閱文集團內容系統統一與整合

大嘴
xvhfeng@gmail.com

阅文集团
CHINA LITERATURE

當時情況

- 腾讯文学：java ＋ mysql
- 创世：java ＋ php ＋ mysql
- 盛大文学：c＃ ＋ mysql ＋ oracle ＋ sqlserver

選型列表

- 應用程序：java開發框架（自研）
- 隊列服務：完成雙向消息通知和業務處理（自研）
- 調度服務：完成定時計算、實時業務處理（自研）
- ID生成器：根據業務需求提供全站唯一ID，供數據路由（自研）
- DFS：分佈式內容存儲，存儲章節、圖片、音視頻（自研）

- MySql：基本數據存儲，一主多從，業務切分
- Redis：緩存，v2版本目前正在開發lest替代它
- Solr：索引，v2版本目前正在開發相關技術替代
- Nginx：http服務和負載均衡
- Aura：圖片裁切web服務，目前JNI技術試驗階段，後期全部用c

分佈式文件系統

DFS-需求

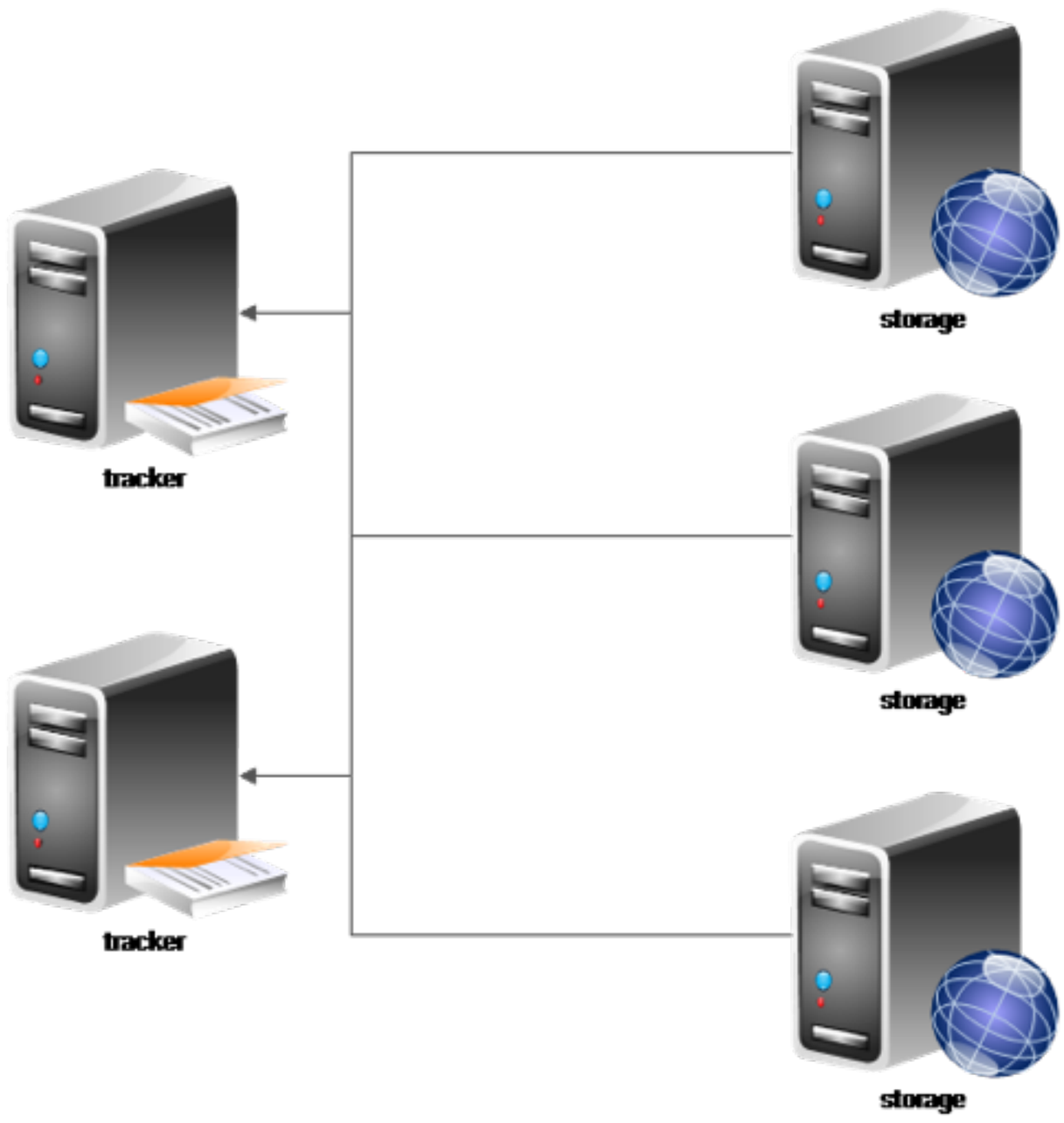
- 横向擴展
- 自動化：數據平衡、數據恢復
- 性能最大化：無slave，全主模式
- 數據一致性
- 支持頻繁的更改操作
- 支持小文件（幾B），大文件（200mb）

DFS-選型

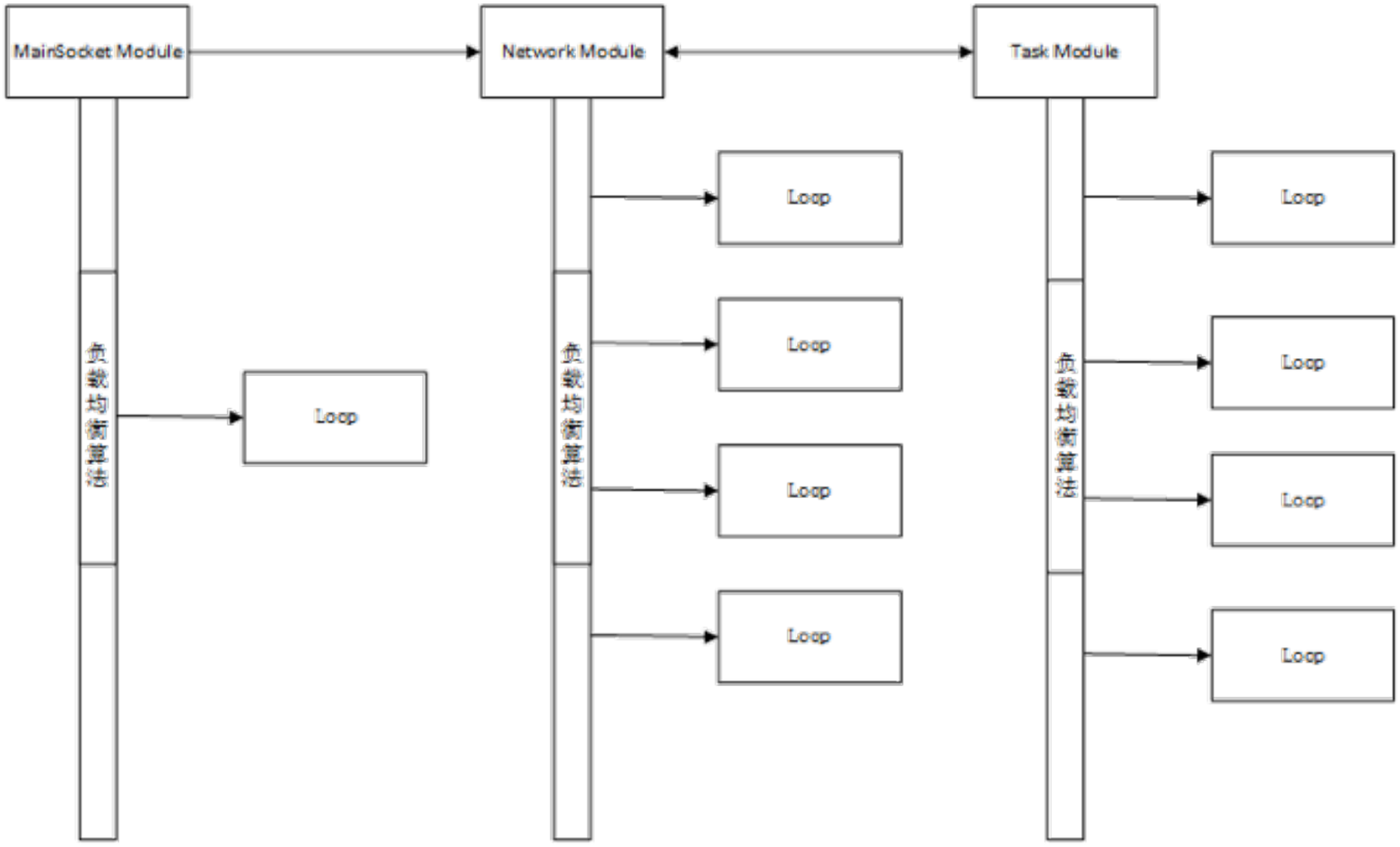
- Fastdfs：
  - chunkfile版本控制不太滿意
  - group為鏡像的粒度太大，擴容不好控制
- CFS（騰訊內部）：
  - 小文件支持不好
  - 大文件849mb 寫13.777 讀 25.45 刪 5.565
  - 小文件 478mb 寫28m37s
- 掛載運行，沒有管理層，客戶端還經常莫名其妙的爆內存
- 剩下的：
  - redis，mongodb…

最核心：對於更改幾乎所有的DFS支持都不太好

DFS-架構



DFS-核心算法



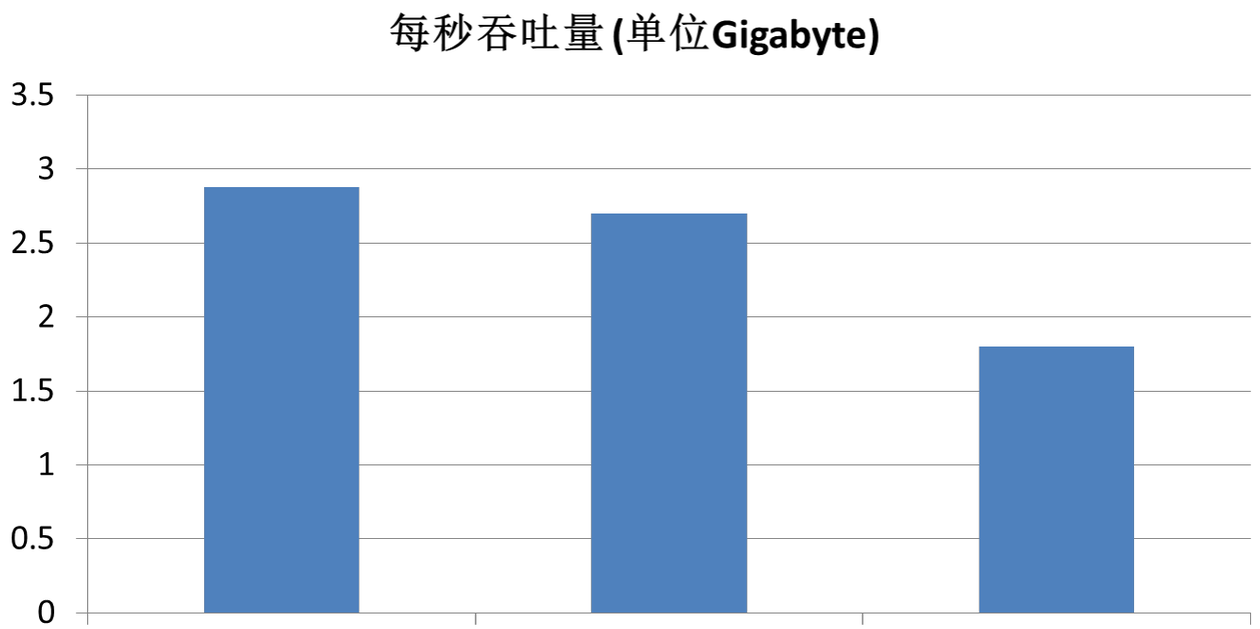
- 線程的邏輯分層
- 線程調度算法（無鎖編程算法）

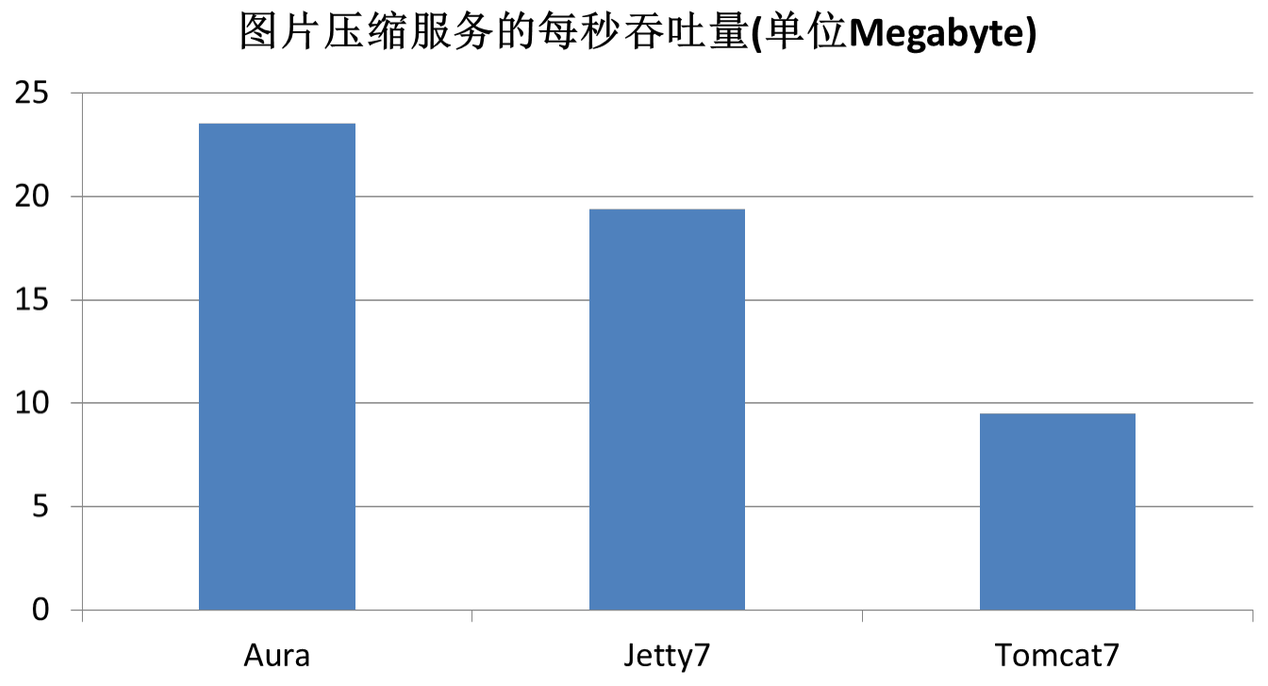
DFS-文件存儲

- 文件空洞
- 多master的版本問題
- 文件版本、context版本，操作版本問題
- vector clock算法
  - 每個storage有一個版本算法
  - 每次操作和磁盘context一個版本
  - 時間戳也是版本的一個部分
  - 修改靠這個版本號來為此一致性
- 實際長度 * 120 % = 總長度
- hashcode簽名

Aura-DFS的訪問端

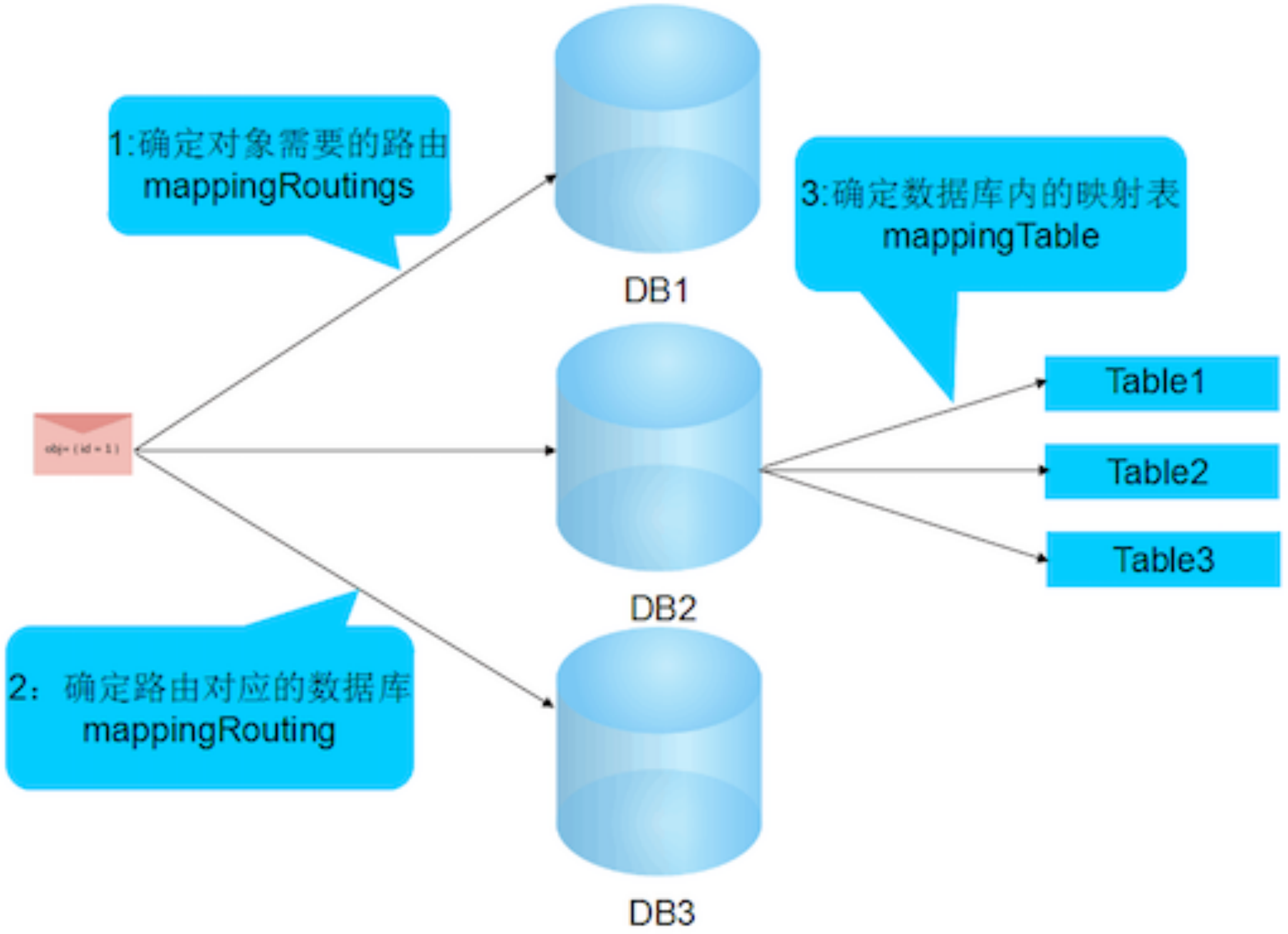
- 主要用來提供DFS的HTTP訪問服務，比如圖片，視頻
- 默認Reactor事件模式
- 邊緣觸發，效率更高
- 高並發支持，異步IO+線程池，10k+/s qps
- 支持RESTFUL風格API
- 無鎖設計，支持資源緩存
- 自帶監控接口





數據路由實現與策略

數據路由實現



數據路由實現-傳統id

- int類型，數據庫自增
- int數據庫自增，自定義步長
- GUID / UUID
- 自定義String

數據路由實現-需求

- 唯一，必須全站唯一
- 短，盡可能的短
- 生成速度足夠快
- 運算簡單，對於id的業務擴展簡單容易
- 必須附帶業務信息，比如時間、類型
- 部分信息可以自定義，特別是可以自定義路由標識
- 不僅僅適合機器識別，更重要的是人類可識別
- 對索引盡可能的友好

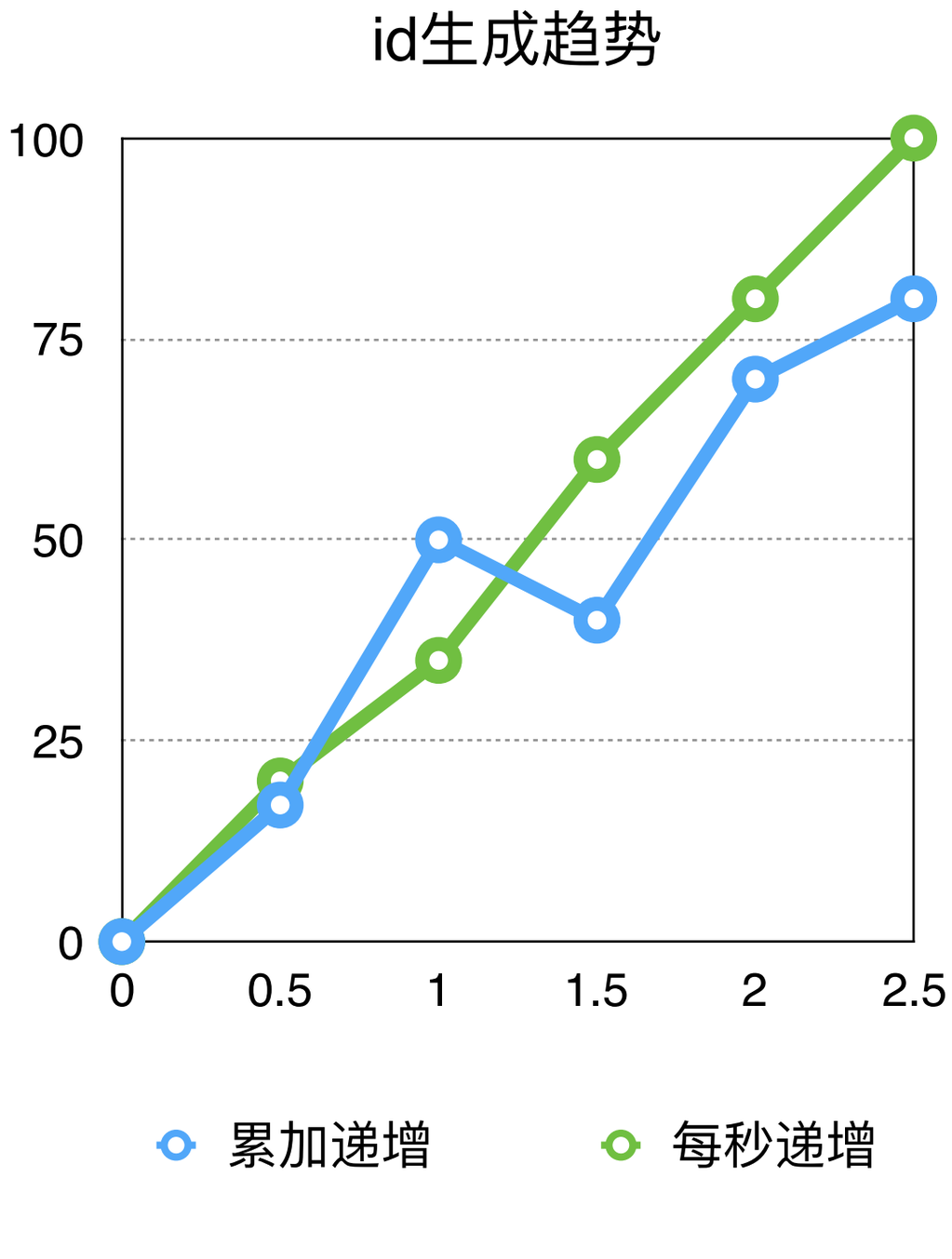
snowflake算法—twitter

- 0XFFFF FFFF FFFF FFFF， uint64的最大值
- 組成：時間戳-機器位-隨機數
- 41位 + 10位 + 12位
- 9223 3720 3257 7650 688
- 0111 1111 1111 1111 1111 1111 1111 1111 1111 1111
- 0000 0001 0001 0000 0100 0000 0000 0000
- 2199023251472-264-0

數據路由實現-id算法

- 1844 6744 0737 0955 1656， uint64的最大值
- 4294967295-0000-01-1-00
- 算法：時間戳-隨機數位-類型位-機器位-庫位
- 具體位數：10位 + 4位 + 2位 + 1位 + 2位
- 為什麼比最大值少了一位？

數據路由實現-id特性



- 累加遞增：長時間（2s內）內保證單調遞增，短時間（1s內）內不保證單調遞增
- 每秒遞增：單機肯定遞增，每秒都會從0開始計算

數據路由實現-應用邏輯

按照時間分

類型

4294967295-0000-01-1-00

分表位

分庫位

數據路由策略

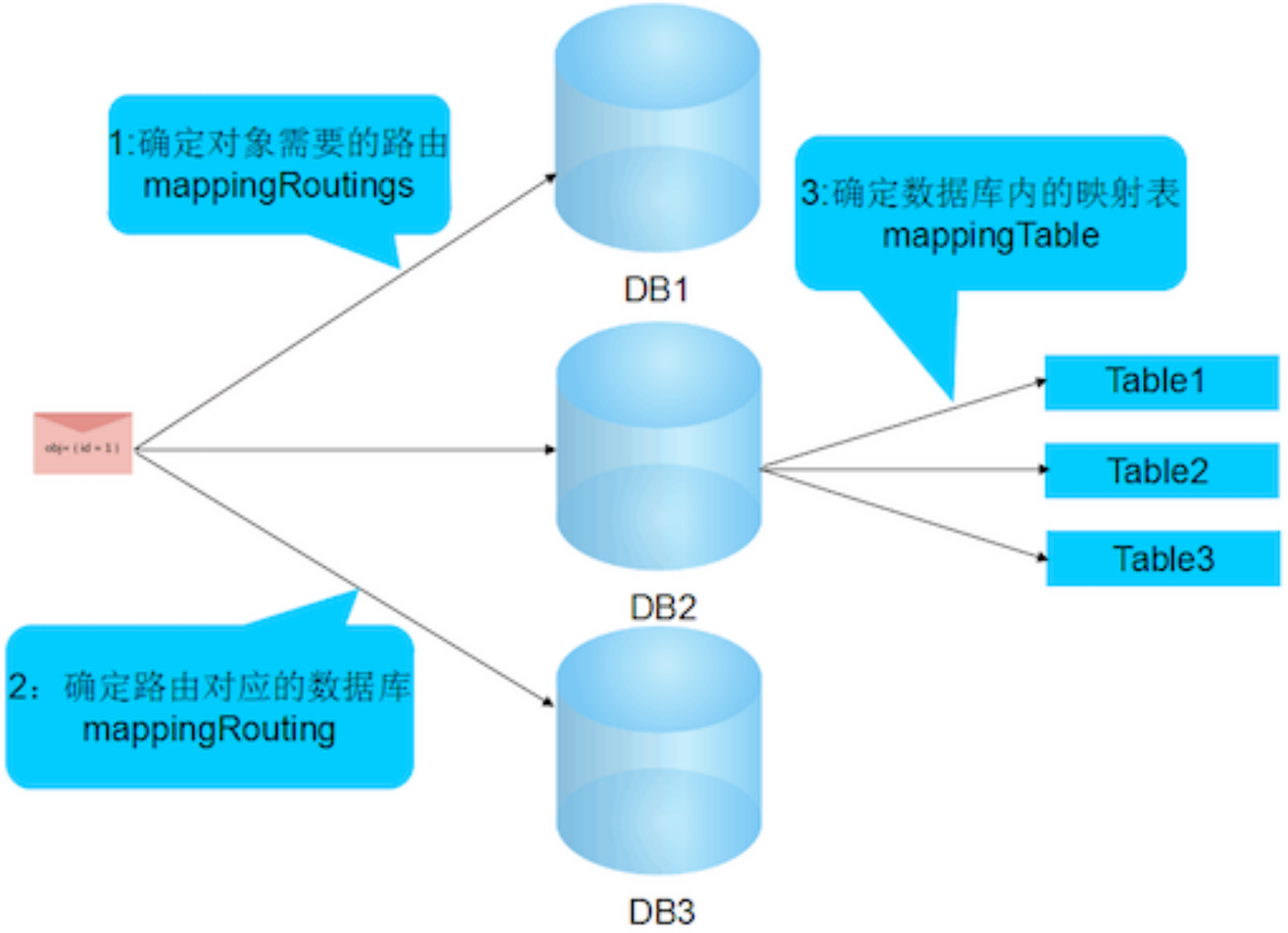
數據聚合問題，比如SQL的JOIN

- Slor等搜索引擎解決

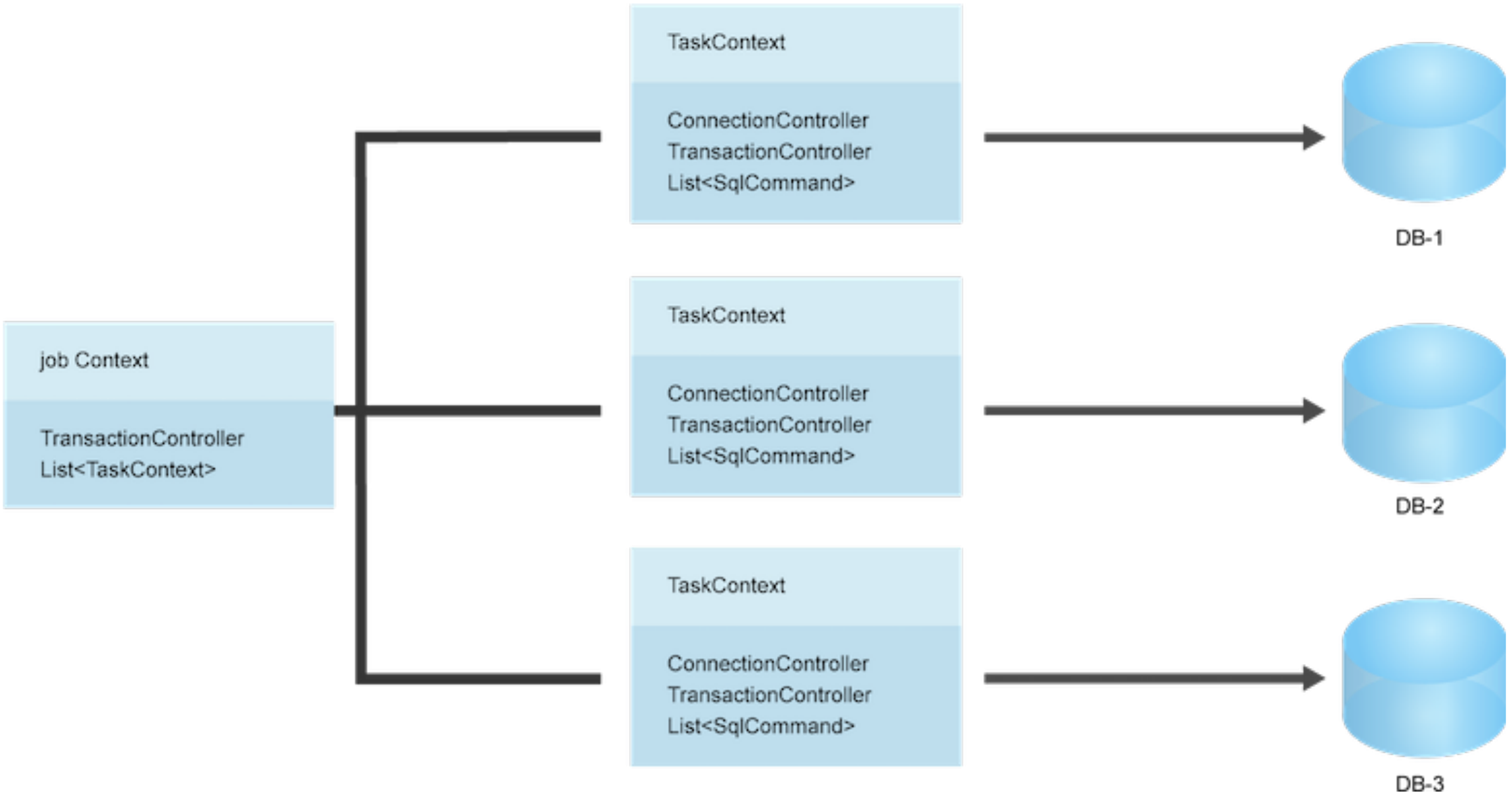
數據越大，數據的遷移問題

- 根據業務，設計一種不需要遷移數據的
- 實在沒辦法，使用“二叉樹”遷移數據方法

回到數據路由實現

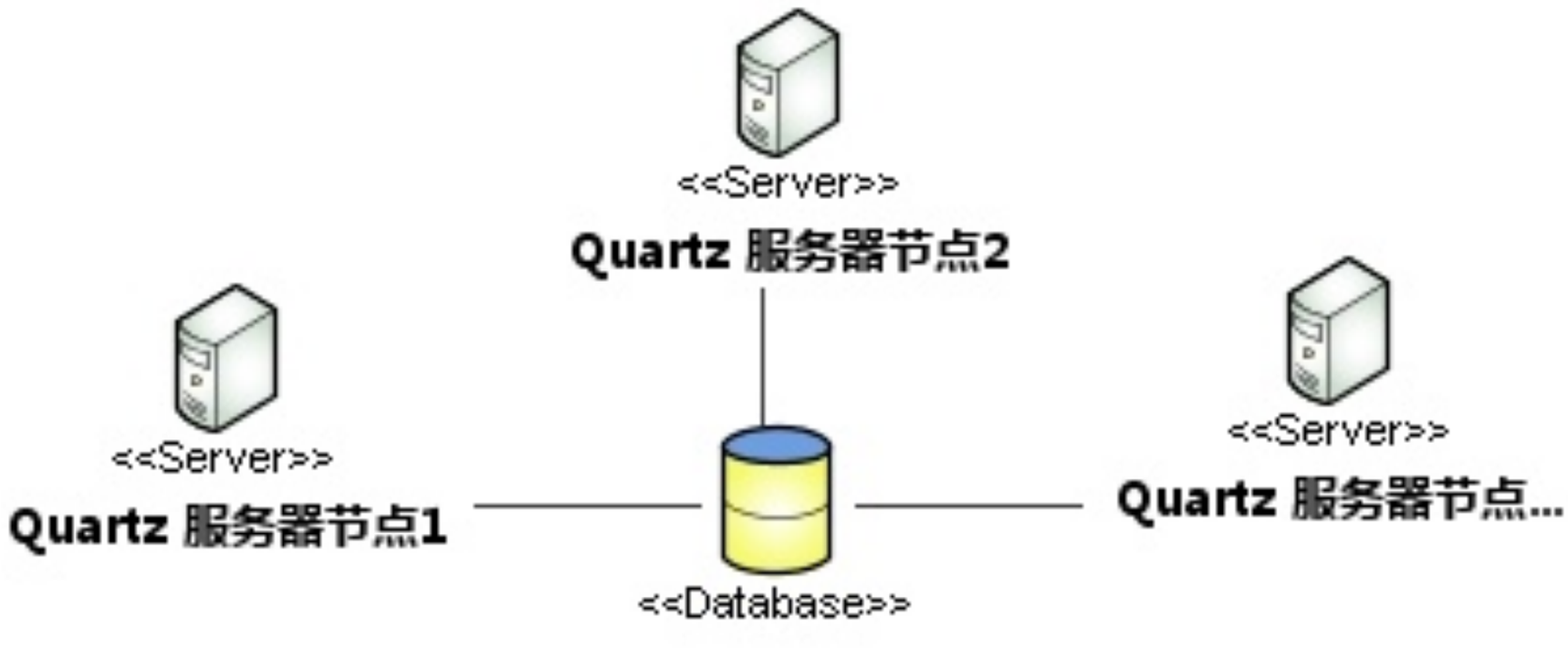


數據路由實現-DTC模型



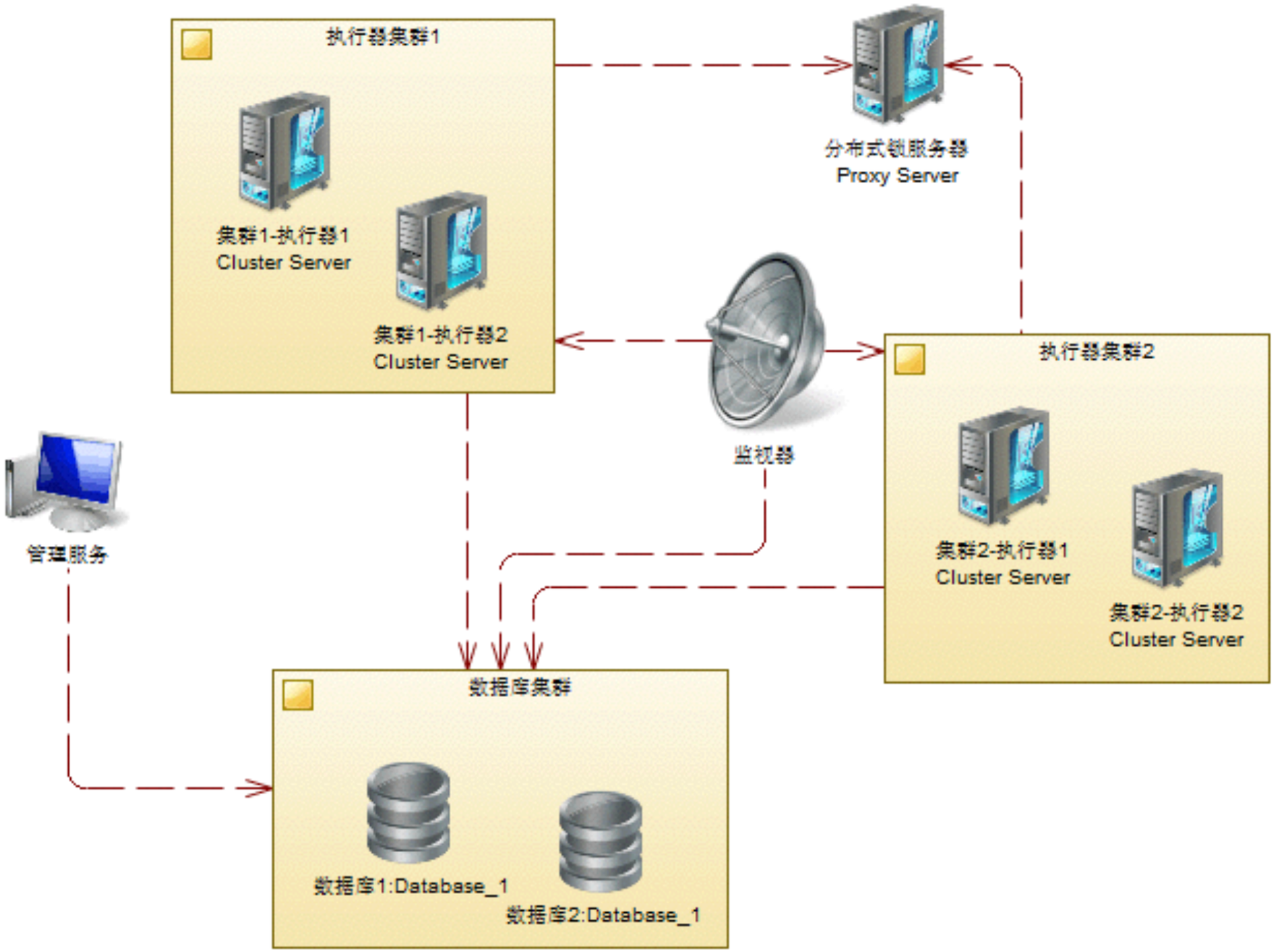
任務調度

任務調度V1-quartz實現



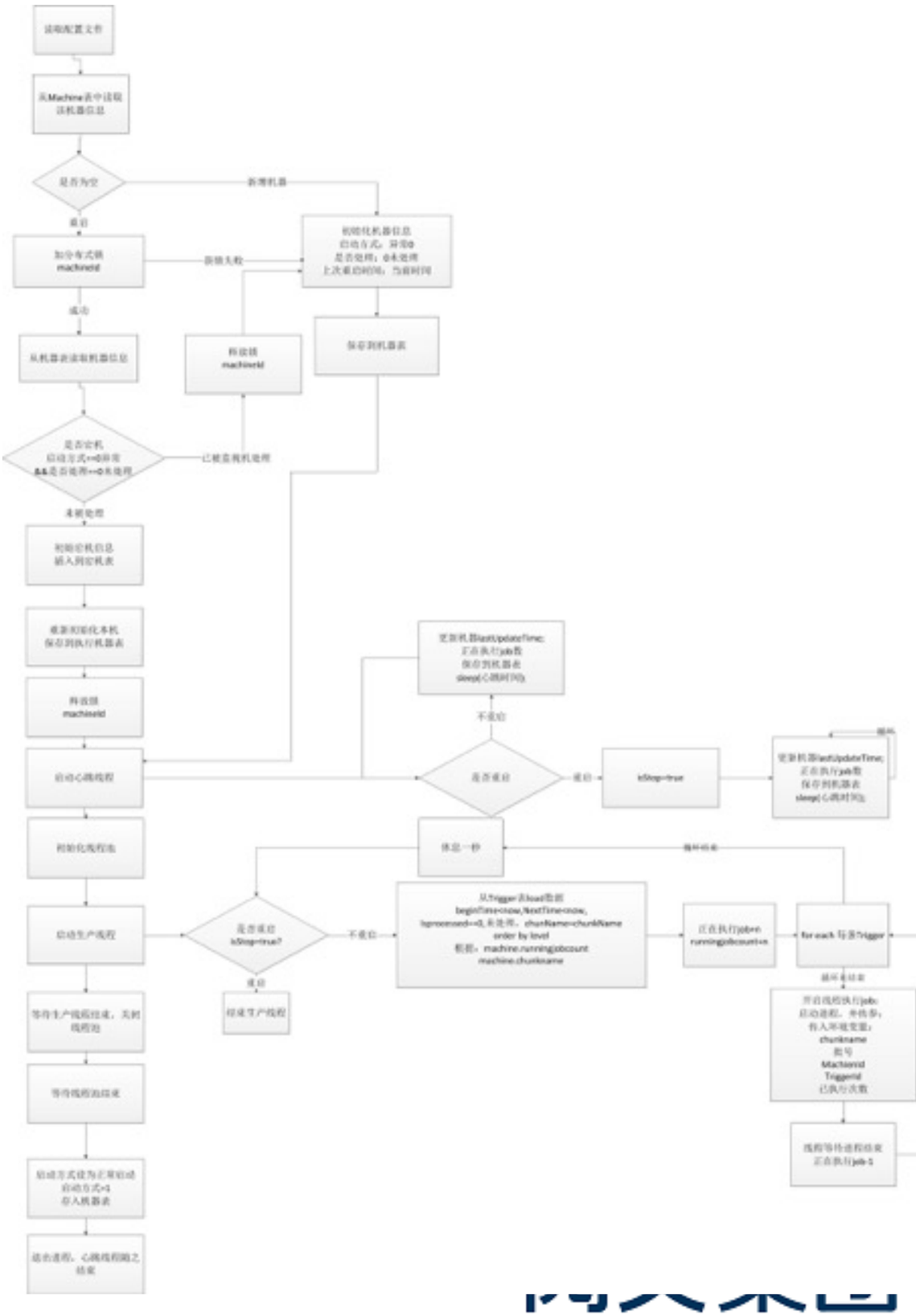
- 由多線程模型改成多進程模型
- 數據庫依賴太嚴重，特別是不可並發job
- 二次開發難度太大，代碼一坨一坨的

任務調度V2-自主實現



任務調度器-scher

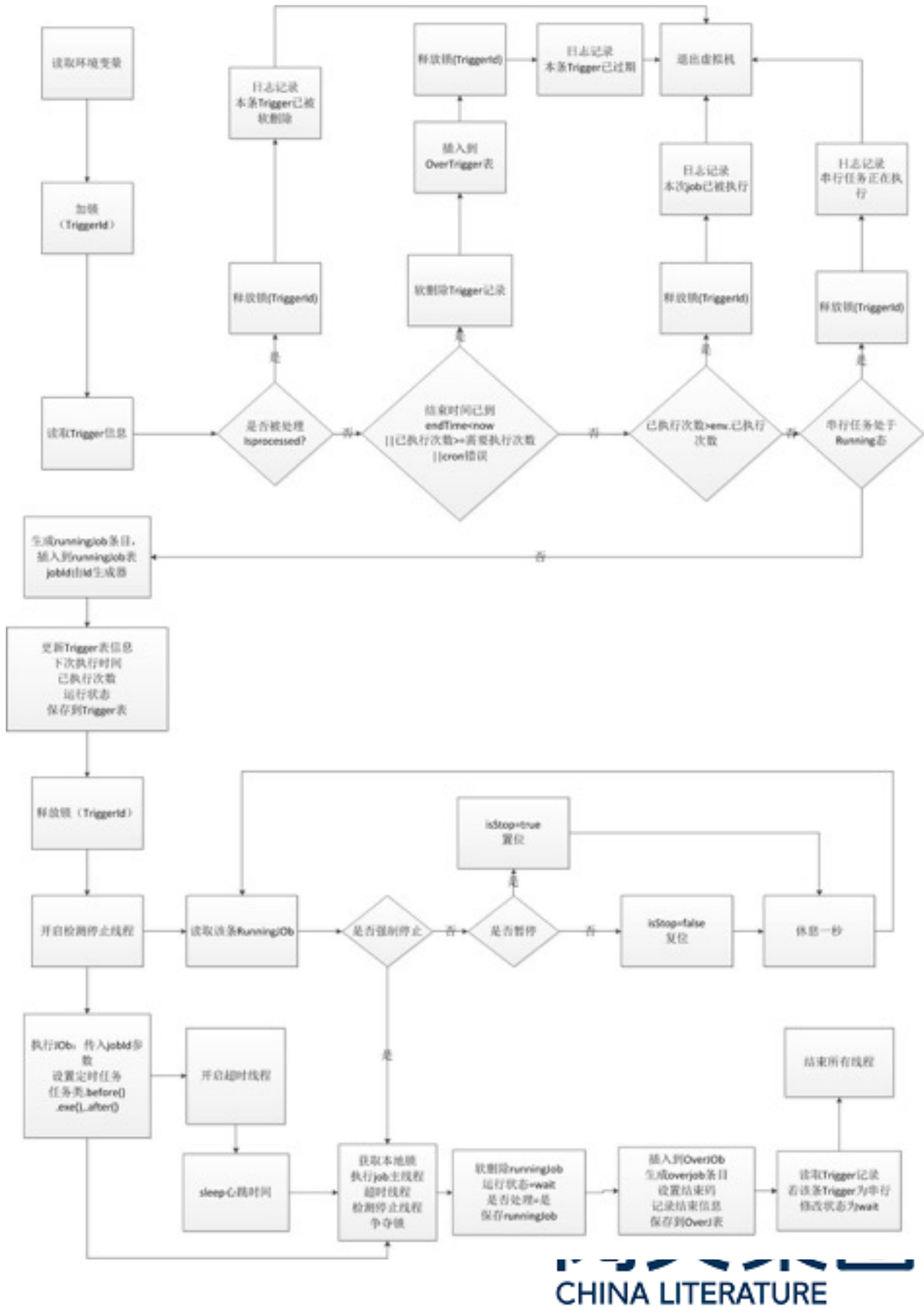
- batchid，每次生命週期的批號，被用來判斷是否已經重啟
- scher db，獲取需要觸發的job
- 多進程模型
- 執行container



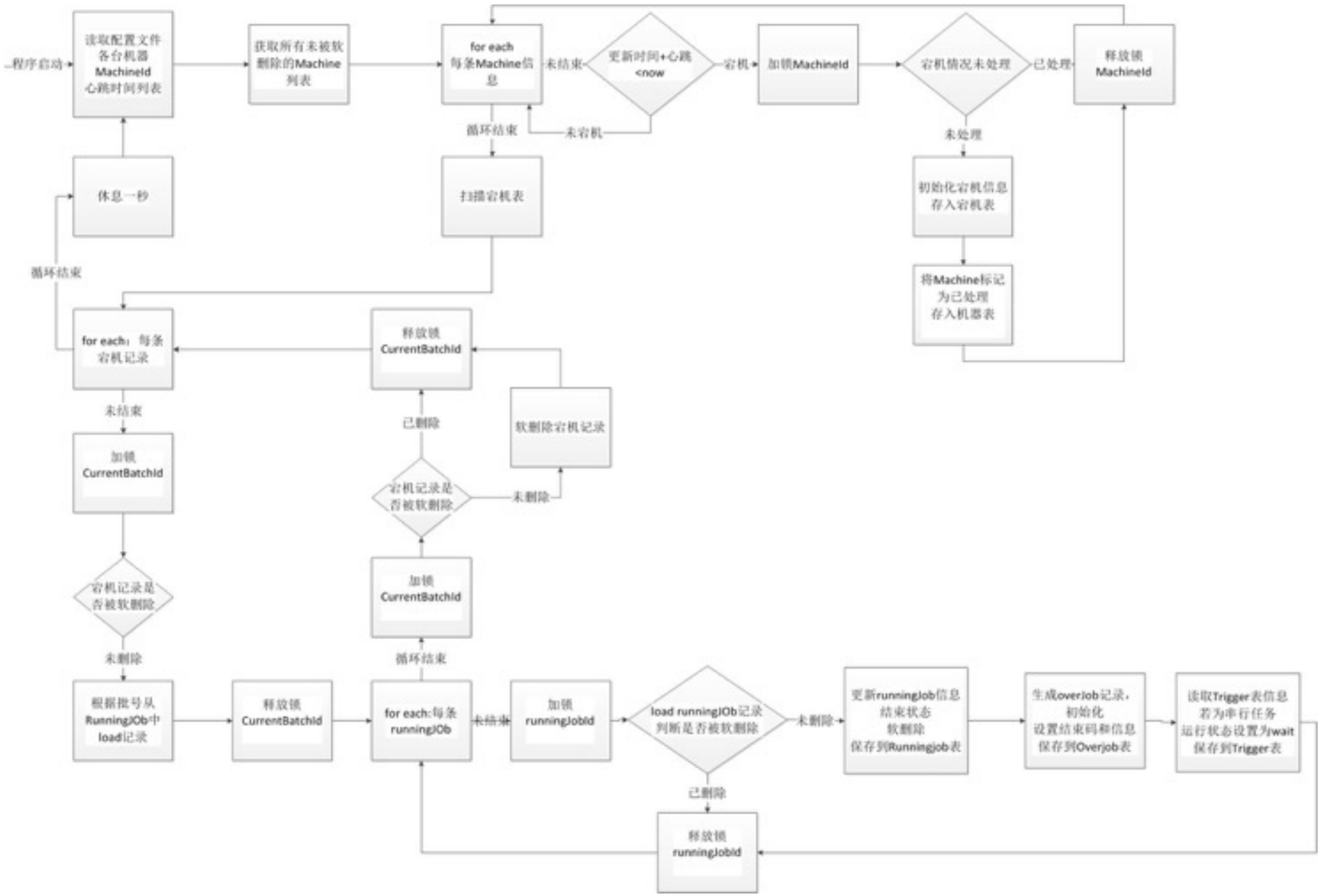
CHINA LITERATURE

任務調度器-container

- 每個job的包裝
- job分為可並行和不可並行
- job具有超時機制，超時后kill-self



任務調度器-arbiter



- 獲取被kill的scher，進行清理
- 清理發生異常的job

阅文集团
CHINA LITERATURE

任務調度器的使用問題

1. 調度時間間隔太短，比如1s一發
   - 限定時間間隔大小

2. 可並行job並沒有設置並發數量
   - 增加並發數量限制

3. 所有的job等級全部設置成10（10表示最優先級）
   - 這個只能通過管理限制

Q&A

微信公眾號

blog

阅文集团
CHINA LITERATURE