〈PPT第一頁 P1〉

**Leo**

**The early years-LEO**

* LinkedIn 的起步與今天許多網站的起步一樣，作為一個單一的整體應用程序來完成所有事情。  
  LinkedIn started as many sites start today, as a single monolithic application doing it all.
* 它為所有不同的頁面託管 web servlet，處理業務邏輯，並連接到少數 LinkedIn 數據庫。  
  It hosted web servlets for all the various pages, handled business logic, and connected to a handful of LinkedIn databases.
* 隨著網站的發展，Leo 也在發展，其角色和責任也在增加，自然而然地也增加了它的複雜性。負載平衡有助於啟動多個 Leo 實例。但增加的負載正在加重 LinkedIn 最關鍵的系統——其會員資料數據庫的負擔。  
  As the site grew, so did Leo, increasing its role and responsibility, and naturally increasing its complexity. Load balancing helped as multiple instances of Leo were spun up. But the added load was taxing LinkedIn’s most critical system - its member profile database.
* 我們做的一個簡單的修復是經典的垂直縮放——投入更多的 CPU 和內存！  
  An easy fix we did was classic vertical scaling - throw more CPUs and memory at it!

However, as the site began to see more and more traffic

* 隨著網站流量越來越大，我們的單體應用 Leo 經常在生產環境中宕機，很難進行故障排除和恢復，也很難發布新代碼。高可用性對 LinkedIn 至關重要。很明顯，我們需要“殺死 Leo”並將其分解為許多小型功能。  
  As the site began to see more and more traffic, our single monolithic app Leo was often going down in production, it was difficult to troubleshoot and recover, and difficult to release new code. High availability is critical to LinkedIn. It was clear we needed to “Kill Leo” and break it up into many small functional.

〈PPT第二頁 P2〉

Kafka

*Example*

* 需要將成批數據發送到我們的Hadoop 工作流中進行分析  
  we needed to send batches of data into our Hadoop workflow for analytics
* 收集並聚合來自每個服務器的日誌  
  we collected and aggregated logs from every service
* 收集頁面瀏覽量等跟踪事件  
  we collected tracking events like pageviews
* 當有人更新個人資料時，讓搜索系統保持最新。  
   we needed to keep our people search system up to date whenever someone updated their profile.
* 為了收集越來越多的數據  
  To collect its growing amount of data
* 解決 data pipeline 問題。不同於其他的訊息處理系統，Apache Kafka 是針對 big data streaming 處理而設計，可以輕易達到每秒上萬等級 requests
* 當使用 Kafka 後，可以將低其元件之間的耦合，且讓資料流變的清晰可見。
* 結果是開發了我們的分佈式發布-訂閱消息傳遞平台Kafka 。  
  The result was the development of Kafka, our distributed pub-sub messaging platform.

〈PPT第三頁 P3〉

Kafka 成為一個通用管道

* 圍繞提交日誌的概念構建，並且在構建時考慮了速度和可擴展性  
  Kafka became a universal pipeline, built around the concept of a commit log, and was built with speed and scalability in mind.
* 它支持近乎實時地訪問任何數據源  
  It enabled near real-time access to any data source.
* 支持 Hadoop 作業  
  empowered our Hadoop jobs
* 允許構建實時分析  
  allowed us to build real-time analytics
* 極大地改進我們的站點監控和警報能力  
  vastly improved our site monitoring and alerting capability

→如今，Kafka每天處理超過500 億個事件。（Today, Kafka handles well over 500 billion events per day.）

〈PPT第四頁 P4〉

The LinkedIn engineering team has developed and built Apache Kafka into a powerful open source solution for managing streams of information.   
LinkedIn 工程團隊已經開發並將Apache Kafka構建為一個強大的開源解決方案，用於管理信息流。

Today, some of the common scenarios at LinkedIn that leverage Kafka include:

如今，LinkedIn 使用 Kafka 的一些常見場景包括：

* Monitoring監控

All hosts at LinkedIn emit metrics pertaining to their system and application health through Kafka. These are then collected and processed to create monitoring dashboards and alerts.  
LinkedIn 的所有主機都通過 Kafka 發出與其係統和應用程序健康相關的指標。然後收集和處理這些信息以創建監控儀表板和警報。

* Messaging消息傳遞

Various applications at LinkedIn leverage Kafka as a traditional messaging system for standard queuing and pub-sub messaging.   
LinkedIn 的各種應用程序利用 Kafka 作為標準隊列和發布-訂閱消息傳遞的傳統消息傳遞系統。  
These applications range from Search, Content Feed and Relevance and they publish processed data into online data serving stores like Voldemort, etc.  
這些應用程序包括搜索、內容提要和相關性，~~它們將處理後的數據發佈到 Voldemort 等在線數據服務商店。~~

* Analytics分析

LinkedIn tracks data to better understand how our members use our products. Information such as which page got viewed and which content got clicked on are sent into a Kafka cluster in each data center. These events are all centrally collected and pushed onto our Hadoop grid for analysis and daily report generation.

LinkedIn 跟踪數據以更好地了解我們的會員如何使用我們的產品。諸如查看了哪個頁面和點擊了哪些內容等信息被發送到每個數據中心的 Kafka 集群中。這些事件全部集中收集並推送到我們的 Hadoop 網格上以進行分析和每日報告生成。

* As a building block (log) in various distributed applications/platforms  
  作為各種分佈式應用程序/平台中的構建塊（日誌）

Kafka is also leveraged as a core building block (distributed log) by other products like our big data warehousing solution Pinot. We are also working on using Kafka as an internal replication and change propagation layer for our distributed database Espresso.

Kafka 還被其他產品（如我們的大數據倉庫解決方案Pinot）用作核心構建塊（分佈式日誌）。我們還致力於將 Kafka 用作分佈式數據庫 Espresso 的內部複製和更改傳播層。

https://www.infoq.cn/article/2015/02/kafka-linkedin-application