Cache
OutputCache Redis
10.3 GB
5 FreeableMemory
139 k
7 StringBasedCmds
3.67
CPU
18.3
CPU
Filer EBS Idle
Seconds
300
60
Filer3 (Filer1 (42)
150
30.0
0
0
15:47
16:47
Storage
Filer3
Filer1
Filer7
Filer5
FileCache EBS Idle
Seconds
Seconds
60.0
300
50.0
250
FileCache (42)
40.0
200
30.0
150
20.0
100
10.0
50.0
0
0
15:47
16:47
FileCache5
FileLog2
FileCache4
FileCache3
FileLog1
Thirdparty API-Reques...
--
29 RequestCount
DynamoDB
--
--
14 Index ReadThrottleEvents
15 Table Query ThrottledReq...
S3-Img-TotalRequestLatency
33.0
30.1
27.2
16:00
16:15
16:30
16:45
S3-Img-AllRequests
10.9 k
AllRequests
ELB UnHealthyHosts
0
--
0
67 G1
68 G2
69 G3
點數中心 API 被存取數
Count
439
220
0
16:00
16:15
16:30
16:45
點數中心平均回應延遲 (越低越好)
Seconds
0.5
Alarm (加開機器) (0.5)
0.264
0.028
16:00
16:15
16:30
16:45
內部點數
外部中繼
CPU (超過70%走SOP)
5 %
TYO-Loyalty1
0 %
TYO-Loyalty11
5.25
AVG
5.5
MAX
點數中心 ELB 5XX
11.0
10.0
9.00
5xx (10)
No data available.
Try adjusting the dashboard time range.
15:50
15:55
16:00
16:05
16:10
16:15
16:20
16:25
16:30
16:35
16:40
16:45
System Monitoring









RECAP

Application Monitoring / *Business Monitoring

系統監控

RECAP

-devops-sys-monitor-

☆ | 130 | 9 | 系統監控：線上服務狀況、異常回報。與系統維護有關的訊息，請至

Yesterday

推播情報員 APP 07:00

今日推播在下列時段超過五十萬筆:

時段	推播數
2019-09-02T10:00:00	1439698
2019-09-02T12:00:00	1447084
2019-09-02T15:00:00	650029
2019-09-02T21:00:00	946459

系統監控 APP 08:00

[RD5 值班] Levi Chen
September 2nd, 2019

[重要資訊] 全聯第一階段上線
September 2nd, 2019

Site24x7 APP 22:13

Site24x7: Family Mart Login Page 全家登入頁 is Critical

Display Name
Family Mart Login Page 全家登入頁

Site monitored
https://ap.family.com.tw/V2/Member/91SignIn?response_type=token&state=http%3a%2f%2fmart.family.com.tw%2fv2%2fofficial&client_id=6eb1c5aa-d61c-4188-a427-03e5916d0ca9&redirect_uri=https%3a%2f%2fservice.91app.com%2fV2%2fLogin%2fThirdpartyBasedOAuthSuccess

Monitor status
CRITICAL

Critical since
August 25, 2019 10:13 PM CST

Monitor Dashboard Link
https://www.site24x7.com/app/client#/home/monitors/297802000000082048/Summary

Downtime in UNIX Format
1566742384598

Resolved IP
210.64.137.213

Up From Locations :
Tokyo - JP

Critical From Locations :
Tokyo - JP

Reason
Response time from Tokyo - JP exceeded 15000 ms.

Event Alert / Notification

RECAP

預防型維運管理

- **決定服務等級目標 - Service-Level Objective (SLO)**
  - 99% 前台每秒User訪問延遲 < 300ms
- **測量服務當前狀態 - Service-Level Indicator (SLI)**
  - 目前狀況:99% 前台每秒User訪問延遲 < 75ms
- **決定服務等級領先目標**
  - 綠燈:99% 前台每秒User訪問延遲 < 150ms
  - 黃燈:99% 前台每秒User訪問延遲介於150ms 到 200ms
  - 紅燈:99% 前台每秒User訪問延遲 > 200ms
- **定期每月、每季Review領先目標**
  - 針對黃紅燈項目列出Action Item

SLA ·····▶ SERVICE LEVEL AGREEMENT

the agreement you make with your clients or users

SLOs ·····▶ SERVICE LEVEL OBJECTIVES

the objectives your team must hit to meet that agreement

SLIs ·····▶ SERVICE LEVEL INDICATORS

the real numbers on your performance

https://www.atlassian.com/incident-management/kpis/sla-vs-slo-vs-sli

背景: 91APP Queue System

91APP

https://www.91app.tech/blog/Architecture/91app-queue-system/

Case Study

狀況: 帳號註冊的驗證簡訊發送

情境:

會員在註冊帳號的過程中，需要驗證手機號碼。91APP 系統會發出驗證簡訊，使用者收到後輸入驗證碼，即可完成手機號碼驗證。

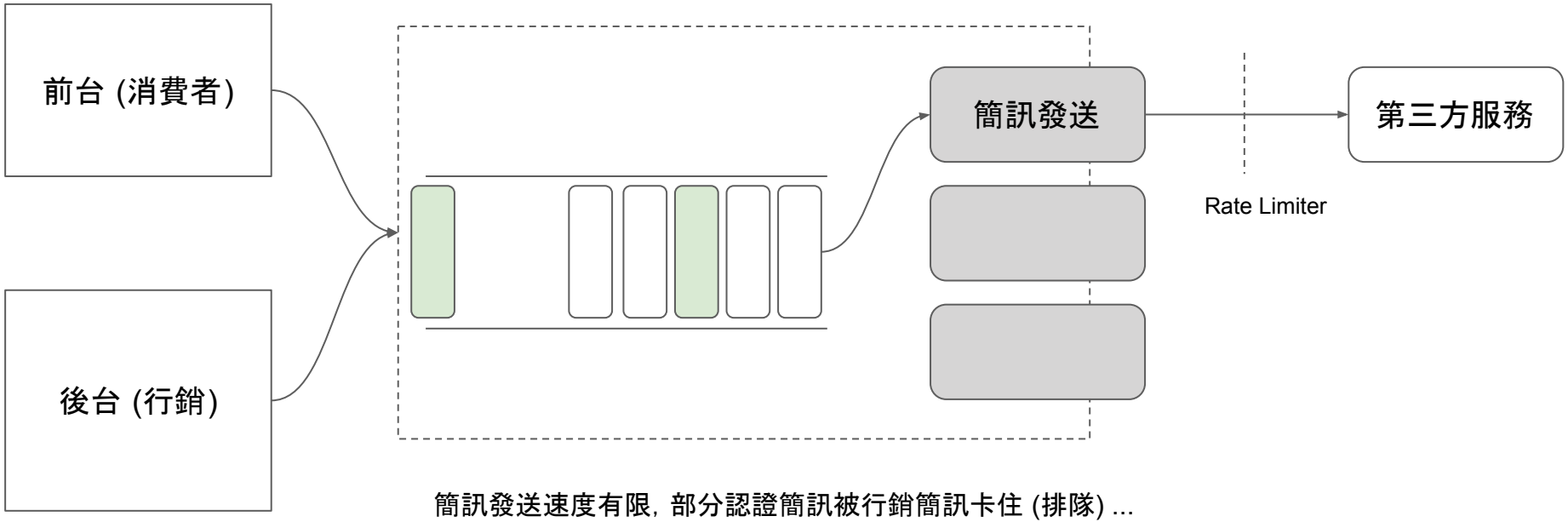
要求:

為了顧及使用者的體驗，系統必須在 5 sec 內完成發送的作業。

挑戰:

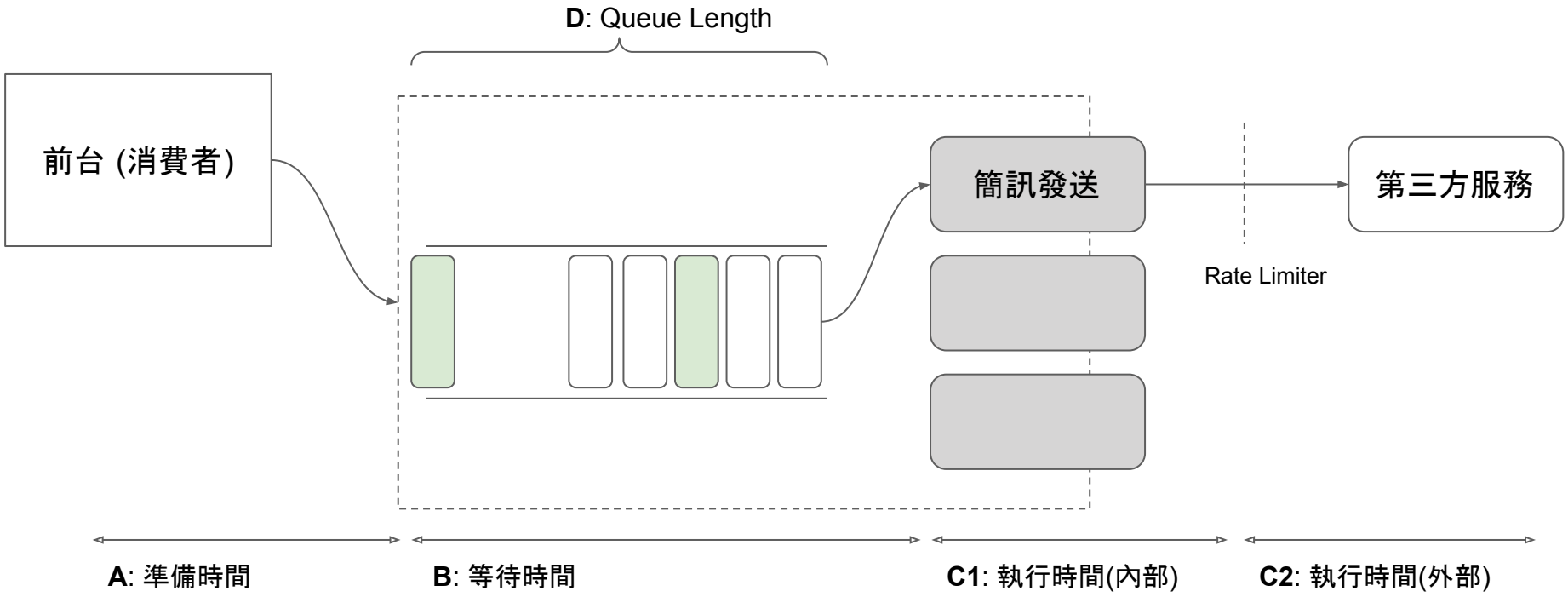
1. 對時間敏感的任務，必須盡量避免被干擾
   (但是過度配置專屬資源會提高成本)
2. 要顧及外部系統的處理能力
   (避免簡訊堆積在外部系統)
3. 維運團隊需要第一時間掌控狀況

Case Study: 大量發送行銷簡訊，認證簡訊受到影響延遲...



簡訊發送速度有限，部分認證簡訊被行銷簡訊卡住 (排隊) ...

SLI: 我們監控了什麼?



SLO: 我們期待的目標是?

業務上的說法:

消費者按下 "發送驗證簡訊", **5 秒內** 就要送到手機上

工程的視角:

(A) + (B) + (C1) + (C2: 簡訊商 -> 電信商 -> 手機端 的時間) < 5 sec

診斷: 有監控數據 (診) 才能找出效能瓶頸的所在 (斷)

91APP

如果:

(A) 的數值過高: 前端系統產生驗證簡訊的速度太慢;

(B) 的數值過高: 訊息在 Queue 裡面排隊花太多時間 **Queue 堆積太多行銷簡訊**

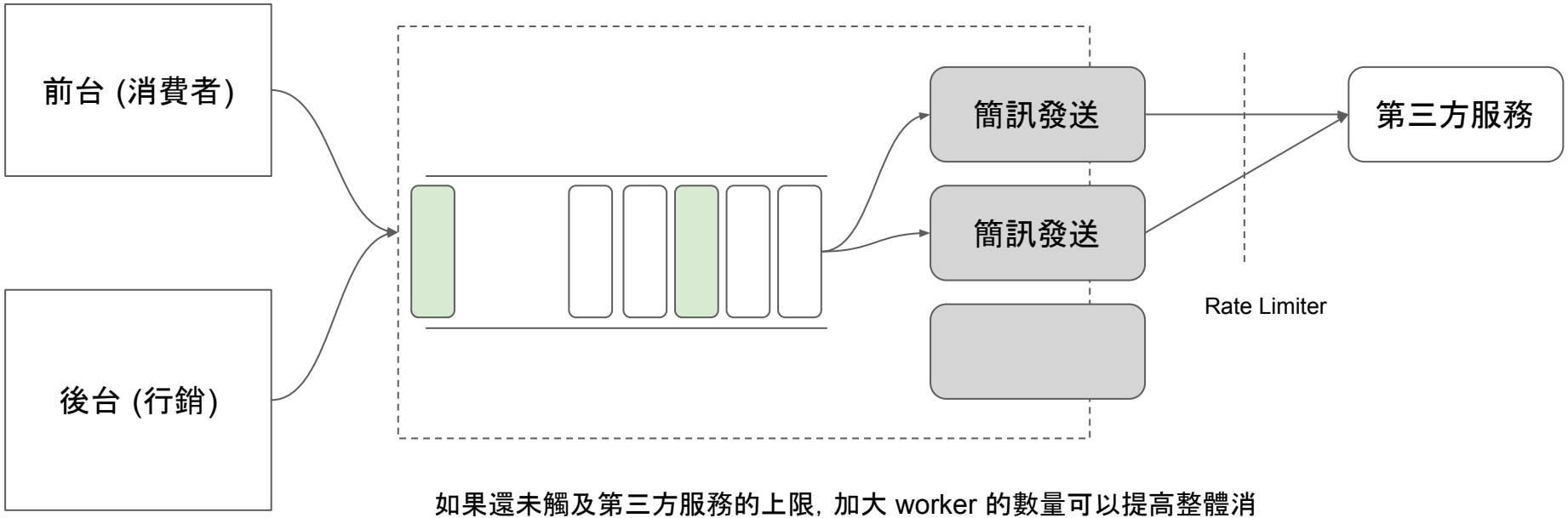
(D) 過高, 訊息堆積太多

(D) 不高, 訊息消化太慢

(C1) 的數值過高: 訊息消化太慢

(C2) 的數值過高: 第三方的處理效能太慢

Case #1, Message Worker Scaleout ...

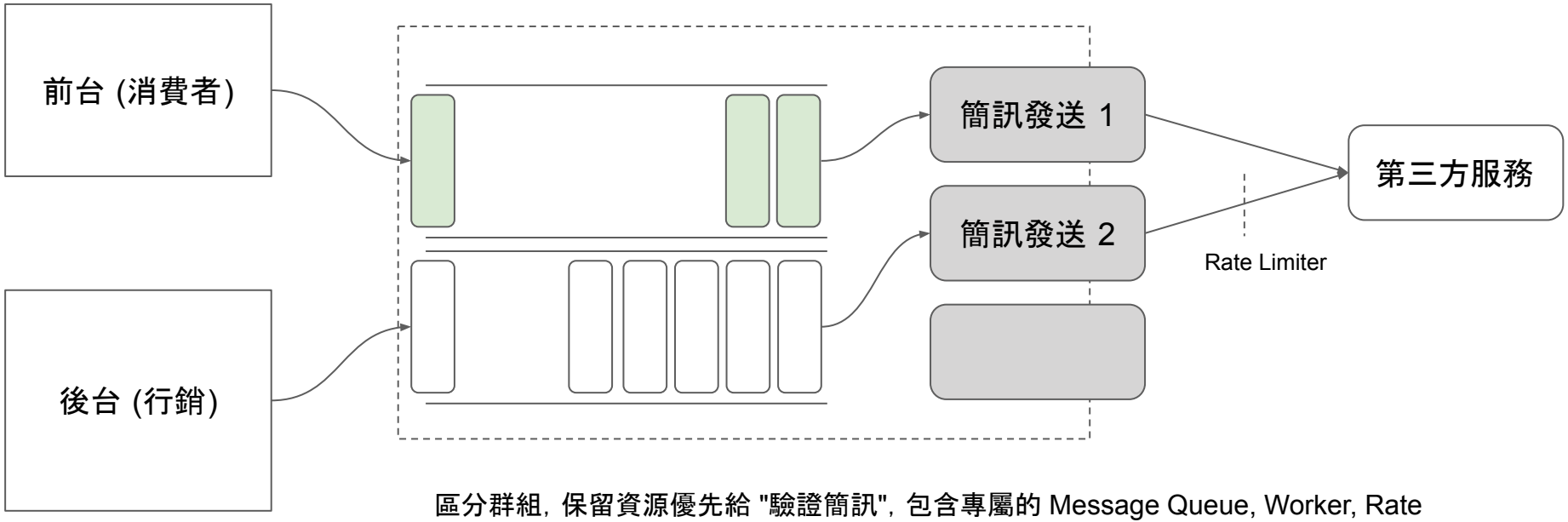


如果還未觸及第三方服務的上限，加大 worker 的數量可以提高整體消化速度。

代價: 耗用兩倍的運算能力

=> 錢沒花在刀口上，因為不需要被加速的行銷簡訊也被加速了...

Case #2, 降低 Queue Length 的方法 => 分群組



區分群組，保留資源優先給 "驗證簡訊"，包含專屬的 Message Queue, Worker, Rate Limiter …

資源花在刀口上，完全用於加速驗證簡訊的發送。

Think: 如果你沒有 "上帝視角" 怎麼辦?

想盡辦法, 把你需要的指標, 放到監控系統 內

實際上可能的狀況會是這樣...

91APP

驗證簡訊延遲？那就加開機器（前台）啊...

驗證簡訊延遲？那就加開機器（Worker）啊...

整個非同步作業都很慢？提高 Message Queue 的規格...

IT / Infra 回報: 以上的處理都沒有效果 ...

Develop Team: 這 code 誰寫的？想辦法優化他 ...

…

...

目標導向: 從開發的第一天, 就弄清楚你期待的 SLO ...

91APP

範例: 消費者按下 "發送驗證簡訊", **5 秒內** 就要送到手機上

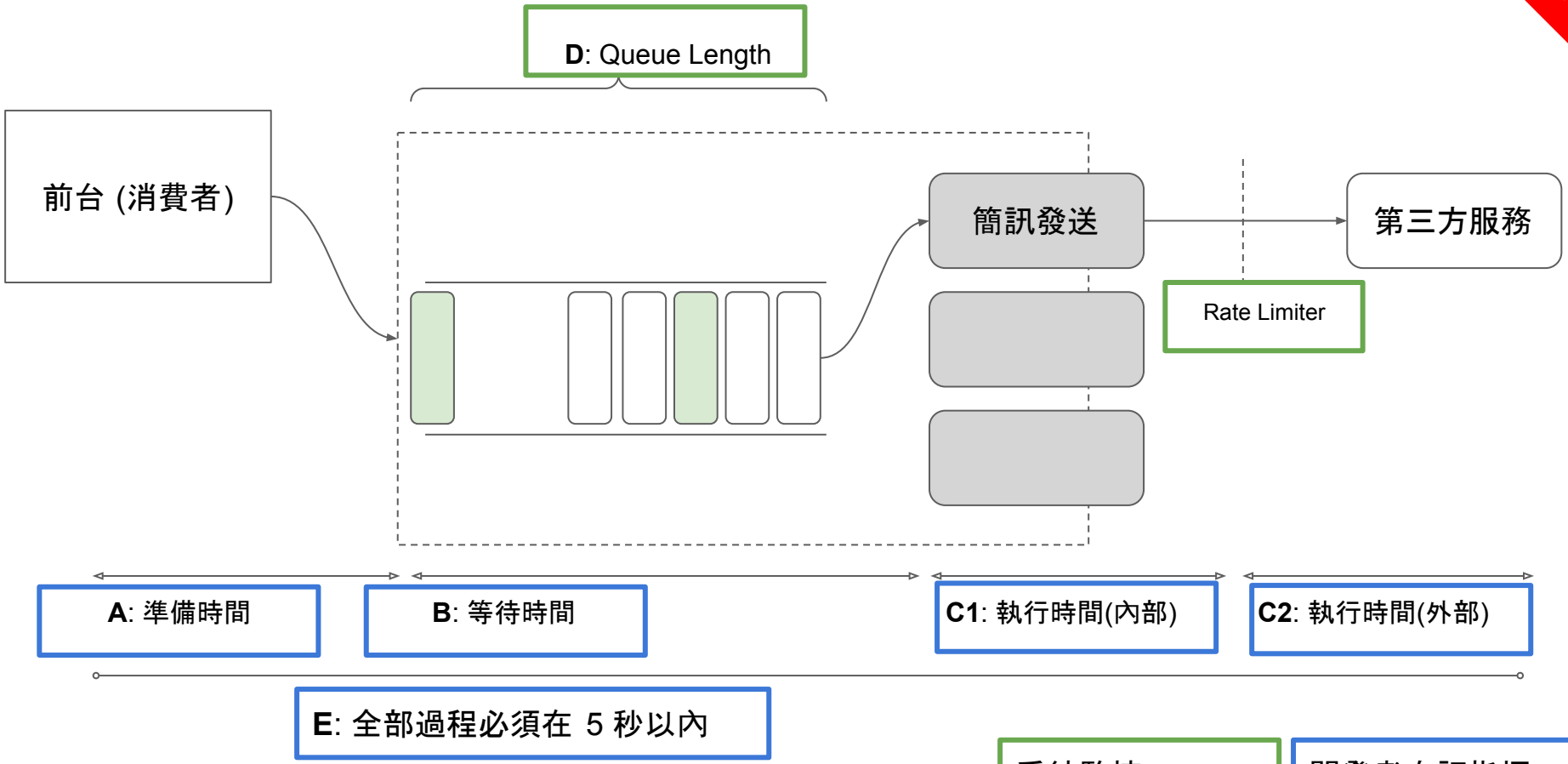
拆解: 這 5 秒內要完成哪些事情？

盤點: 哪些指標我能掌握？

盤點: 如何處理我無法掌握的指標？

行動: 善用監控的服務, 透過 logs 分析, 或是 metrics API 來達成

SLI: 我們監控了什麼?



想辦法讓自己擁有 "上帝視角", 湊齊你需要的指標

91APP

挑選合適的監控平台 (ex: AWS cloud watch, Azure application insight, ELK, …)

找尋適當的系統指標收集工具

RECAP

淺談系統監控
與
AWS CloudWatch 的應用

Rick Hwang
AWS User Group Taiwan
Jun 21, 2017

RECAP

如果我想要看的指標
CloudWatch 沒有怎麼辦？

RECAP

CloudWatch Custom Metrics

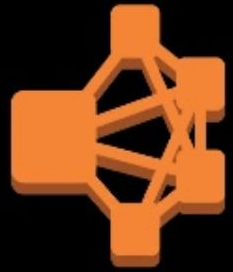
- 兩個常見的需求
  - EC2 Memory Utilization
  - EC2 Disk Utilization
- How
  - AWS CLI / SDK: put-metric-data
  - AWS CloudWatch Logs
  - Third Party Agents:
    - Collected
    - Telegraf

```
aws cloudwatch put-metric-data \
   --metric-name mem \
   --namespace /CWL-Demo/App \
   --unit Percent --value 23 \
   --dimensions InstanceId=1-23456789,InstanceType=t2.small
```

RECAP

監

Watch
Monitor
Observe
Measure
Dashboard



Targets

控

Control
Command
Handle
Manage
Console

高效率 + 精確度 的要求

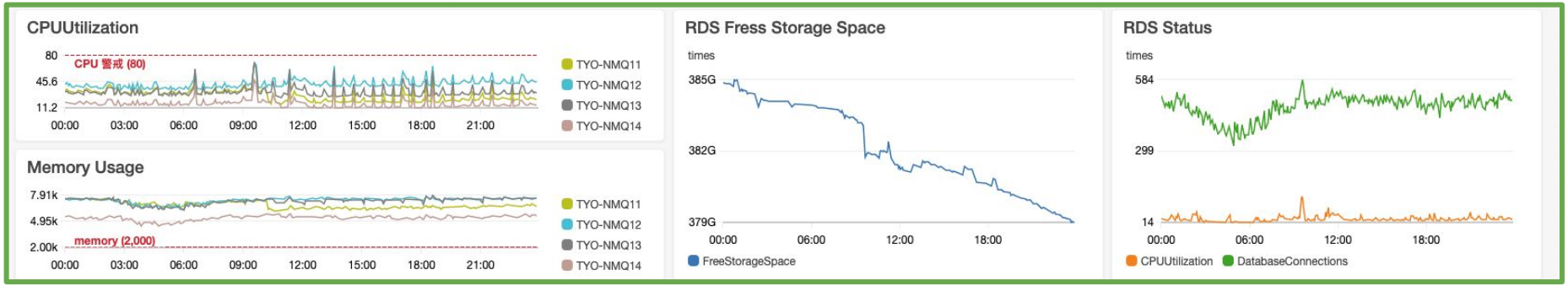
91APP

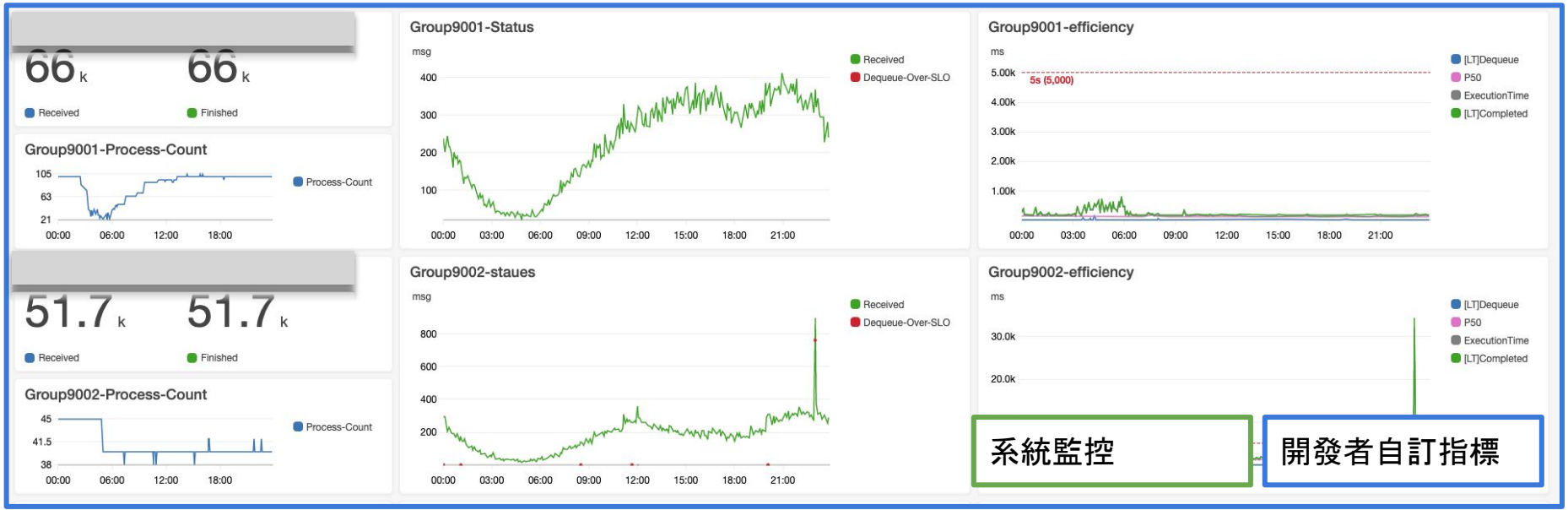
比起考試考 100 分，難度更高的是想考幾分就考幾分

Microsoft 面試考題: 讓 CPU utilization 顯示 sin wave

2020 雙十一 監控 dashboard: 我們監控了什麼？

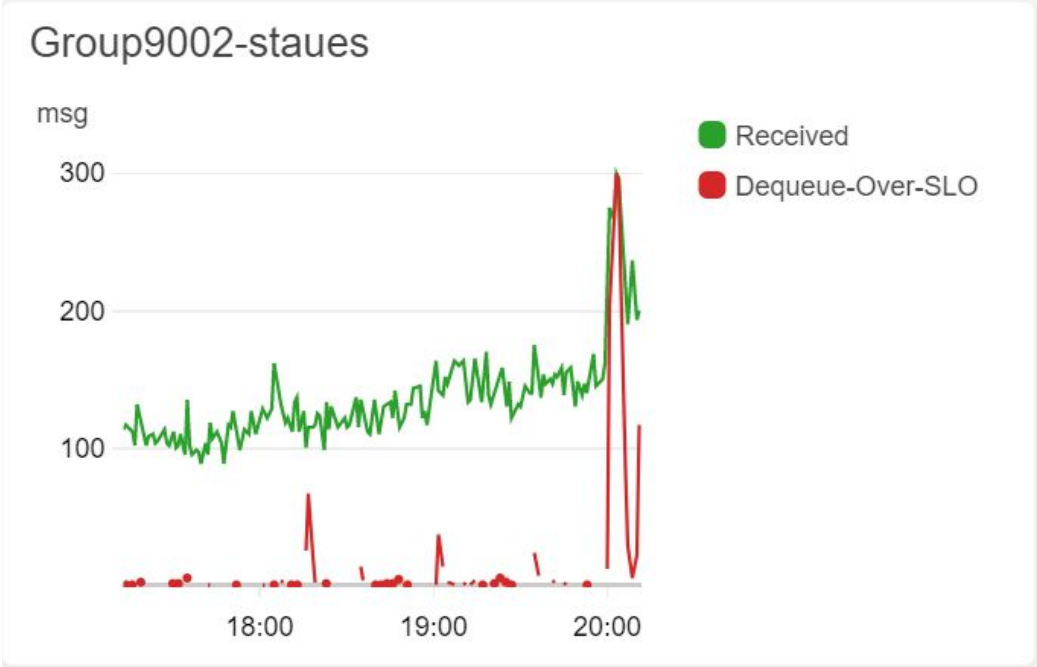
91APP

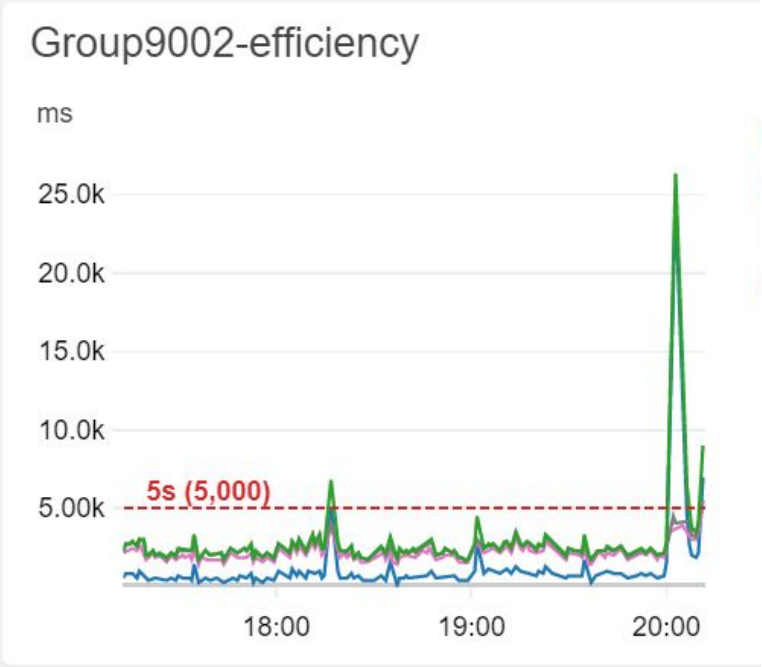




指標定義:

- Received: Task 從 Message Queue 取出執行的數量統計
- Dequeue-Over-SLO: 所有從 Queue 取出的 Task 中, 取出當下就已經超過 SLO 要求的數量
  (**A** + **B** > **5** sec, 持續 **3** 分鐘狀況沒解除就會發送警告通知)

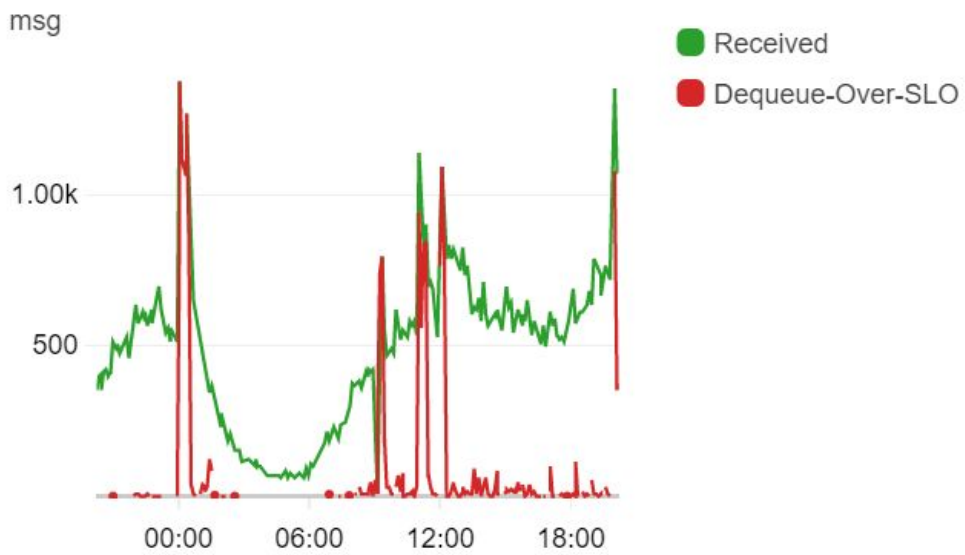




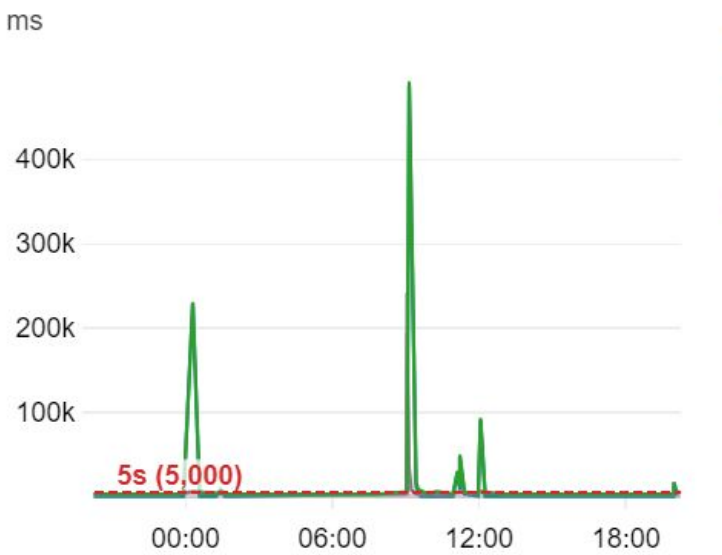
指標定義:

- Efficiency: 每個任務從 create task 開始，到 task complete 為止的時間
  (**A** + **B** + **C**)

Group9002-staues



Group9002-efficiency



Think: 有了 "上帝視角" 之後?

面對各種狀況的應對方式

Case #1, 突然有大量的簡訊發送任務, 都超出 SLO 的要求...

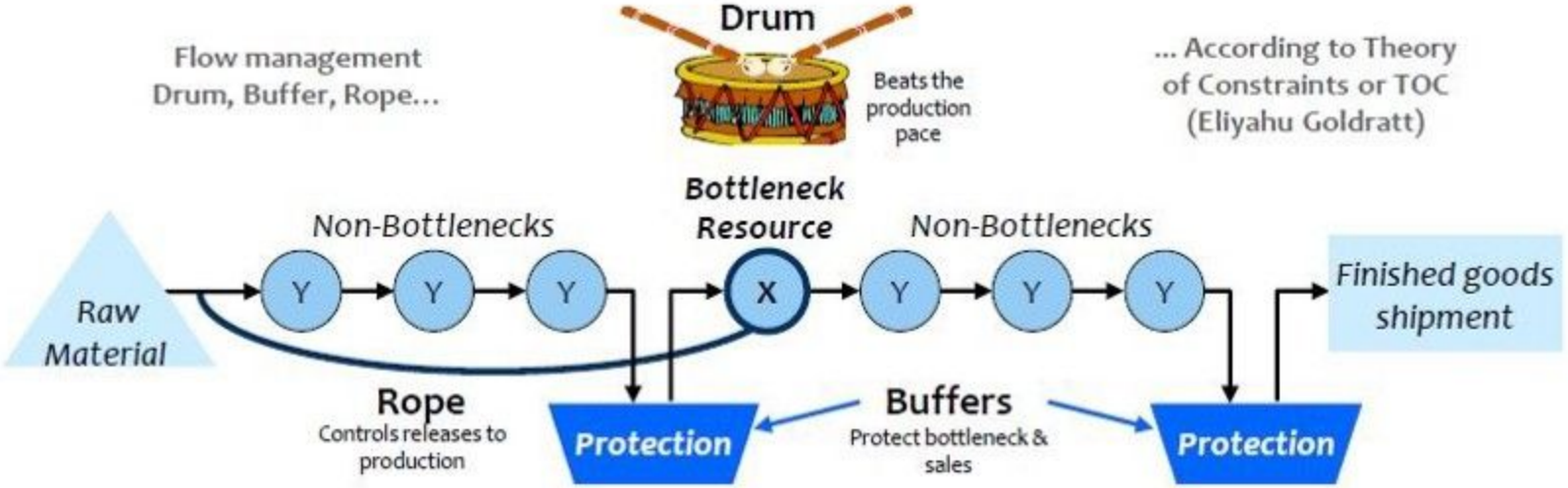
指標定義:

- Received: Task 從 Message Queue 取出執行的數量統計
- Dequeue-Over-SLO: 所有從 Queue 取出的 Task 中, 取出當下就已經超過 SLO 要求的數量 (**A** + **B** > **5** sec)

Q: 碰到這種狀況，你會...?

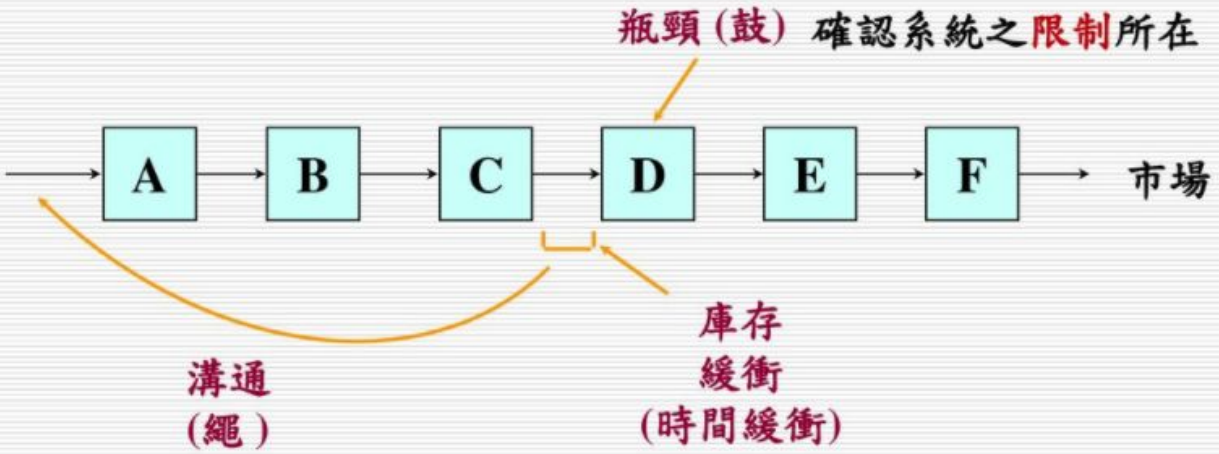
1. 加開 Worker, 增加 instance 個數
2. 改善 Queue 的效率
3. 改善 Task 的效率
4. 改善來源端 (Task Create) 的效率
5. 降低來源端 (Task Create) 的速度
6. ...

從限制理論來看 (TOC, Theory Of Constraints)



『高效率團隊』如何運用限制理論 (Theory of Constraints) 於軟體開發

鼓、緩衝、繩 (1/3)



鼓、緩衝、繩 (2/3)

鼓 (Drum)

控制整個系統的生產節奏(速度)。生產系統中都會有某個控制點，用以控制生產流量的大小，而瓶頸點即為整個系統的最佳控制點，稱為鼓。

緩衝區 (Buffer)

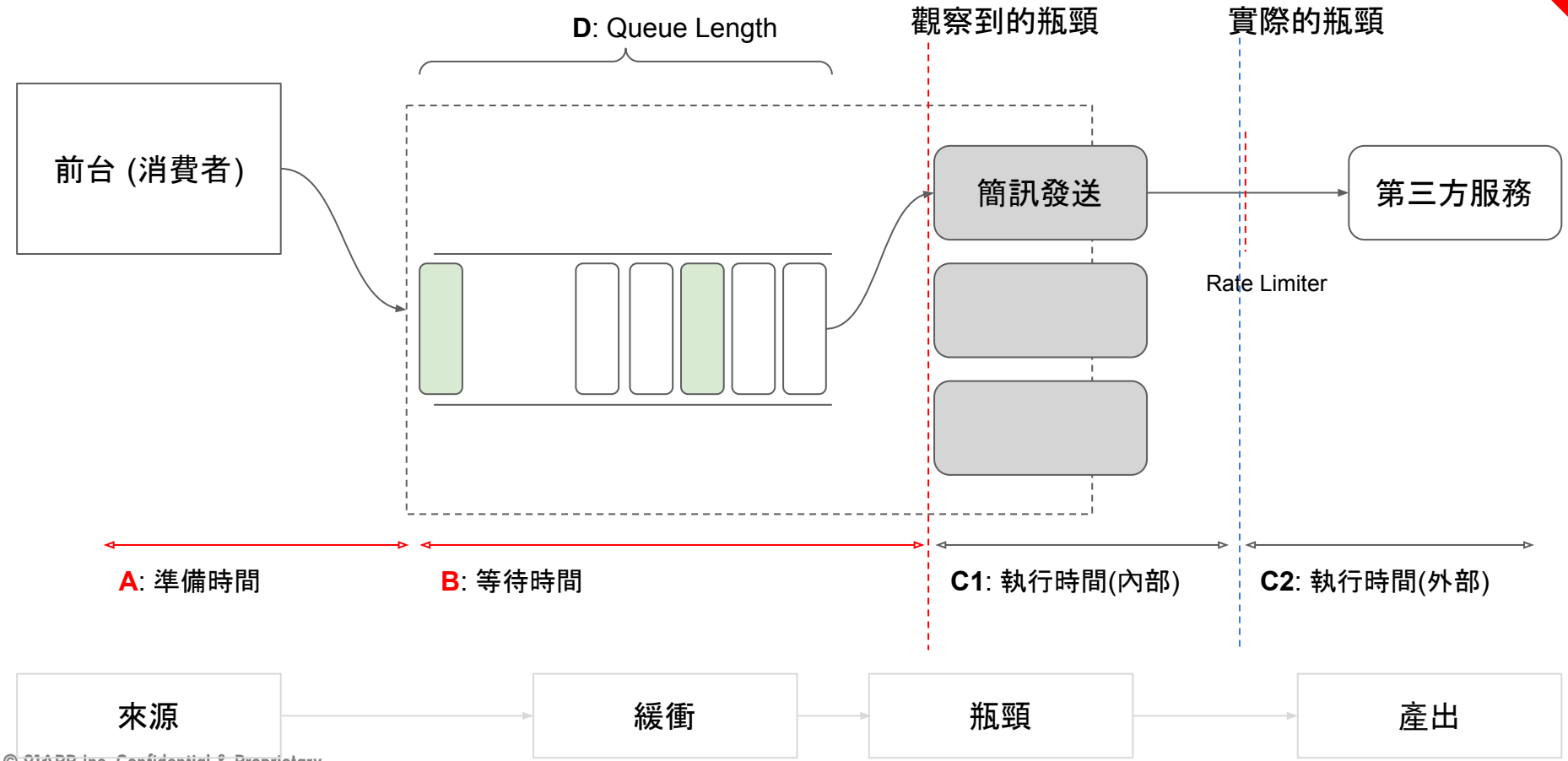
使系統能在不同的狀況下正常的運作。由於系統會因為各種變異造成系統的不穩定，而緩衝區的目的就是用來保護系統使其正常的運作，但並非所有的機台都需要，不過瓶頸機台前一定要設緩衝區。

繩子 (Rope)

用來確認整個系統的運作能和瓶頸點同步。瓶頸點必須提供所需的量等等的生產資訊給上游的工作站，以決定適當的投料時間，避免生產過多造成存貨的堆積。此種溝通、資訊回饋的情形如同繩子。

鼓、緩衝、繩(3/3)

先從數據指標，還原實際的狀況

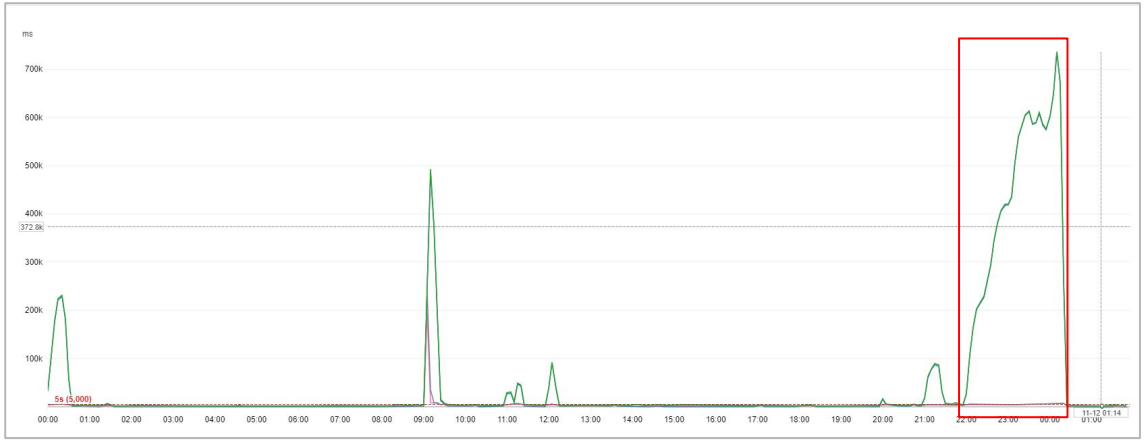


能夠選擇的做法

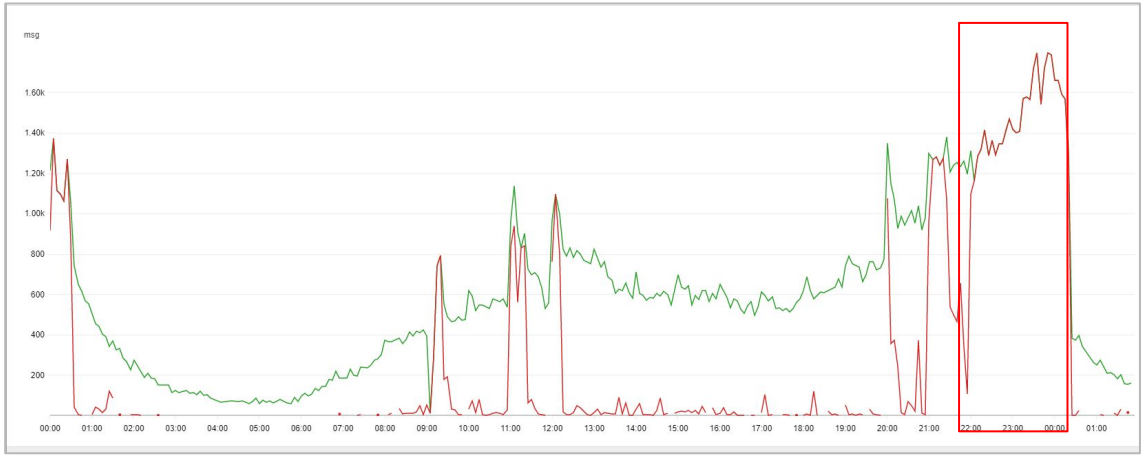
91APP

1. 前端切換其他方式，進行手機號碼驗證 (例如撥號驗證)
2. 降低 SLO 的要求
3. 擴大 91APP 與簡訊商的安全容量 (與第三方廠商確認後，放寬 Rate Limit 限制)
4. 擴充 Worker 的處理能力
5. 改寫 Task, 做好最佳化改善執行速度

Case #2, 交易相關任務隨著流量增加，開始出現長時間的延遲



Efficiency



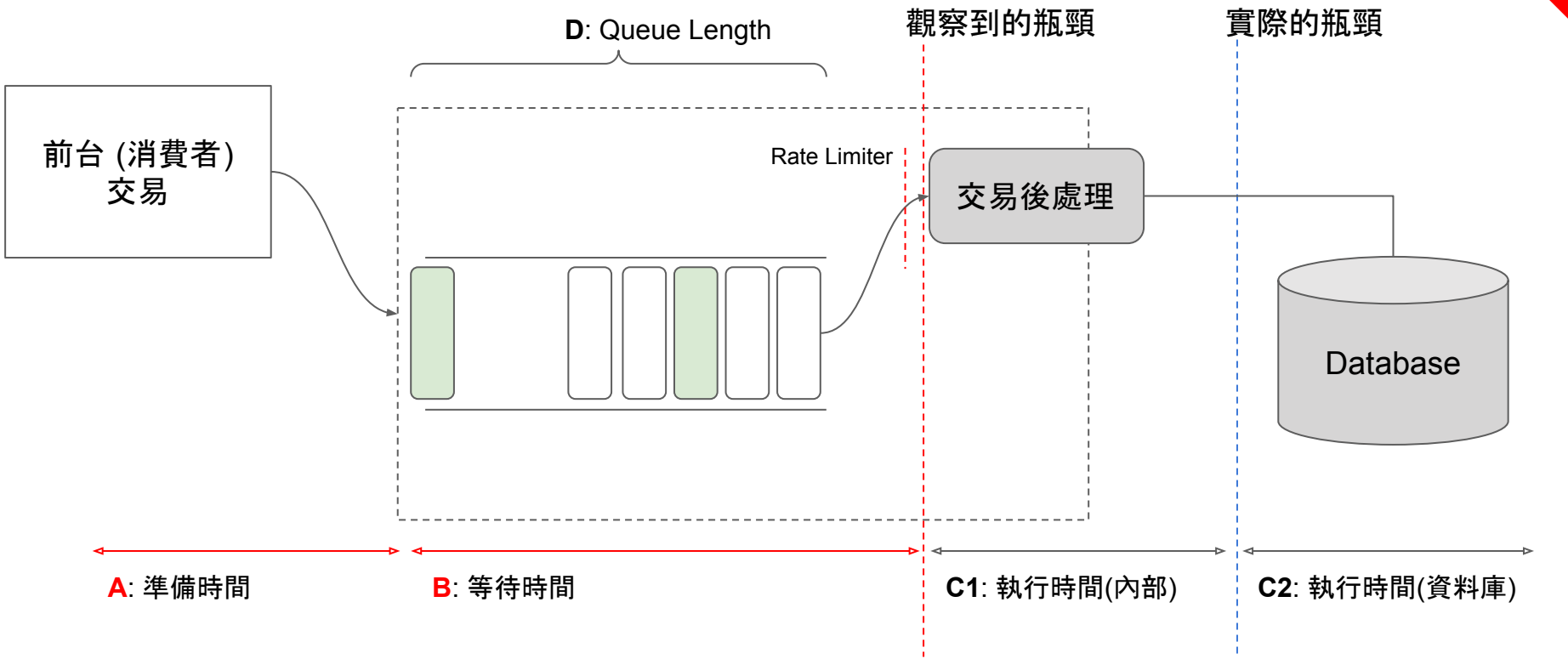
Received / Dequeue-Over-SLO

1. Task 執行的效率不佳

2. 導致消化速度低於 產生速度，
於是 task 開始在 queue 累積，
光是排隊時間就超過額定 SLO

先從數據指標，還原實際的狀況

RECAP

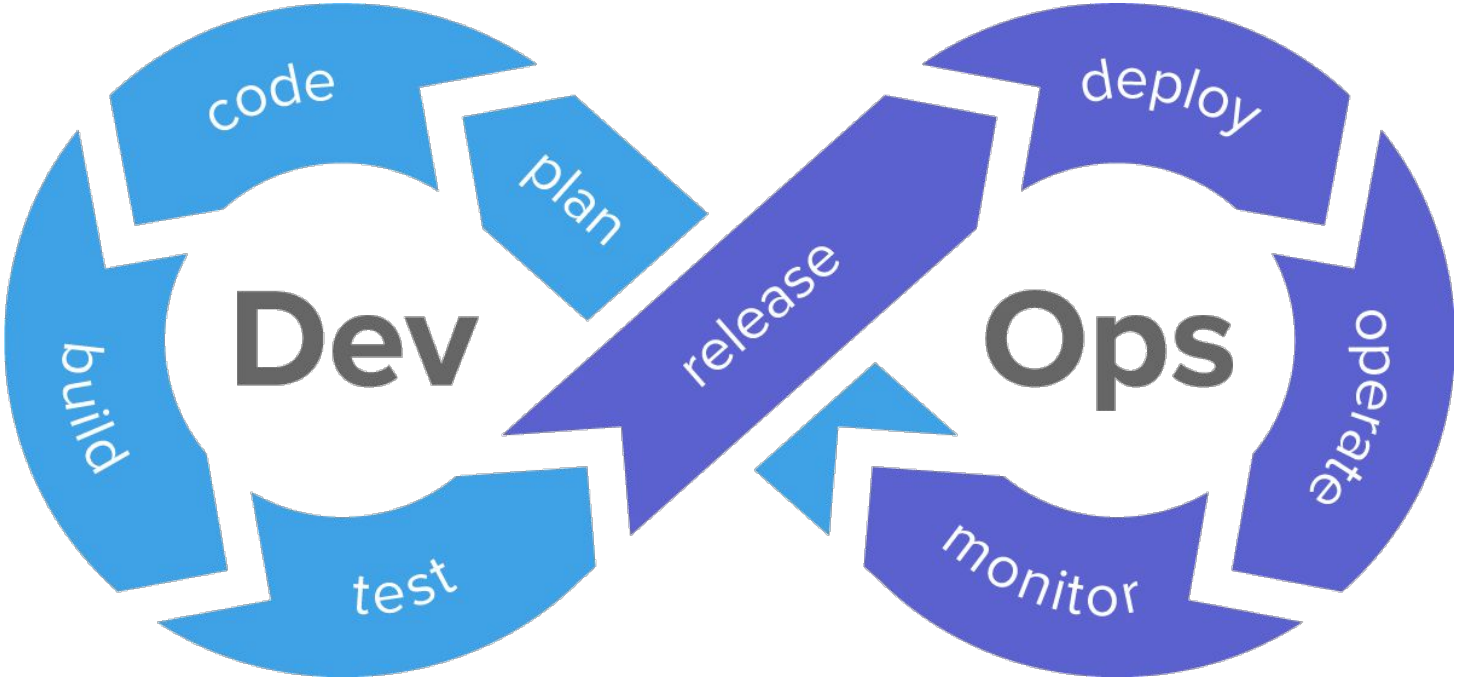


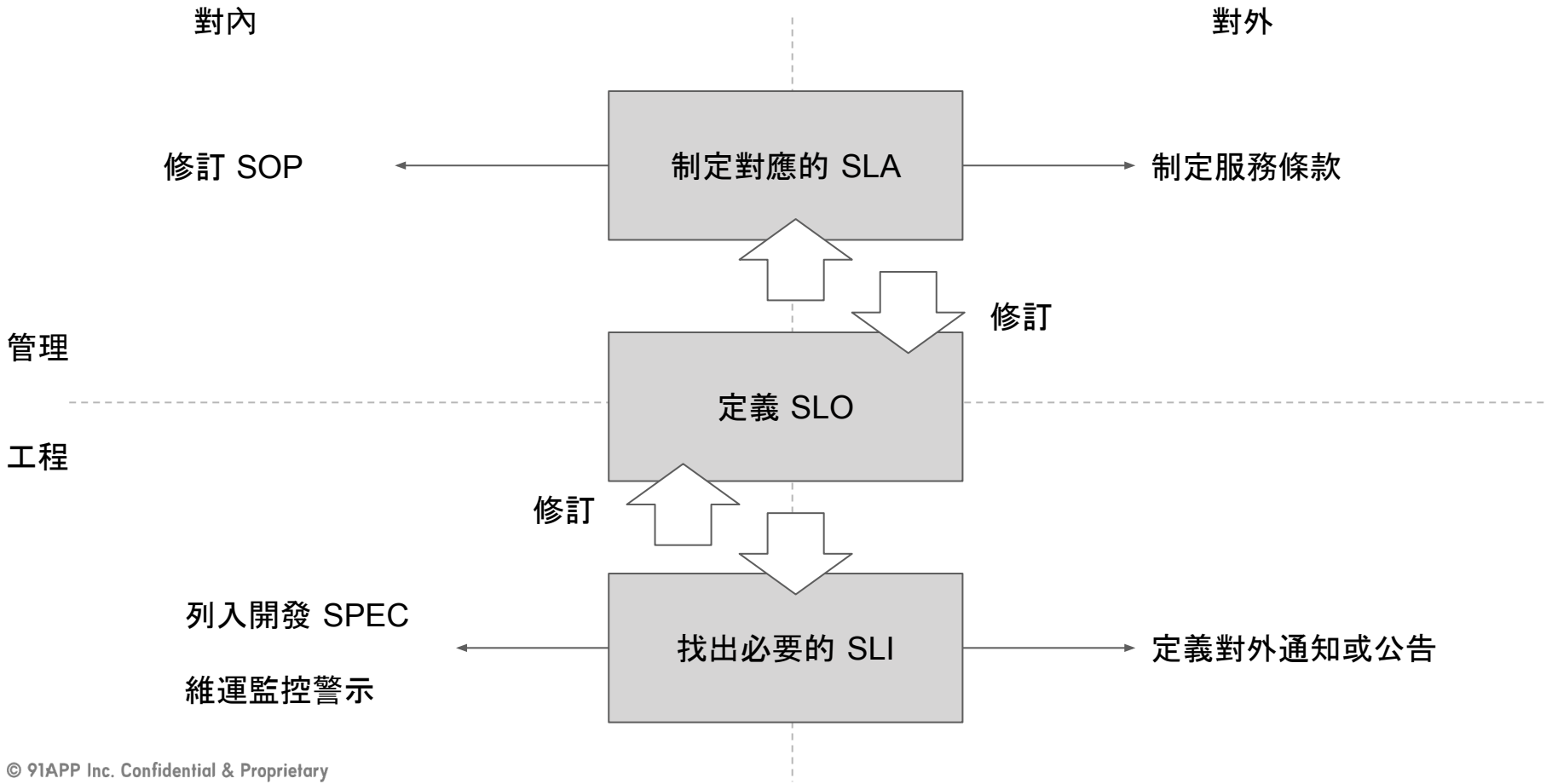
來源 → 緩衝 → 瓶頸 → 產出

能夠選擇的做法

1. 降低 SLO 的要求
2. 擴充 Worker 的處理能力
3. 改寫 Task, 做好最佳化改善執行速度
4. 擴大資料庫的處理能力
5. 限制 Worker 的處理能力（避免過度影響線上的交易）

結論: 落實 DevOps 的精神





處理經驗小結

91APP

1. 整體的平衡 > 單點的最佳化 (並非越快越好，成本與效益的平衡也需要考量)
2. 精準的控制能力 > 單純提高運算能力 (有時候，跑的慢一點比跑的快還好)
3. 無法滿足 SLO, 也有可能是 SLO 定義的不洽當，回頭思考 SLO 是否合宜？
4. 越高的 SLO 需要越高的維運成本；依據需求來決定適當的 SLO, 而非定義一個高不可及的目標。
5. 以目標 (SLO) 導向，來補足你缺乏的環節

   (例: 沒有對應指標，就自己開發，靠 SDK 寫入指標)

   (例: 需要精準控制速度，就自己尋找，或是自己開發 Rate Limiter 服務)
6. 以目標 (SLO) 導向，從開發的第一天就決定 SLO, 搭建能讓你隨時掌控關鍵指標的系統