Ch26: 資料完整性

jnlin @ PIXNET
2018.4.26

從使用者出發的資料完整性定義

- 因為 UI Bug 導致資料還在，只是不顯示
- 因為 meta data 遺失導致出現維修公告
- 「正常維持服務與資料的存取」

舉個例子

- 每年資料會遺漏一次，但是無法恢復
- 每年資料會遺漏好幾次，但每次都能在**使用者發現**前**恢復**
- 如何「主動探測」與「快速修復」是重要的

服務定位會影響資料完整性

- 五個服務最佳化的面向
  - 可用率
  - 延遲
  - 規模
  - 創新速度
  - 隱私

- 如果只考慮創新速度，開發者為了盡可能節省時間，可能會選擇熟悉的API，以致選到錯誤的架構(儲存媒體)

備份與封存

- 沒有人真的想要備份資料;他們只想恢復資料
  - 驗證資料回復比備份重要太多
- 發生問題後的復原時間，以及可以遺漏多少最新資料
- 設計的是「復原」系統，而非備份系統

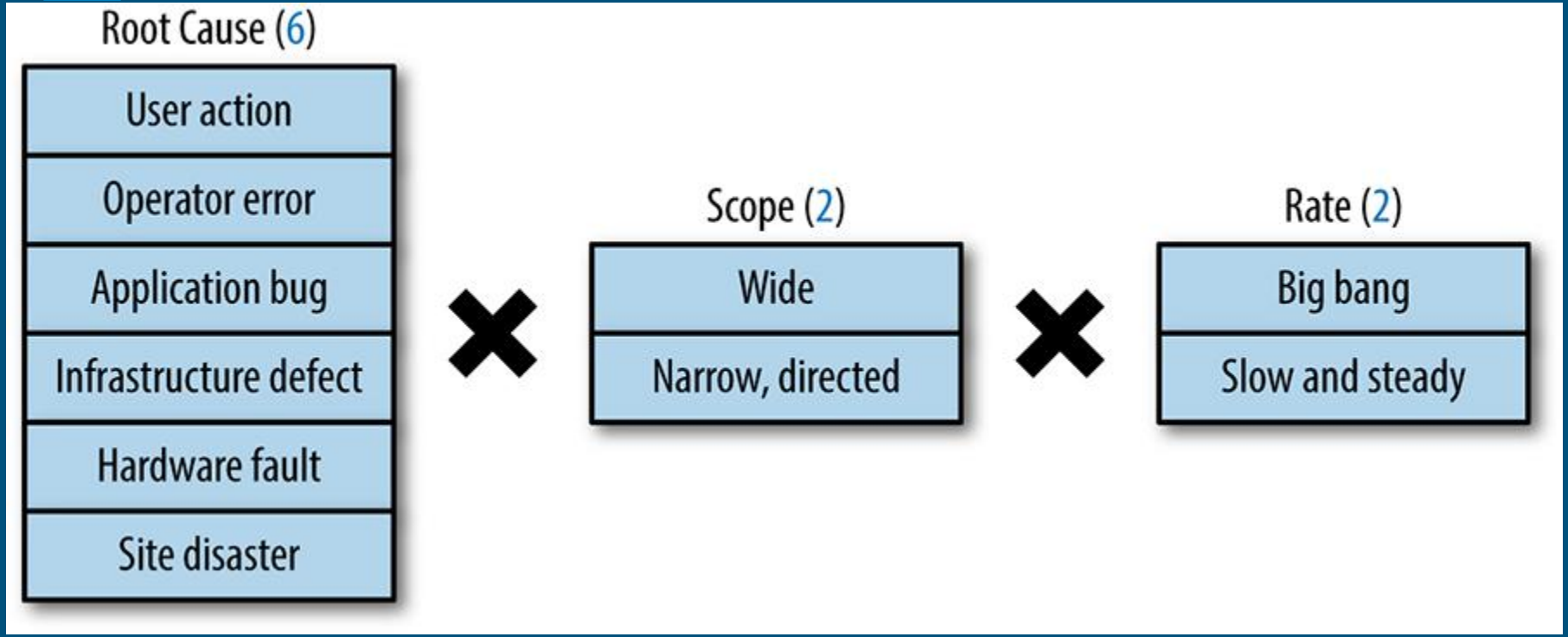
SRE 的目標？

- 資料完整性＆資料可用性
- 再強調：資料完整但是不可用，對使用者來說是沒有意義的

備份就像納稅一樣

- 長期消耗時間和資源，但沒辦法帶來任何現在可見的好處
- 出問題的時候會造成嚴重後果
- 不應該強調應該納稅，還是應該強調稅可以提供什麼保障
  - 產品團隊應該對不同的失敗情境定義一系列資料可用性（ SLO）的標準
  - 團隊需要定期演練，確保他們有能力完成 SLO

造成資料遺漏的事故類型



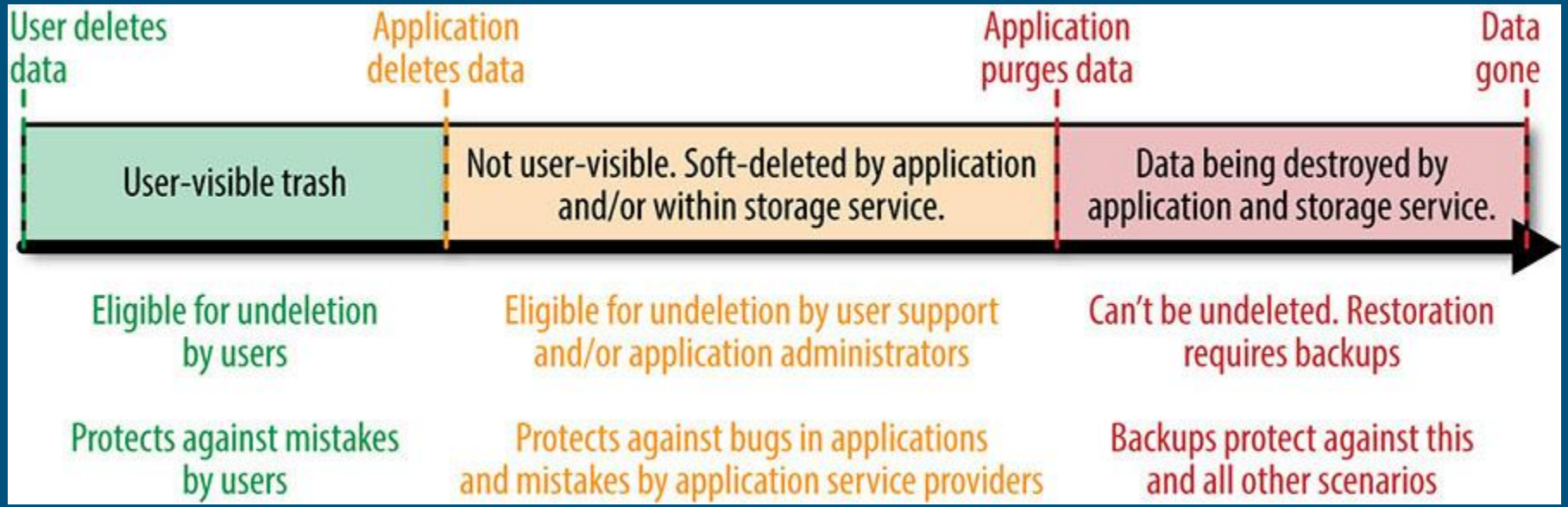
分級備份與時間點恢復

- 軟體 BUG 在線上環境中，過了幾個月才發現造成資料遺漏
- 時間點恢復，在大規模的系統上成本太高
- 策略：最近的資料採用 snapshot 來達成時間點恢復，完整備份或增量備份美兩天進行一次

複製機制與冗餘不代表可恢復性

- 簡單的例子:程式寫壞了, 複製機制執行程式刪除的指令
- 匯出:需要檢查資料匯出是不是正確
- 針對不同的失效情境, 準備多樣的恢復副本
- 資料備份儲存的時間(經過很久時間才發現資料不見了)

Google SRE 保障資料完整的手段



軟刪除（soft-delete）

- 資料是「標記被刪除」，前台看不到，只有管理後台可以看到
  - 垃圾桶機制
- 情境
  - 使用者的誤刪
  - 帳號被劫持
  - 裝了某個有問題的插件
  - 不熟悉的開發者處理刪除資料的工作
- 其他機制：懶刪除（Lazy-delete）、修訂歷程

備份與相關的復原方法

- 備份不重要，重要的是如何復原
- 關鍵問題：在一次資料復原中，允許損失多少資料；要花多久時間復原資料
- 備份資料應該儲存多久？
  - Google 定在 30~90 天，並利用預警系統的建立來保證備份週期落入這個區間
- 分級備份：用合理成本滿足資料復原的需求
  - 可以快速恢復的資料（10分鐘內）
  - 需要數小時恢復的資料（資料複製）
  - 冷儲存：需要數天恢復的資料（離線儲存）

資料量大小造成的問題

- TB 級與 PB 級資料大小差異
  - 700PB，用 SATA 的效能，檢查基本資料完整性需要 80 年
- 平行進行備份與還原工作
  - 正確平衡資料分段
  - 保證每個片段之間的獨立性
  - 避免相鄰平行作業的資源搶佔
- 限制垂直資料量：定期建立可信的資料，之後作增量備份與還原

早期預警

- out-of-band 資料校驗器
  - 寫 Code 需要自動化測試
  - 資料也需要自動化的校驗器
- Gmail 的資料校驗器
  - 一系列手冊，說明如何處理某個校驗失敗的警告
  - 類似 BigQuery 的查詢工具
  - 監控儀表板
- 團隊負擔不起成本
  - 專屬團隊提供框架，由 產品團隊負責業務邏輯

早期預警

- 自動化恢復機制
  - 持續測試恢復機制
  - 自動化測試恢復機制

- Checklist
  - 備份資料是否完整、正確
  - 是否有足夠的運算資源來正確完成整個恢復過程
  - 恢復過程是否在合理的時間完成
  - 是否有紀錄狀態訊息
  - 復原過程是否依賴某些無法控制的元件（例如不是 24x7隨時可用的異地儲存媒介）

案例分析：Gmail

- 2011 年 2 月 27 日（日）
- Gmail 大量的元件同時失效，導致資料遺失
- GTape：為 Gmail 量身定做的磁帶備份系統
  - 計算出大部分使用者恢復的時間
  - 需要多久可以恢復全體使用者
  - 恢復超過 99% 的資料的時間

案例分析:Google Music

- 2012 年 3 月 6 日(二)
- 使用者回報音樂無法讀取
- 檢測發現，有 60 萬筆音樂遺失，約影響 2.1 萬使用者
  - 資料量超過 1.5PB
  - 超過 5000 卷磁帶
- 同時找問題並嘗試回復資料

案例分析:Google Music

- 資料刪除作業造成問題
- 資料量太大，導致 Race Condition
- 有 160,100 筆音樂在磁帶中找不到備份
  - 商店出售的音樂，有原始檔案
  - 透過自動重新上傳機制解決

小結

- 資料可用性是重點
- 用自動化軟體測試的方式來測試資料可用性與可恢復性
- 關注目標，而非手段；多層防護涵蓋最廣的失敗場景
- 持續執行、不斷演練
- 自動資料檢測，在使用者發現之前完成恢復作業


Thank You