Big Data Programming using Map/Reduce-HDFS in Hadoop

楊順發

shunfa@nchc.narl.org.tw

自由軟體實驗室

國家高速網路與計算中心

TAIWAN www.nchc.org.tw National Applied Research Laboratories NCHC

Virtual Box image download

- http://hadoop.nchc.org.tw/~h2833/

- 帳號：hadooper
- 密碼：hadoop

課程大綱 – (上午)

10m	教學環境確認
09:00~09:20	**Thinking in Big Data**
09:20~10:10	**Hadoop簡介，實作：Hadoop 單機安裝與基本操作**
10:10~10:20	*休息*
10:20~11:10	**Hadoop Overview** **Hadoop Distributed File System 簡介** **實作： HDFS 指令操作練習**
11:10~11:20	*休息*
11:20~12:00	**Map Reduce 介紹** **實作： 執行 MapReduce 基本運算**
補充	**設定參數解析**

課程大綱-(下午)

13:30~15:00	**MapReduce 程式設計** ***實作：Hadoop 程式編譯與執行***
15:00~15:10	*休息*
15:10~15:20	**用Eclipse開發 Hadoop**
15:20~16:00	**Hadoop應用 - Crawlzilla搜尋引擎安裝與實作**
16:00~16:30	**Hadoop 叢集安裝設定解析** ***實作：Hadoop 叢集環境操作***

Two Questions

- Linux?
- Java?

Thinking in Big Data

楊順發

shunfa@nchc.narl.org.tw

自由軟體實驗室

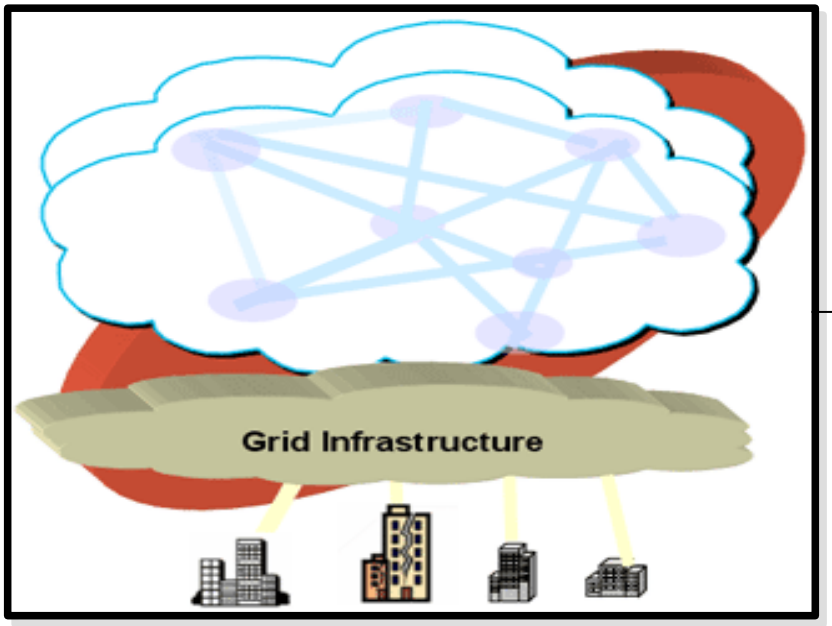
國家高速網路與計算中心

TAIWAN www.nchc.org.tw National Applied Research Laboratories NCHC

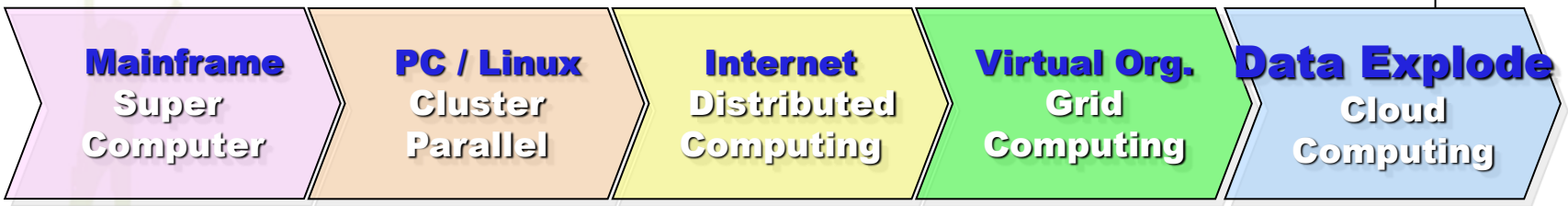
Outline

- **Big Data?**
- 資料運算演進
- 資料分析
  - 傳統
  - 分散式計算

資料運算演進



Source: http://mmdays.com/2008/02/14/cloud-computing/



雲端運算vs.網格運算		
	雲端運算	網格運算
主要推動者	資訊供應商（如Google、Yahoo、IBM、Amazon等）	學術機構（如歐洲粒子研究中心CERN、中研院、國家高速網路與計算中心）
標準化程度	無標準化，各家採用的技術架構也不同。	有標準化的協定和信任機制
開源幅度	部分開源，目前有開源Hadoop框架，但Google GFS和資料庫系統BigTable則未開源。	完全開源
網域限制	企業內部網域	可跨企業、跨管理網域
單一運算叢集可支援的硬體	相同標準規格的個人電腦（如x86處理器、硬碟、4GB記憶體、Linux等）	可混合異質性伺服器（不同處理器、不同作業系統、不同編譯器版本等）
擅長處理的資料特性	單次運算資料量小（可於單臺個人電腦上執行），但需要重複大量處理次數的應用。	單次運算資料量大的應用。例如單筆數GB的衛星訊號分析。
資料來源：iThome整理，2008年6月		

Big Data Age

Inside Ancestry.com's Top-Secret Data Center
Posted by Diane

Inside the unassuming building that is the data center for Ancestry.com and other Generations Network properties, rows and rows of cabinets house the 5,328 servers that hold the Web site, all those indexes and digital images, and users' family trees.

In all, it's 2.5 petabytes of data (one petabyte is equivalent to 283,000 DVDs).

A lot of security protects that data. A guard watches cameras 24/7. Windows are bulletproof. Sensors monitor windows and doors. The Ancestry.com guy walking us around had to swipe his badge at several doors, then lay his palm in a *Mission: Impossible*-like handprint reader to enter the server rooms.

I can't disclose the location and photographs weren't permitted (darn it, I forgot my hidden-camera lapel pin), but the folks at Ancestry.com sent these approved images:

Some rows of server-filled cabinets:

- Ancestry.com, the genealogy site, stores around 2.5 petabytes of data.
  - http://www.ancestry.com/

Big Data Age

New York Stock Exchange Ticks on Data Warehouse Appliances

Netezza deployment replaces mega data warehouses while cutting query times from hours to seconds.

By **Doug Henschen** InformationWeek
May 16, 2008 04:59 PM

It's not a typical enterprise data warehouses story, but then NYSE Euronext (NYSE), the parent company of the New York Stock Exchange, is not a typical enterprise. For one thing, NYSE has not one but three warehouses, each approaching 100 terabytes. Then consider NYSE's queries, some of which interrogate more than 40 terabytes of data. The extreme data volumes and extreme query complexity led to an upgrade onto data warehouse appliances.

After a period of rampant growth and mergers with two smaller exchanges, NYSE knew its large and aging Oracle data warehouses needed replacement. After exploring alternatives in 2006, the company concluded a successful 45-day proof-of-concept project on a Netezza Performance Server (NPS) appliance in early 2007. The main warehouse for the New York Stock Exchange was migrated within two and a half months and went into production in May 2007. A second device, consolidating what had been two separate warehouses for the Chicago-based Arca Equities and Options markets, went into production in July. Yet another warehouse, one housing legacy data, will be migrated onto a third Netezza NPS.

Big Data Age

Facebook Trumps Most Photo Sharing Sites With 10 Billion Photos

October 15, 2008 by **Stan Schroeder**

74

Sometimes I forget how big the Internet is, and then something reminds me just how flabbergastingly, enormously huge the damn thing is. This time it's Facebook, whose software engineer Doug Beaver announced that it now hosts a total of 10 billion photos. And it's not even exclusively a photo sharing site!

Furthermore, since Facebook stores four sizes for each stored photo, this actually translates to 40 billion files. However, what I find fascinating about this is that Facebook is no Photobucket; it's not just some repository (at least from what I've seen) where people dump all kinds of images just because they can. Vast majority of photos I see on Facebook are actually real life photographs taken by users; I'm not sure if this means much in the grand scheme of life, universe and everything, but it seems like quite an accomplishment.

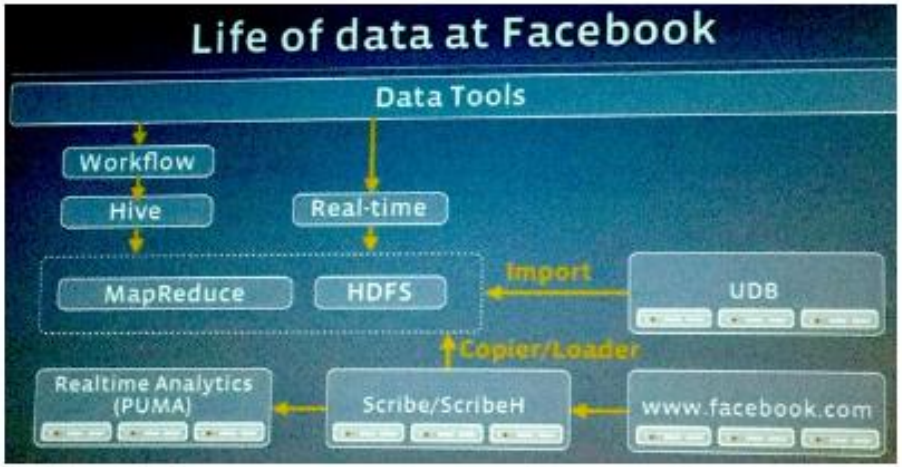
Big Data Age

FB大數字：每天處理3億張照片、25億則發文、27億個「讚」

數位時代網站｜撰文者：陳荻雅編譯　發表日期 2012-08-25

讚 23 人說這讚。成為你朋友中第一個說讚的人。　+1 5

Facebook對媒體揭露了身為社群網站龍頭的網站資料量實際數據－包括每天系統需處理超過25億則的發文、每天乘載超過500TB的流量、每天會有27億個「讚」、每日總上傳照片數約3億張、每半小時就要掃描約105TB的資料。



What Is Big Data?

From Databases to Big Data

Sam Madden • *Massachusetts Institute of Technology*

- Among all the definitions offered for “big data,” my favorite is that it means data that’s too big, too fast, or too hard for existing tools to process.

資料存儲

- 容量：1,370MB, 傳輸速度：4.4M/s
  - 1990年代，傳輸完一顆硬碟的速度約5分鐘
- 100M/s的傳輸速度
  - 2010年代
  - 每天在社群網站約產生500T的資料量
  - 複習一下：K -> M -> G -> T
  - 傳輸完一天所產生的資料需要約**1388888.89**小時，也就是**57870.37**天，也就是**158.55**年...
  - 這！還不包括資料分析的時間！

資料分析 - Traditional

- 以美國氣象資料為例：

0057
332130 # USAF weather station identifier
99999 # WBAN weather station identifier
19500101 # observation date
0300 # observation time
4
+51317 # latitude (degrees x 1000)
+028783 # longitude (degrees x 1000)
FM-12
+0171 # elevation (meters)
99999
V020
320 # wind direction (degrees)
1 # quality code
N
0072
1
00450 # sky ceiling height (meters)
1 # quality code
CN
010000 # visibility distance (meters)
1 # quality code
N9
-0128 # air temperature (degrees Celsius x 10)
1 # quality code
-0139 # dew point temperature (degrees Celsius x 10)
1 # quality code
10268 # atmospheric pressure (hectopascals x 10)
1 # quality code

資料分析 – Traditional

- 利用awk進行分析 1901~2001的所有資料

```
#!/usr/bin/env bash
for year in all/*
do
echo -ne `basename $year .gz`"\t"
gunzip -c $year | \
awk '{ temp = substr($0, 88, 5) + 0;
q = substr($0, 93, 1);
if (temp !=9999 && q ~ /[01459]/ && temp > max) max = temp }
END { print max }'
done
```

- 執行結果

```
1901 317
1902 244
1903 289
1904 256
1905 283
```

- ***費時：42min on EC2 High-CPU Extra Large Instance***

Hadoop 簡介

楊順發

shunfa@nchc.narl.org.tw

自由軟體實驗室

國家高速網路與計算中心

Outline

- Hadoop?
- Why?
- Hadoop History
- Hadoop Ecosystem
- Hadoop Releases
- 實作：Hadoop安裝

Hadoop

- Hadoop是Apache軟體基金會所研發的開放源碼並行運算編程工具和分散式檔案系統，與MapReduce和Google檔案系統的概念類似。

About

- 以Java開發
- 自由軟體
- 上千個節點
- Petabyte等級的資料量
- 創始者 Doug Cutting
- 為Apache 軟體基金會的 top level project

Why Hadoop?

- **Need to process Multi Petabyte Datasets**
- **Data may not have strict schema**
- **Expensive to build reliability in each application.**
- **Nodes fail every day**
  - Failure is expected, rather than exceptional.
  - The number of nodes in a cluster is not constant.
- **Need common infrastructure**
  - Efficient, reliable, Open Source Apache License

特色

- 巨量
  - 擁有儲存與處理大量資料的能力
- 經濟
  - 可以用在由一般PC所架設的叢集環境內
- 效率
  - 籍由平行分散檔案的處理以致得到快速的回應
- 可靠
  - 當某節點發生錯誤，系統能即時自動的取得備份資料以及佈署運算資源

起源

- **Nutch**
  - nutch是基於開放原始碼所開發的web search
  - 利用Lucene函式庫開發
- **Lucene**
  - 用Java設計的高效能文件索引引擎API
  - 索引文件中的每一字，讓搜尋的效率比傳統逐字比較還要高的多

起源：Google論文

- **Google File System**
  - SOSP 2003 : “The Google File System”
  - OSDI 2004 : “MapReduce : Simplifed Data Processing on Large Cluster”
  - OSDI 2006 : “Bigtable: A Distributed Storage System for Structured Data”
  - 可擴充的分散式檔案系統
  - 大量的用戶提供總體性能較高的服務
  - 對大量資訊進行存取的應用
  - 運作在一般的普通主機上
  - 提供錯誤容忍的能力

起源:2004~

- Dong Cutting 開始參考論文來實做
- Added DFS & MapReduce implement to Nutch
- Nutch 0.8版之後，Hadoop為獨立項目
- Yahoo 於2006年僱用Dong Cutting 組隊專職開發
  - Team member = 14 （engineers, clusters, users, etc.)
- 2009 年跳槽到Cloudera

History

- Dec 2004 – Google GFS paper published
- July 2005 – Nutch uses MapReduce
- Feb 2006 – Starts as a Lucene subproject
- Apr 2007 – Yahoo! on 1000-node cluster
- Jan 2008 – An Apache Top Level Project
- May 2009 – Hadoop sorts Petabyte in 17 hours
- Aug 2010 – World's Largest Hadoop cluster at Facebook
  - 2900 nodes, 30+ PetaByte

Who Use Hadoop?

- Amazon/A9
- Facebook
- Google
- IBM
- Joost
- Last.fm
- New York Times
- PowerSet
- Veoh
- Yahoo!

- **http://wiki.apache.org/hadoop/PoweredBy**

用途

- Search
  - *Yahoo, Amazon, Zvents*
  - *關於Yahoo!與Hadoop*
    - http://www.slideshare.net/ydn/hadoop-yahoo-internet-scale-data-processing
- Log processing
  - *Facebook, Yahoo, ContextWeb. Joost, Last.fm*
- Recommendation Systems
  - *Facebook*
- Data Warehouse
  - *Facebook, AOL*
- Video and Image Analysis
  - *New York Times, Eyealike*

Hadoop於yahoo的運作資訊

年份	日期	節點數	耗時（小時）
2006	四月	188	47.9
2006	五月	500	42
2006	十一月	20	1.8
2006	十一月	100	3.3
2006	十一月	500	5.2
2006	十一月	900	7.8
2007	七月	20	1.2
2007	七月	100	1.3
2007	七月	500	2
2007	七月	900	2.5

Hadoop於Yahoo的部屬情形

資料標題：Yahoo! Launches World's Largest Hadoop Production Application

資料日期：February 19, 2008

Number of links between pages in the index	roughly 1 trillion links
Size of output	over 300 TB, compressed!
Number of cores used to run single Map-Reduce job	over 10,000
Raw disk used in the production cluster	over 5 Petabytes

Hadoop於Yahoo的部屬情形

資料標題：Scaling Hadoop to 4000 nodes at Yahoo!
資料日期：September 30, 2008

Total Nodes	**4000**
Total cores	**30000**
Data	**16PB**

	500-node cluster		4000-node cluster	
	write	**read**	**write**	**read**
number of files	990	990	14,000	14,000
file size (MB)	320	320	360	360
total MB processes	316,800	316,800	5,040,000	5,040,000
tasks per node	2	2	4	4
avg. throughput (MB/s)	**5.8**	**18**	**40**	**66**

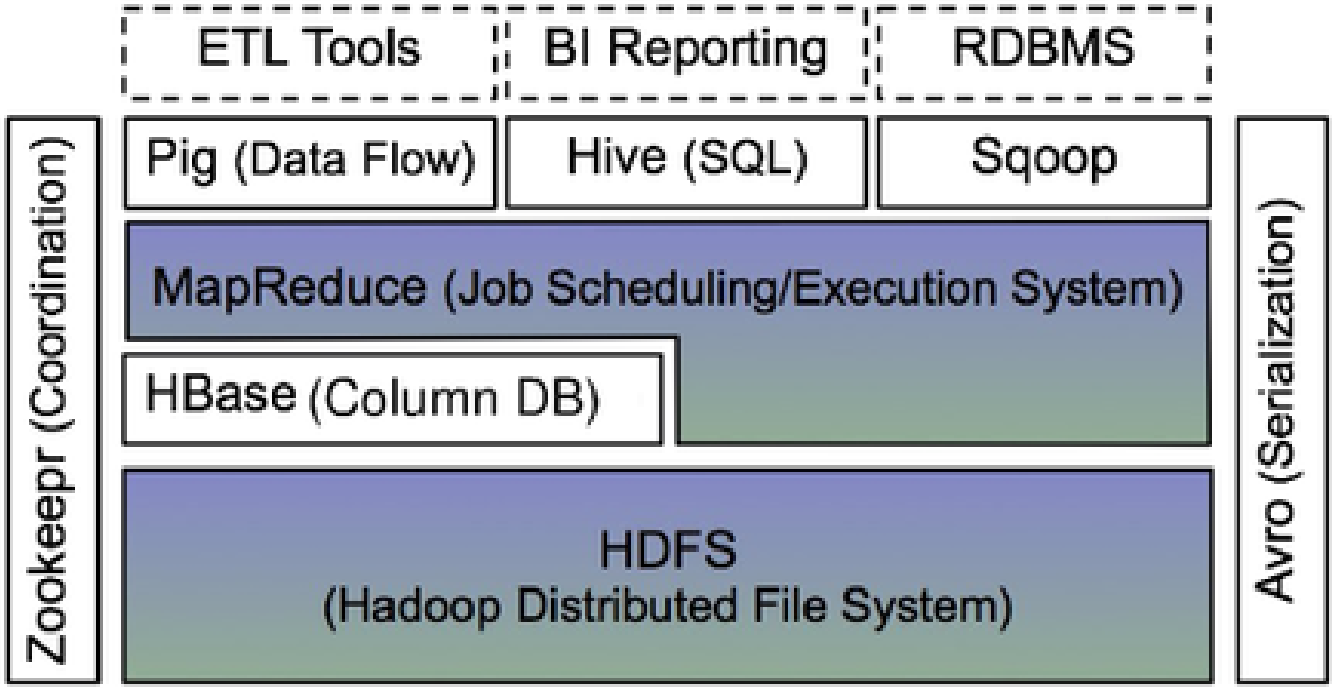
Hadoop 與Google的對應

Develop Group	Google	Apache
Sponsor	Google	Yahoo, Amazon
Algorithm Method	MapReduce	Hadoop
Resource	open document	open source
File System (MapReduce)	GFS	HDFS
Storage System (for structure data)	big-table	Hbase
Search Engine	Google	nutch
OS	Linux	Linux / GPL

Features supported by Hadoop release series

Feature	1.x	0.22	2.X
Secure authentication	Y	N	Y
Old configuration names	Y	Deprecated	Deprecated
New configuration names	N	Y	Y
Old MapReduce API	Y	Y	Y
New MapReduce API	Y	Y	Y
MapReduce 1 runtime (Classic)	Y	Y	N
MapReduce 2 runtime (YARN)	N	N	Y
HDFS federation	N	N	Y
HDFS high-availability	N	N	Y

The Hadoop Ecosystem



cloudera

Hadoop 0.20.2單機安裝

楊順發
shunfa@nchc.narl.org.tw
自由軟體實驗室
國家高速網路與計算中心

參考文件

- **http://trac.nchc.org.tw/cloud/wiki/Hadoop_Lab1**

Hadoop Overview

楊順發

shunfa@nchc.narl.org.tw

自由軟體實驗室

國家高速網路與計算中心

作業系統的最核心！

儲存空間的資源管理

記憶體空間與
行程分配

名詞

- **Job**
  - 任務
- **Task**
  - 小工作
- **JobTracker**
  - 任務分派者
- **TaskTracker**
  - 小工作的執行者
- **Client**
  - 發起任務的客戶端
- **Map**
  - 應對
- **Reduce**
  - 總和

- **Namenode**
  - 名稱節點
- **Datanode**
  - 資料節點
- **Namespace**
  - 名稱空間
- **Replication**
  - 副本
- **Blocks**
  - 檔案區塊 (64M)
- **Metadata**
  - 屬性資料

管理資料

Namenode

- Master
- 管理HDFS的名稱空間
- 控制對檔案的讀/寫
- 配置副本策略
- 對名稱空間作檢查及紀錄
- 只能有一個

Datanode

- Workers
- 執行讀/寫動作
- 執行Namenode的副本策略
- 可多個

分派程序

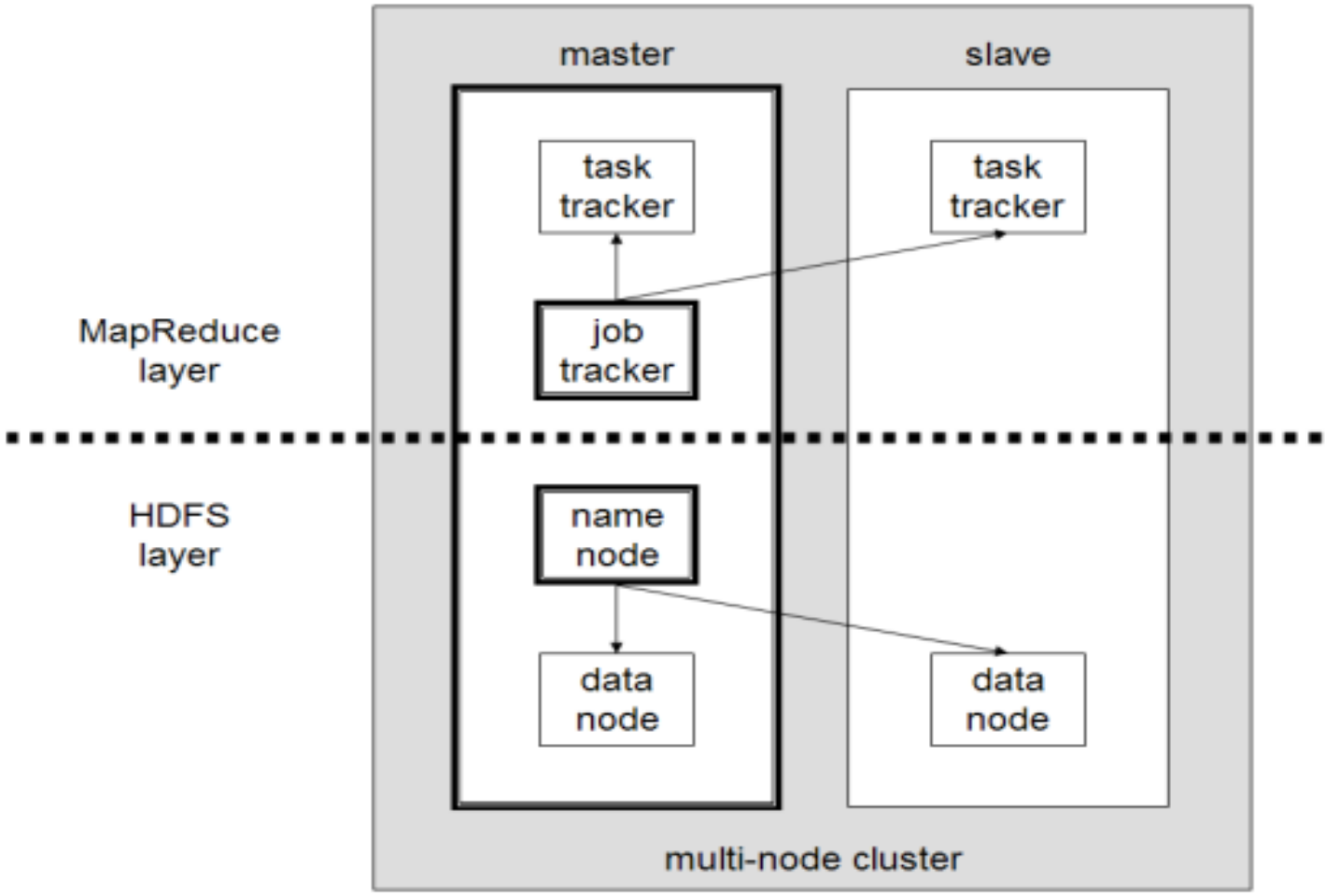
Jobtracker

- Master
- 使用者發起工作
- 指派工作給 Tasktrackers
- 排程決策、工作分配、錯誤處理
- 只能有一個

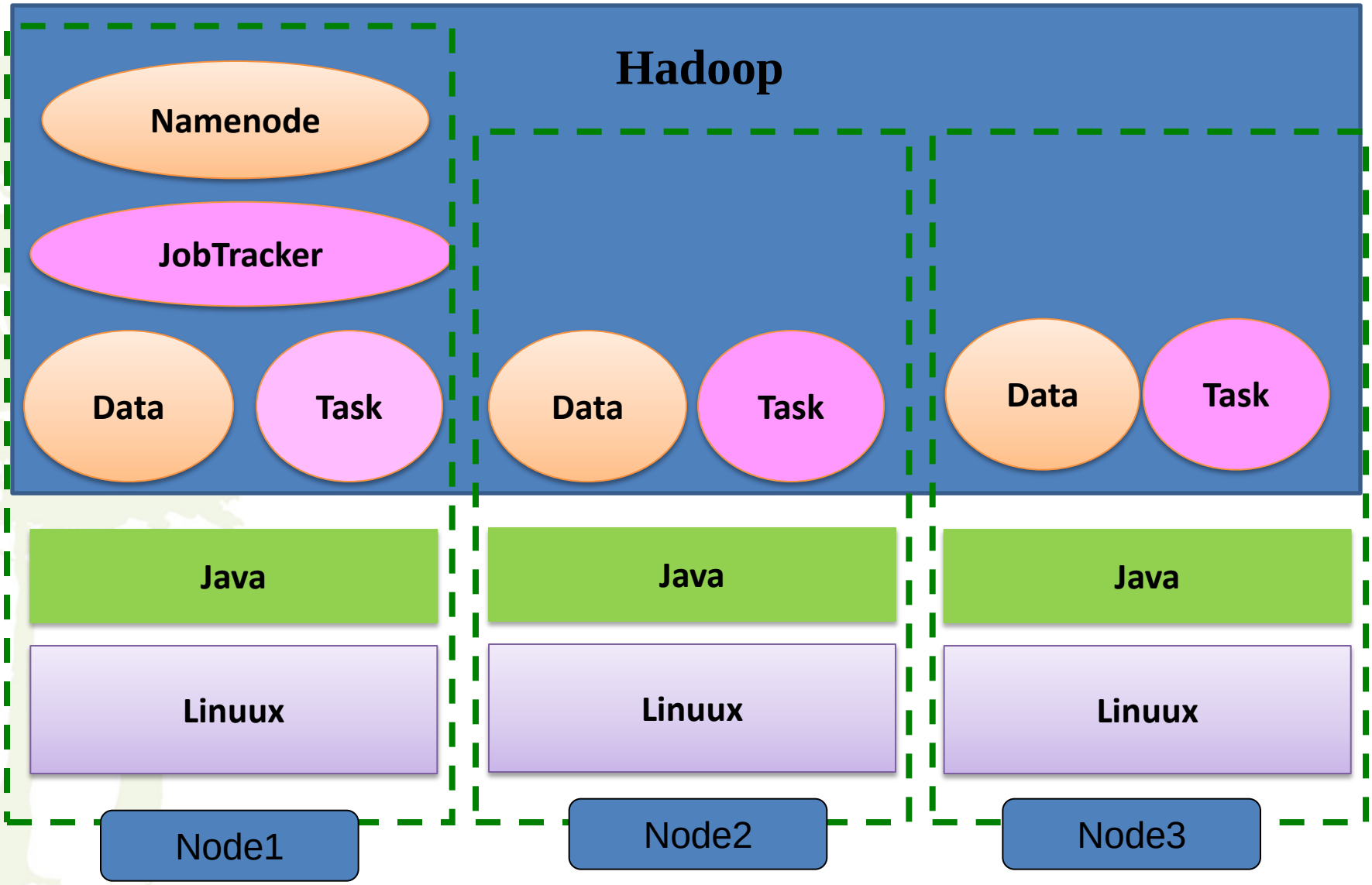
Tasktrackers

- Workers
- 運作Map 與 Reduce 的工作
- 管理儲存、回覆運算結果
- 可多個

Hadoop的各種身份



Building Hadoop



HDFS介紹

楊順發
shunfa@nchc.narl.org.tw
自由軟體實驗室
國家高速網路與計算中心

Outline

- HDFS?
- The Design of HDFS
- HDFS Concepts
- HDFS High-Availability
- HDFS Federation
- POSIX Like
- Data Flow

HDFS ?

- **Hadoop Distributed File System**
  - Hadoop：自由軟體專案，為實現Google的MapReduce架構
  - HDFS: Hadoop專案中的檔案系統
- **實現類似Google File System**
  - GFS是一個易於擴充的分散式檔案系統，目的為對大量資料進行分析
  - 運作於廉價的普通硬體上，又可以提供容錯功能
  - 給大量的用戶提供總體性能較高的服務

Goals of HDFS

- **Very Large Distributed File System**
  - 10K nodes, 1 billion files, 100 PB
- **Assumes Commodity Hardware**
  - Files are replicated to handle hardware failure
  - Detect failures and recovers from them

Goals of HDFS

- **Optimized for Batch Processing**
  - Data locations exposed so that computations can move to where data resides
  - Provides very high aggregate bandwidth
- **User Space, runs on heterogeneous OS**
- Streaming data access
  - Write Once, read many times.

不適合HDFS的情況

- **Low-latency data access**
- **Lots of small files**
- **Multiple writers, arbitrary file modifications**

HDFS Concepts

- Blocks
- Namenodes and Datanodes
- HDFS Federation(2.x version)

Blocks

- 檔案存儲的最小單位
- **In HDFS, Block Size is 64MB by default**
- **Why?**
  - to minimize the cost of seeks
- 查看檔案**block**分配情形
  - % hadoop fsck / -files -blocks

Block Placement

- **Policy**
  - 在本端機架的本端節點上放置一份複本
  - 在本端機架的不同節點上放置第二份複本
  - 在遠端機架上放置第三份複本
  - 他複本則隨機放置
- **用戶端會讀取位置最近的複本**

Namenodes and Datanodes

- **Namenode**
  - 只能有一個(Master)
  - 負責存儲檔案系統的metadata
  - In Local Disk
    - namespace image
    - edit log
- **Datanode**
  - 可多個
  - 檔案系統的workhorses

Heartbeats

- DataNode 傳送Heartbeats給 NameNode
  - 每 3 秒傳送一次
- NameNode 使用Heartbeats來偵測 DataNode問題。

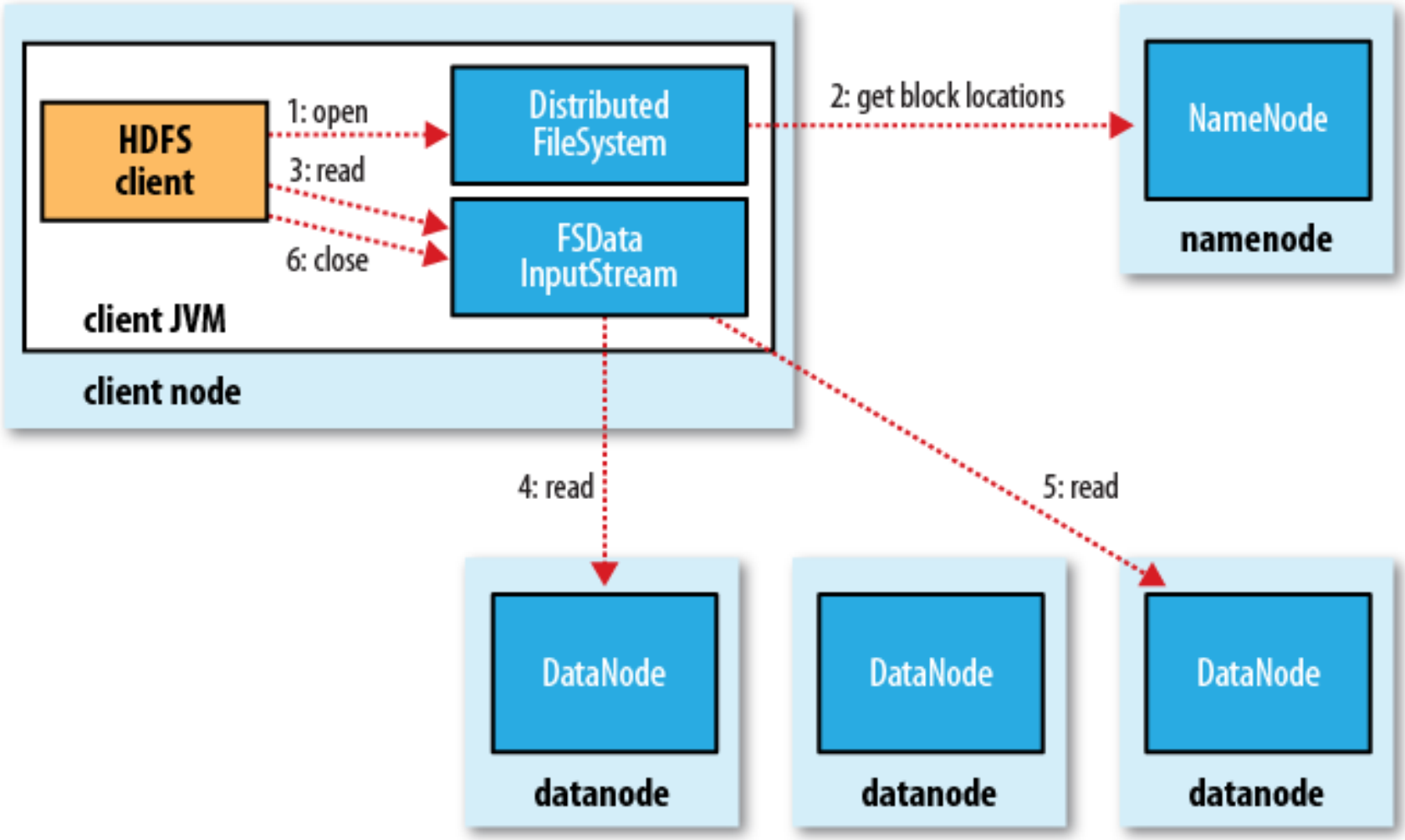
Others

- **Secondary Namenode**
  - Not act as a namenode
  - periodically merge the namespace image with the edit log to prevent the edit log from becoming too large
  - 須大量的運算資源，通常運作於另一台主機上

Data Correctness

- 使用Checksum來驗證資料
  - Cyclic Redundancy Check (CRC32)
- 檔案建立
  - 用戶端每隔 512 個位元組就會計算Checksum
  - DataNode 儲存Checksum的資訊
- 檔案存取時
  - 用戶端會同時擷取資料與Checksum
  - 如果驗證失敗，用戶端會嘗試使用其他複本

Data Flow - Read

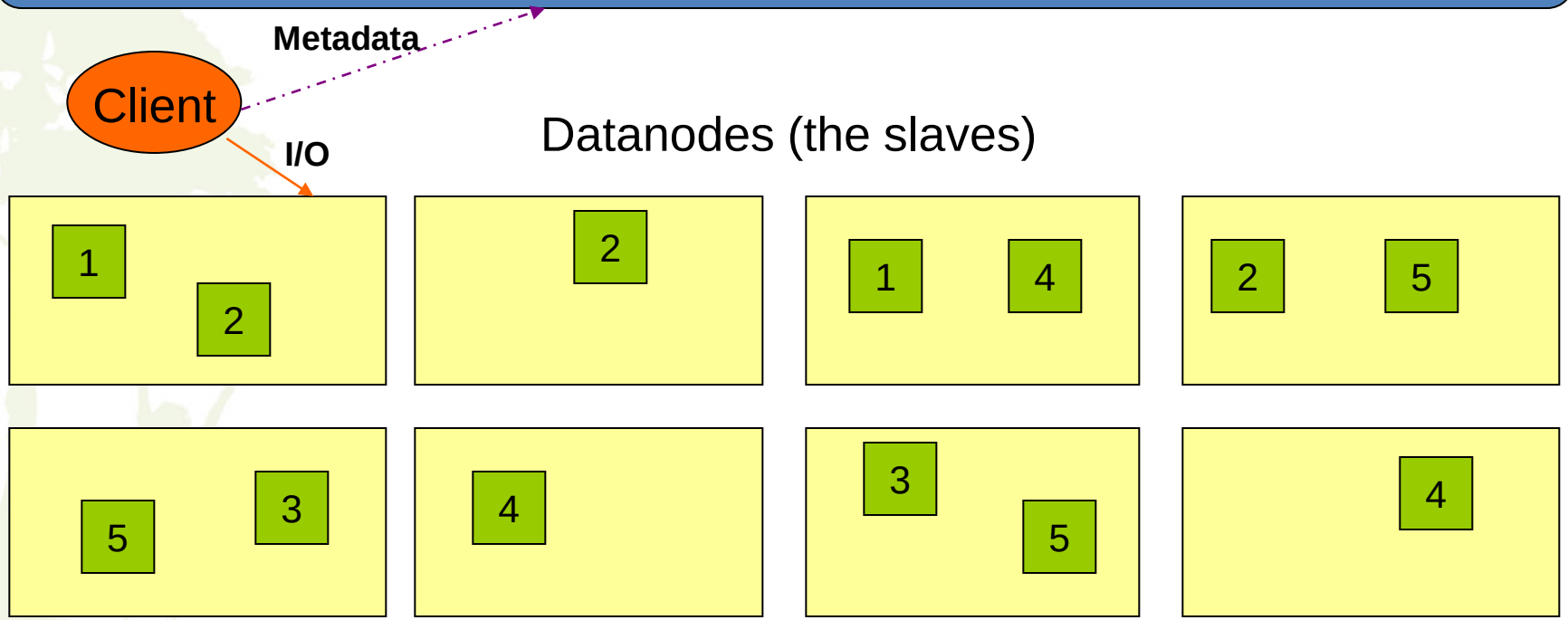


HDFS 運作

Namenode (the master)

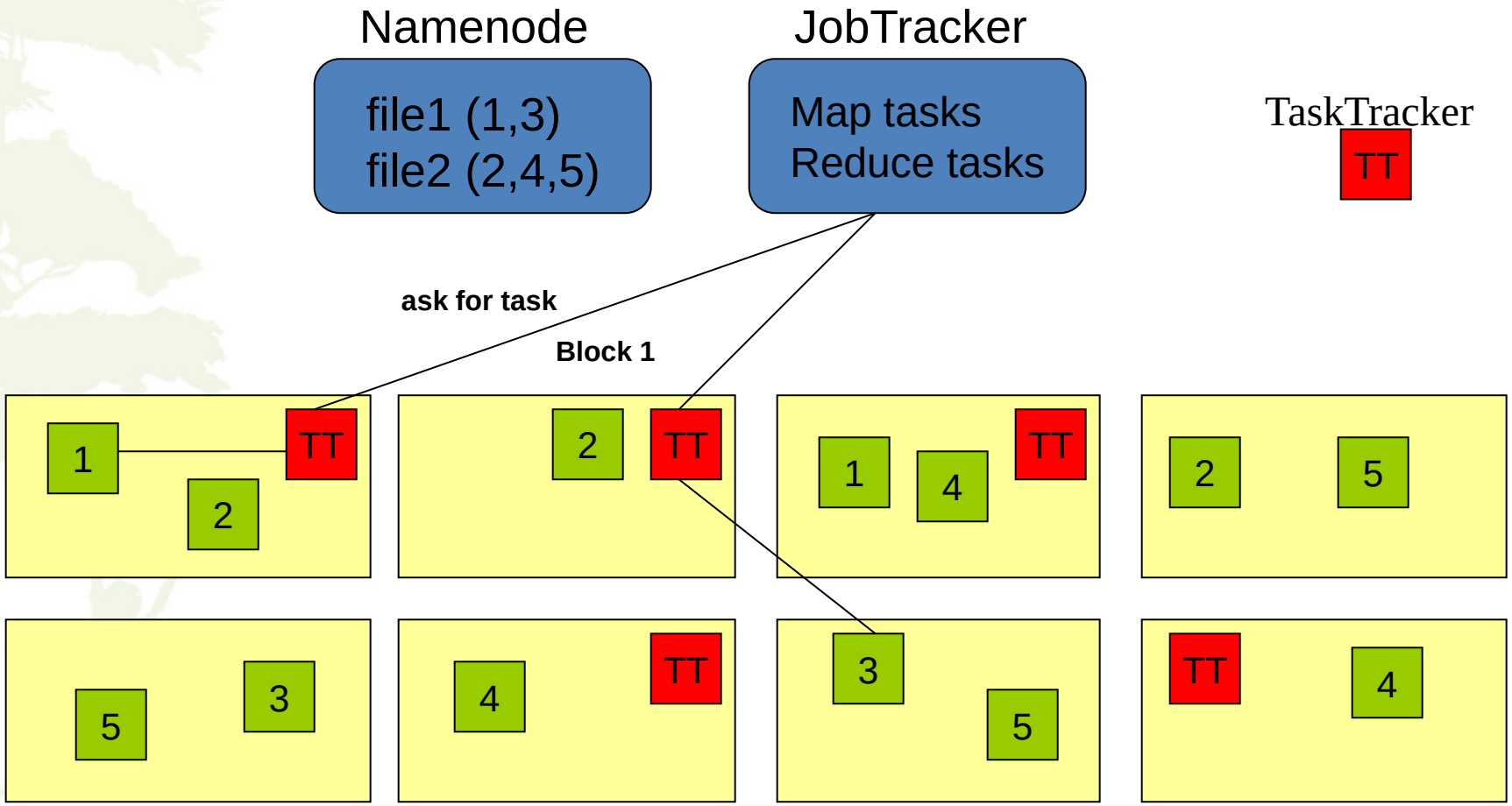
檔案路徑 - 副本數 , 由哪幾個block組成

name:/users/joeYahoo/myFile - copies:2, blocks:{1,3}
name:/users/bobYahoo/someData.gzip, copies:3, blocks:{2,4,5}

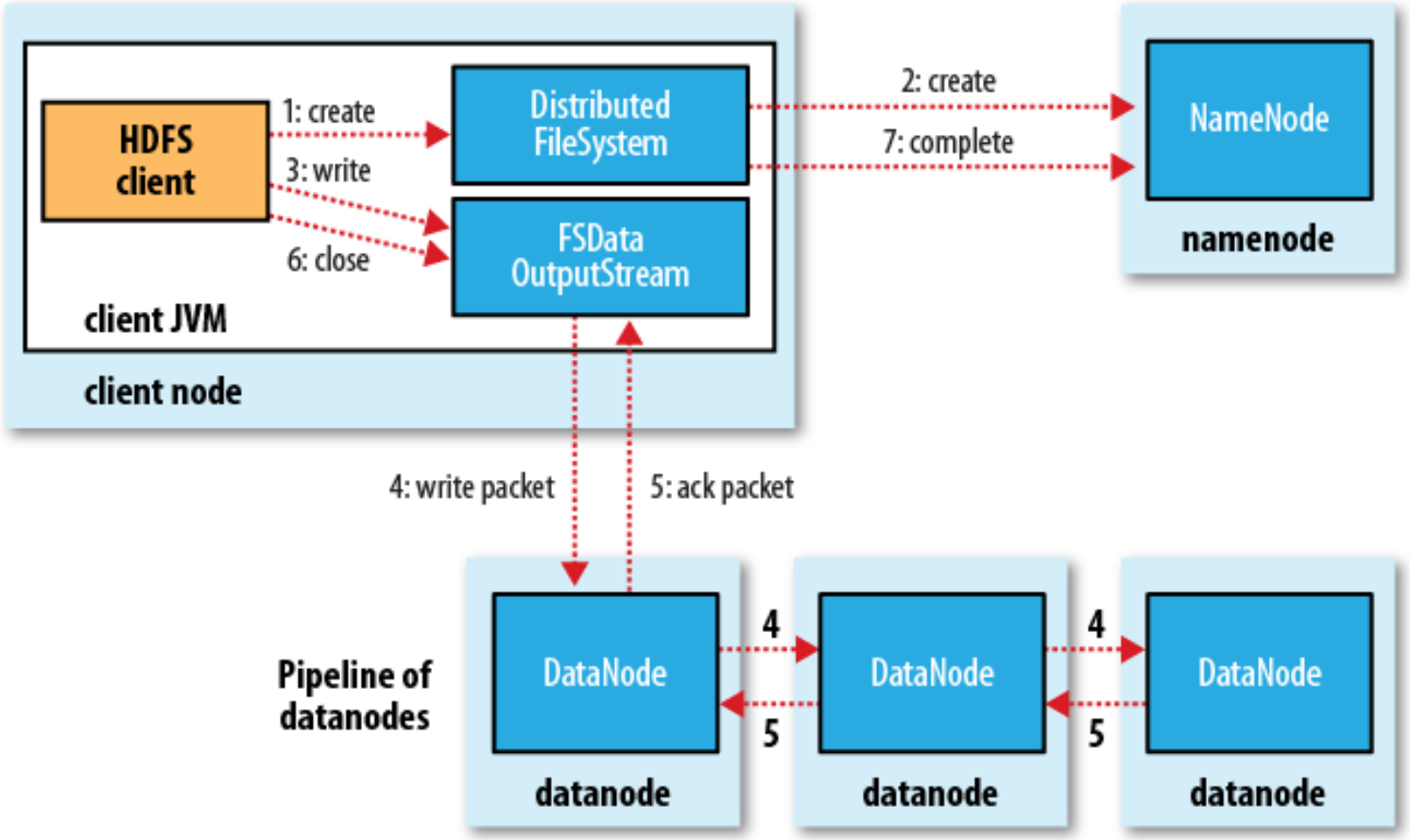


HDFS 運作

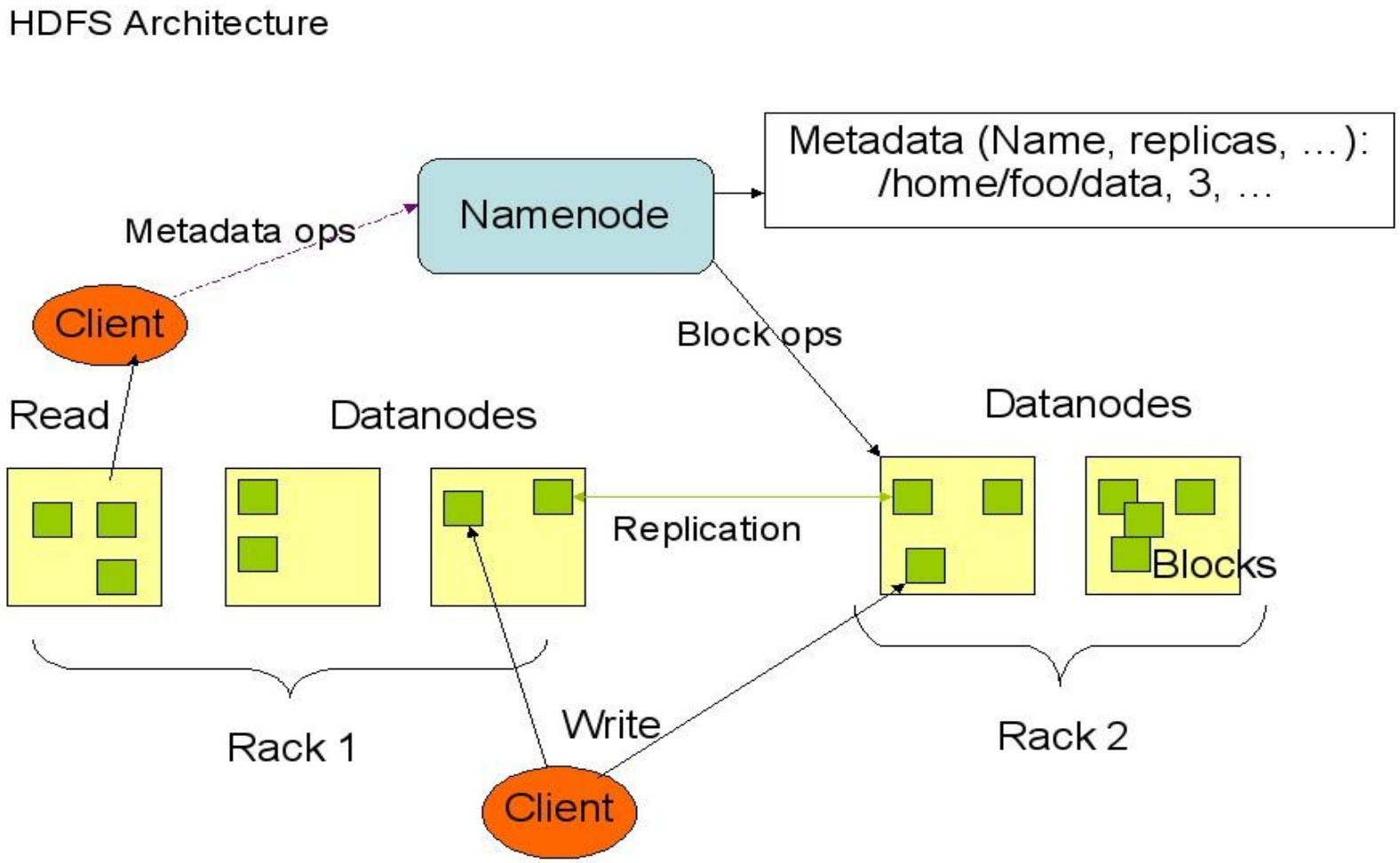
- 目的：提高系統的可靠性與讀取的效率
  - 可靠性：節點失效時讀取副本已維持正常運作
  - 讀取效率：分散讀取流量（但增加寫入時效能瓶頸）



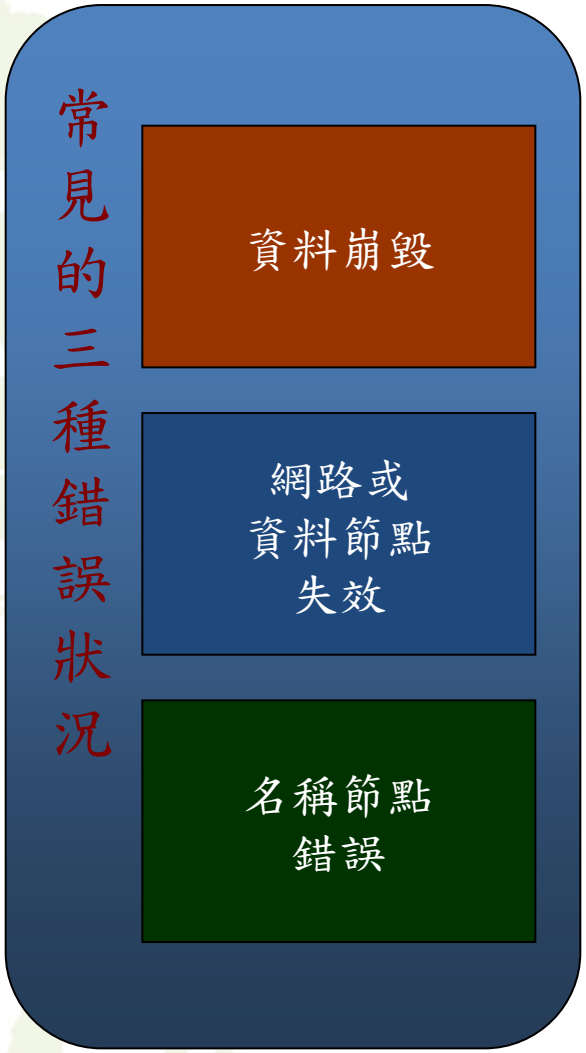
Data Flow - Write



管理資料


HDFS Architecture
Metadata (Name, replicas, …):
/home/foo/data, 3, …
Namenode
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Rack 1
Write
Rack 2
Client

可靠性機制



- **資料完整性**
  - checked with CRC32
  - 用副本取代出錯資料
- **Heartbeat**
  - Datanode 定期向Namenode送heartbeat
- **Metadata**
  - FSImage、Editlog為核心印象檔及日誌檔
  - 多份儲存，當NameNode壞掉可以手動復原

一致性與效能機制

- 檔案一致性機制
  - 刪除檔案＼新增寫入檔案＼讀取檔案皆由Namenode負責
- 巨量空間及效能機制
  - 以Block為單位： 64M為單位
  - 在HDFS上得檔案有可能大過一顆磁碟
  - 大區塊可提高存取效率
  - 區塊均勻散佈各節點以分散讀取流量

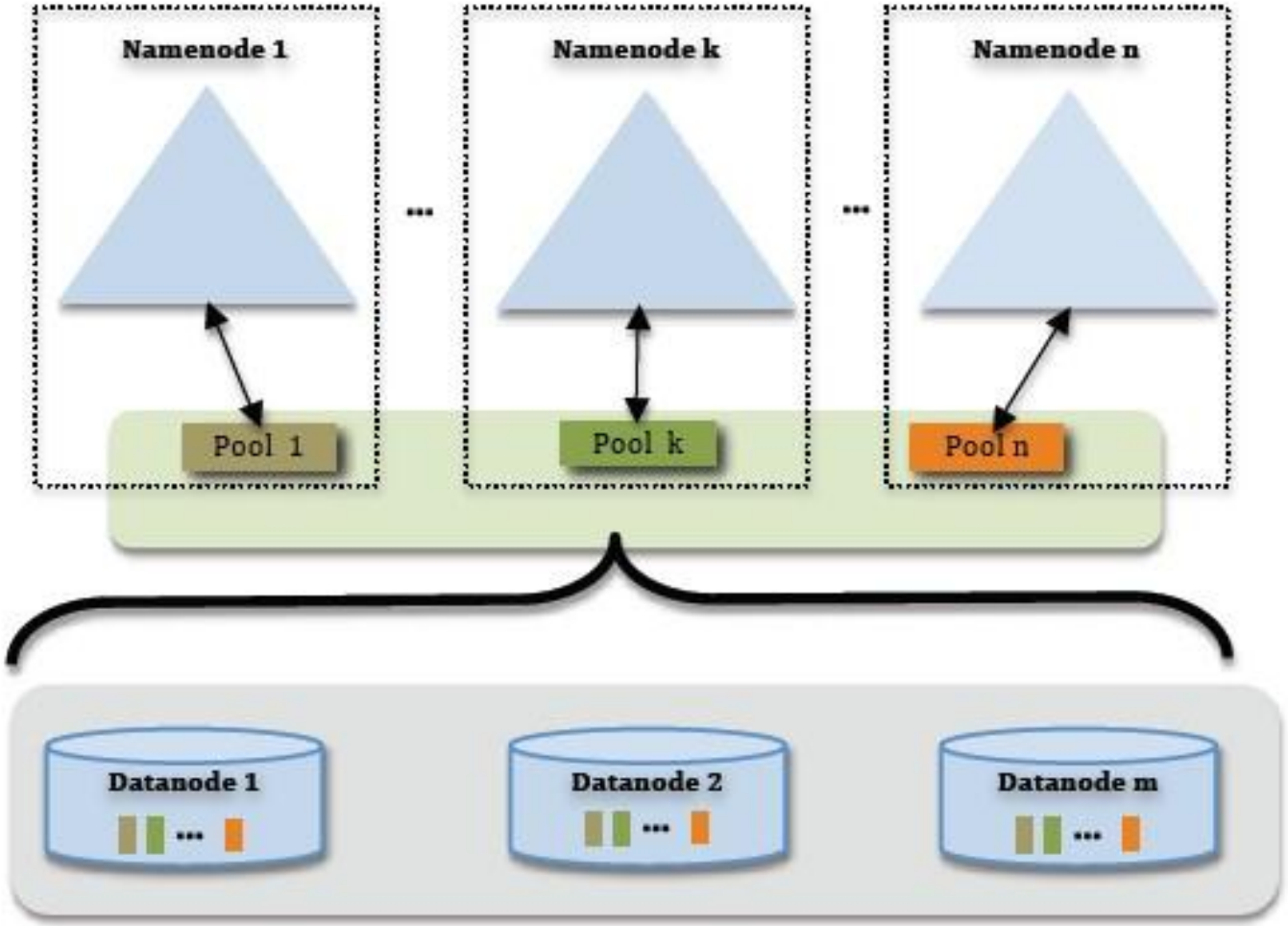
Rebalancer

- **Goal: % disk full on DataNodes should be similar**
  - Usually run when new DataNodes are added
  - Cluster is online when Rebalancer is active
  - Rebalancer is throttled to avoid network congestion
- **Disadvantages**
  - Does not rebalance based on access patterns or load
  - No support for automatic handling of hotspots of data

HDFS Federation(v2.x)

- 解決單一Namenode的問題
  - 拓展性
  - 性能
    - 一個namenode約可support 60k tasks
    - 未來Hadoop將可支援超過100k tasks
  - 隔離性
    - 不同的Group使用相同權限，共享同一個檔案系統

HDFS Federation(v2.x)



HDFS Federation(v2.x)

Current

- HDFS Cluster
  - 1 Namespace
  - A set of blocks
- Implemented as
  - 1 Namenode
  - Set of DNs

New

- HDFS Cluster
  - N Namespaces
  - Set of block-pools
    - Each block-pool is set of blocks
    - Phase 1: 1 BP per NS
      - Implies N block-pools
- Implemented as
  - N Namenode
  - Set of DNs
    - Each DN stores the blocks for each block-pool

User Interface

- **API**
  - Java API
  - C language wrapper for the Java API is also avaiable
- **POSIX like command**
  - hadoop dfs -mkdir /foodir
  - hadoop dfs -cat /foodir/myfile.txt
  - hadoop dfs -rm /foodir myfile.txthadoop dfs -rm /foodir myfile.txt
- **DFSAdmin**
  - bin/hadoop dfsadmin –safemode
  - bin/hadoop dfsadmin –report
  - bin/hadoop dfsadmin -refreshNodes
- **Web管理介面**
  - http://host:port/dfshealth.jsp

POSIX Like

```
hadoop fs [-fs <local | file system URI>] [-conf <configuration file>]
        [-D <property=value>] [-ls <path>] [-lsr <path>] [-du <path>]
        [-dus <path>] [-mv <src> <dst>] [-cp <src> <dst>] [-rm <src>]
        [-rmr <src>] [-put <localsrc> <dst>] [-copyFromLocal <localsrc> <dst>]
        [-moveFromLocal <localsrc> <dst>] [-get <src> <localdst>]
        [-getmerge <src> <localdst> [addnl]] [-cat <src>]
        [-copyToLocal <src><localdst>] [-moveToLocal <src> <localdst>]
        [-mkdir <path>] [-report] [-setrep [-R] [-w] <rep> <path/file>]
        [-touchz <path>] [-test -[ezd] <path>] [-stat [format] <path>]
        [-tail [-f] <path>] [-text <path>]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-chgrp [-R] GROUP PATH...]
        [-help [cmd]]
```

Web管理介面

NameNode 'localhost:9000'

Started:	Thu Sep 13 09:40:41 CST 2012
Version:	0.20.2, r911707
Compiled:	Fri Feb 19 08:07:34 UTC 2010 by chrisdo
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

8 files and directories, 1 blocks = 9 total. Heap Size is 15.31 MB / 966.69 MB (1%)

Configured Capacity	:	18.82 GB
DFS Used	:	36 KB
Non DFS Used	:	6.8 GB
DFS Remaining	:	12.02 GB
DFS Used%	:	0 %
DFS Remaining%	:	63.86 %
Live Nodes	:	1
Dead Nodes	:	0

MapReduce 介紹

楊順發
shunfa@nchc.narl.org.tw
自由軟體實驗室
國家高速網路與計算中心

Outline

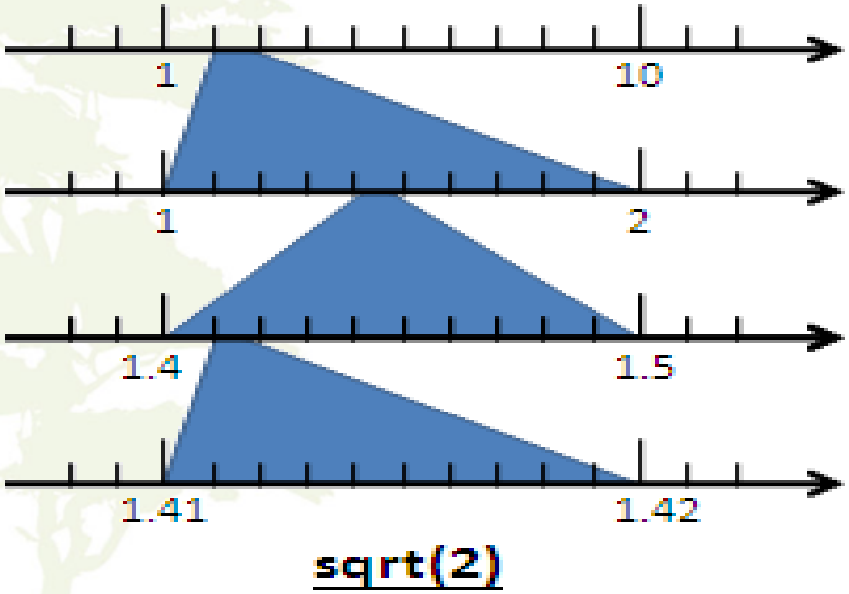
- 概念
  - Divide and Conquer
- **MapReduce起源**
- **MapReduce 運作流程**
- **資料分析範例 – Using MapReduce**
- **MapReduce Works Detial**

Map Reduce 起源

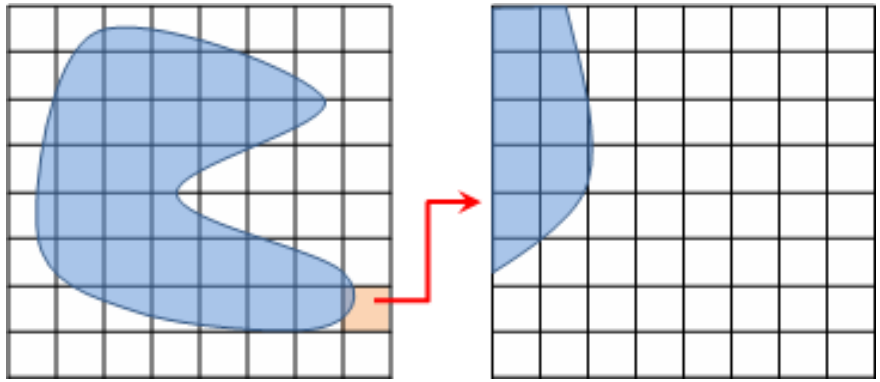
- 演算法(Algorithms)
  - Divide and Conquer
  - 分而治之
- 在程式設計的軟體架構內，適合使用在大規模數據的運算中

Divide and Conquer

範例一：十分逼近法

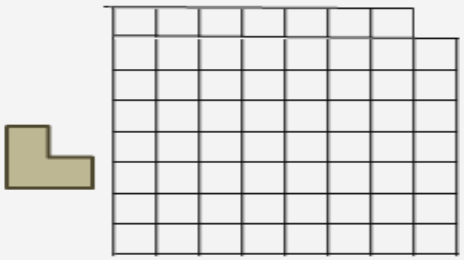


範例二：方格法求面積



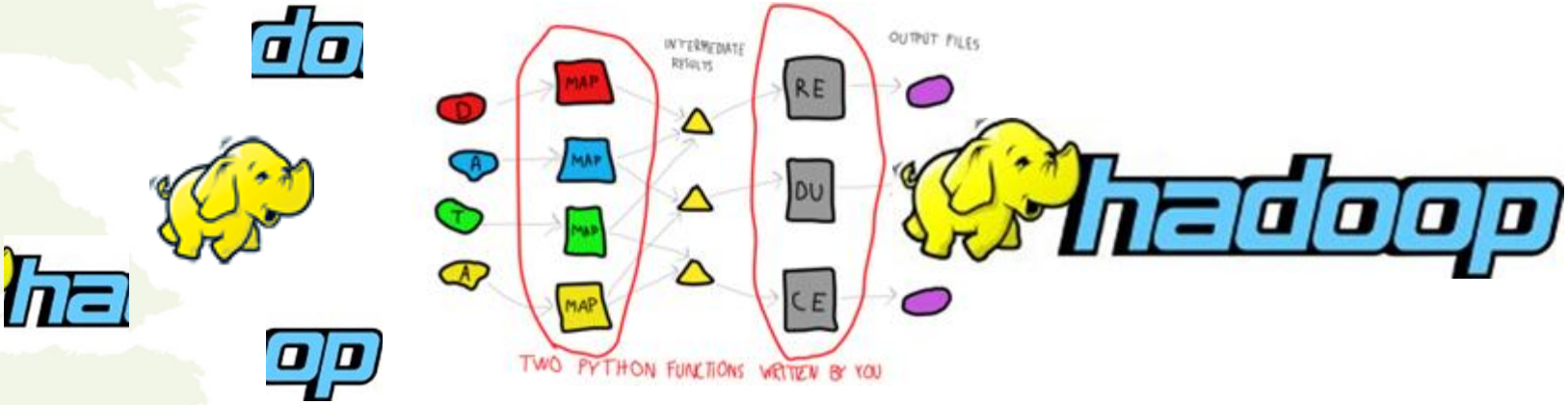
Divide and Conquer

範例三：鋪滿 L 形磁磚



範例四：
眼前有五階樓梯，每次可踏上一階或踏上兩階，那麼爬完五階共有幾種踏法？
Ex : (1,1,1,1,1) or (1,2,1,1)

Hadoop MapReduce 定義



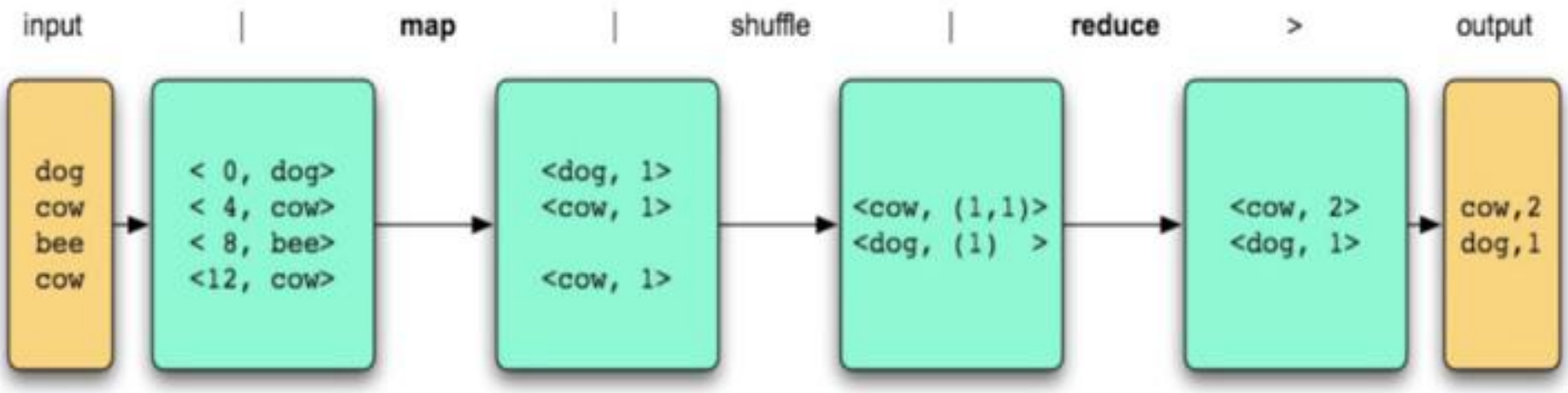
Hadoop Map/Reduce是一個易於使用的軟體平台，以MapReduce為基礎的應用程序，能夠運作在由上千台PC所組成的大型叢集上，並以一種可靠容錯的方式平行處理上Peta-Bytes數量級的資料集。

適用範圍

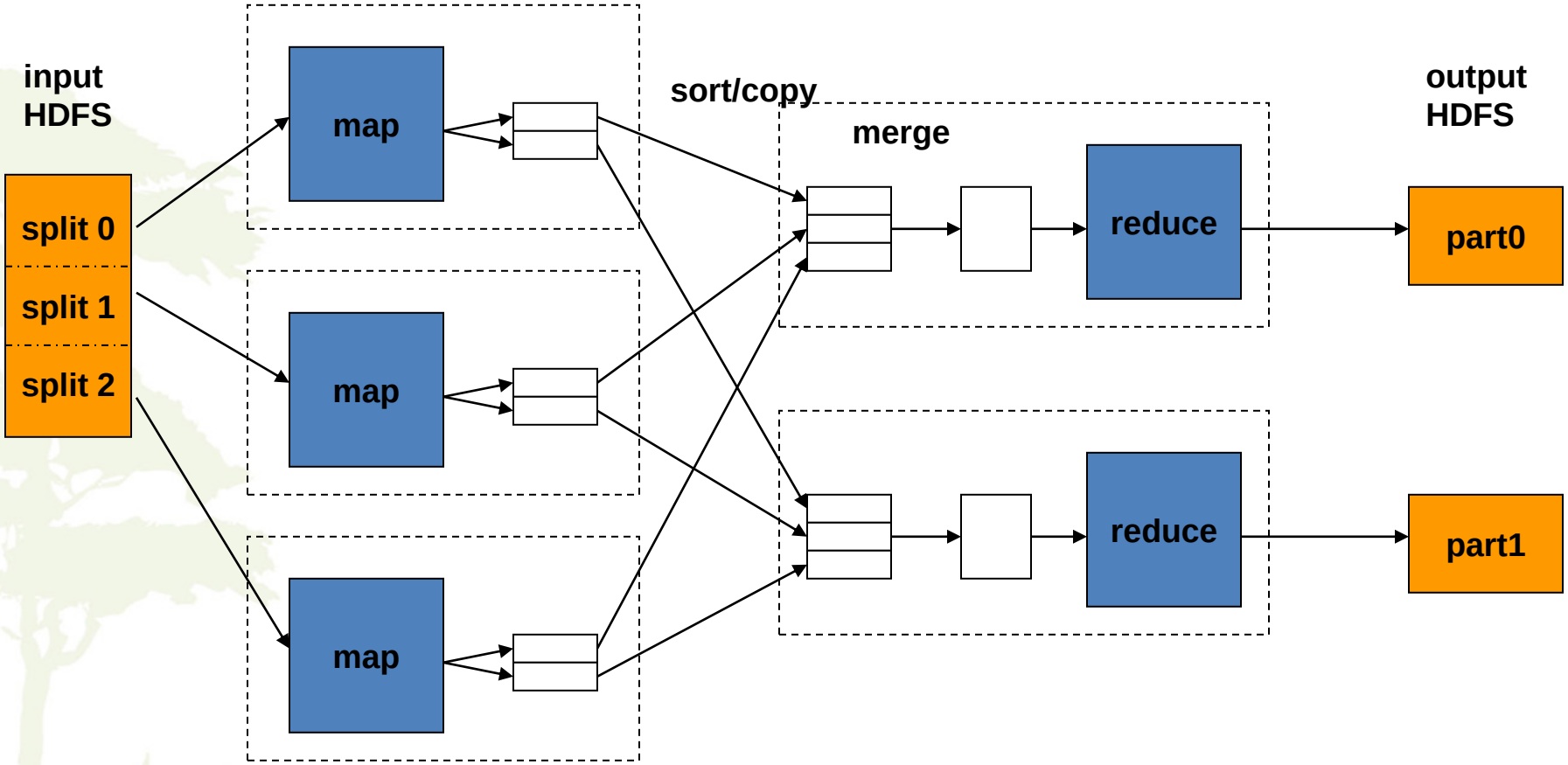
- 大規模資料集
- 可拆解

- **Text tokenization**
- **Indexing and Search**
- **Data mining**
- **machine learning**
- **…**

MapReduce Overview



MapReduce Overview



JobTracker跟NameNode取得需要運算的blocks

JobTracker選數個TaskTracker來作Map運算，產生些中間檔案

JobTracker將中間檔案整合排序後，複製到需要的TaskTracker去

JobTracker派遣TaskTracker作reduce

reduce完後通知JobTracker與Namenode以產生output

Map

- 輸入是”一個” key/value 序對 ，輸出則為另”一組”intermediate key/value 序對組。

Example: Upper-case Mapper

```
let map(k, v) =
  emit(k.toUpper(), v.toUpper())
```

("foo", "bar") → ("FOO", "BAR")
("Foo", "other") → ("FOO", "OTHER")
("key2", "data") → ("KEY2", "DATA")

Example: Explode Mapper

```
let map(k, v) =
  foreach char c in v:
    emit(k, c)
```

("A", "cats") ➔ ("A", "c"), ("A", "a"),
("A", "t"), ("A", "s")

("B", "hi") ➔ ("B", "h"), ("B", "i")

Example: Filter Mapper

```
let map(k, v) =
  if (isPrime(v)) then emit(k, v)
```

("foo", 7) → ("foo", 7)

("test", 10) → (nothing)

Example: Changing Keyspaces

```
let map(k, v) = emit(v.length(), v)
```

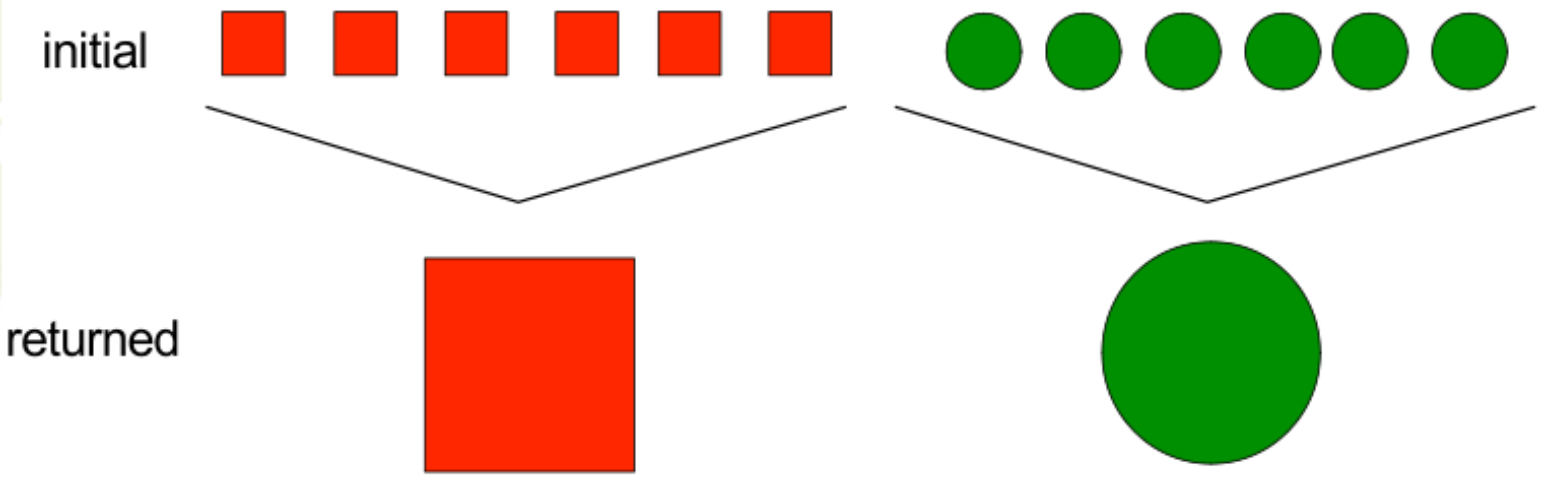
("hi", "test") → (4, "test")
("x", "quux") → (4, "quux")
("y", "abracadabra") → (10, "abracadabra")

Reduce

- 數負責針對相同的 intermediate key 合併其所有相關聯的intermediate values。並產生輸出結果的key/value序對

Reduce

```
reduce (inter_key, inter_value list) ->
       (out_key, out_value) list
```



Reduce

- **Example: sum reducer**

```
let reduce(k, vals)=
sum = 0
foreach int v in vals:
sum +=v
emit(k,sum)
```

(“A”, [42, 100, 312]) → (“A”, 454)

(“B”, [12, 6, -2]) → (“B”, 16)

Reduce

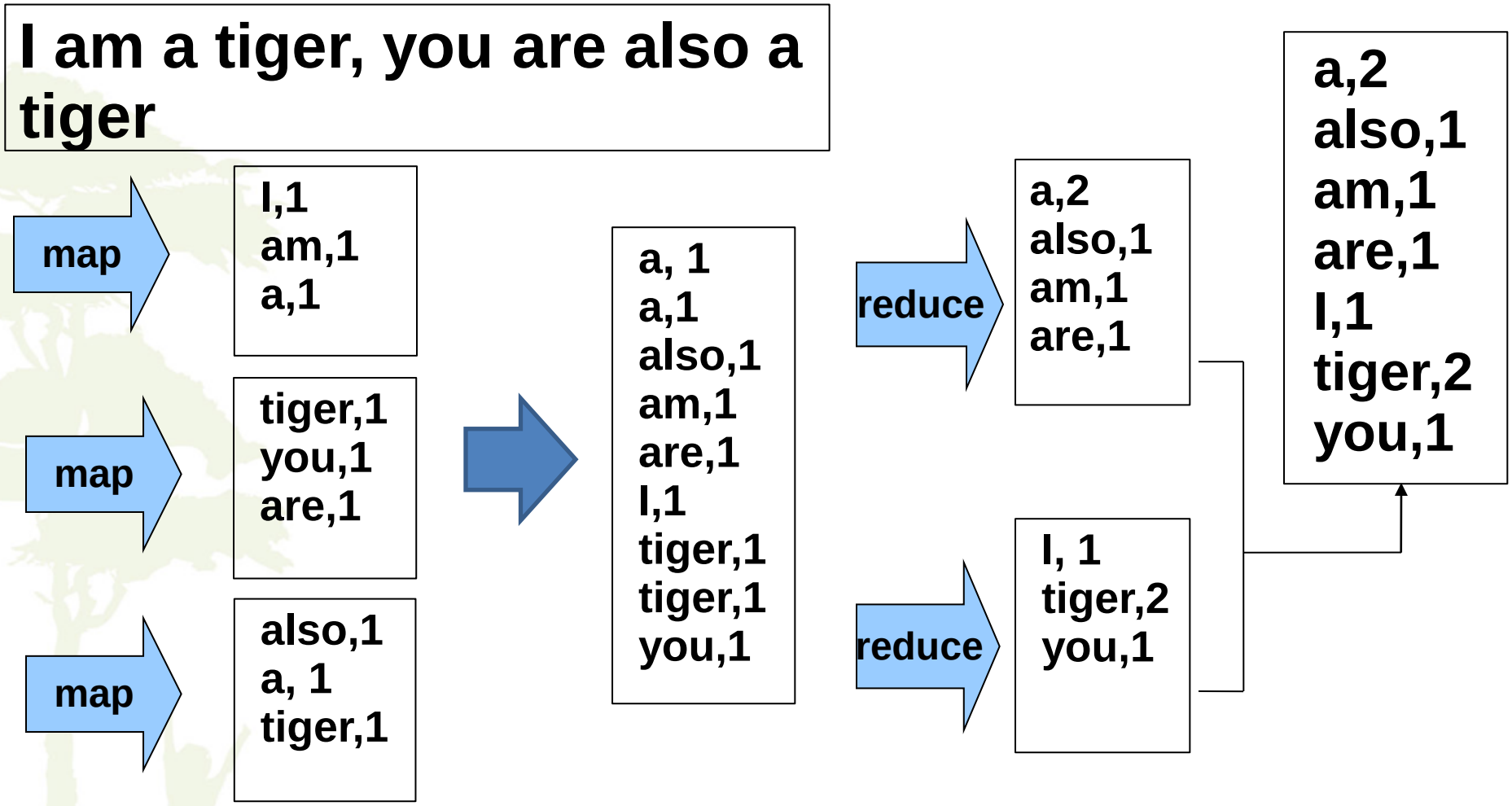
- **Example: Identity Reducer**

```
let reduce(k, vals) =
  foreach v in vals:
    emit(k, v)
```

("A", [42, 100, 312]) ➔ ("A", 42),
("A", 100), ("A", 312)

("B", [12, 6, -2]) ➔ ("B", 12), ("B", 6),
("B", -2)

MapReduce範例



JobTracker先選了三個Tracker做map

Map結束後，hadoop進行中間資料的整理與排序

JobTracker再選兩個TaskTracker作reduce

回到以美國氣象資料為例：

0057
332130 # USAF weather station identifier
99999 # WBAN weather station identifier
19500101 # observation date
0300 # observation time
4
+51317 # latitude (degrees x 1000)
+028783 # longitude (degrees x 1000)
FM-12
+0171 # elevation (meters)
99999
V020
320 # wind direction (degrees)
1 # quality code
N
0072
1
00450 # sky ceiling height (meters)
1 # quality code
CN
010000 # visibility distance (meters)
1 # quality code
N9
-0128 # air temperature (degrees Celsius x 10)
1 # quality code
-0139 # dew point temperature (degrees Celsius x 10)
1 # quality code
10268 # atmospheric pressure (hectopascals x 10)
1 # quality code

資料分析 – MapReduce

- 原始資料

0067011990999991950051507004...9999999N9+00001+99999999999...
0043011990999991950051512004...9999999N9+00221+99999999999...
0043011990999991950051518004...9999999N9-00111+99999999999...
0043012650999991949032412004...0500001N9+01111+99999999999...
0043012650999991949032418004...0500001N9+00781+99999999999...

- **Map – 擷取所需的資訊**

(0, 0067011990999991950051507004...9999999N9+**0000**1+99999999999...)
(106, 0043011990999991950051512004...9999999N9+**0022**1+99999999999...)
(212, 0043011990999991950051518004...9999999N9-**0011**1+99999999999...)
(318, 0043012650999991949032412004...0500001N9+**0111**1+99999999999...)
(424, 0043012650999991949032418004...0500001N9+**0078**1+99999999999...)

資料分析 – MapReduce

- **Map - shuffle**

(1950, 0)
(1950, 22)
(1950, −11)
(1949, 111)
(1949, 78)

- **Map - Output:<Key, Value>**

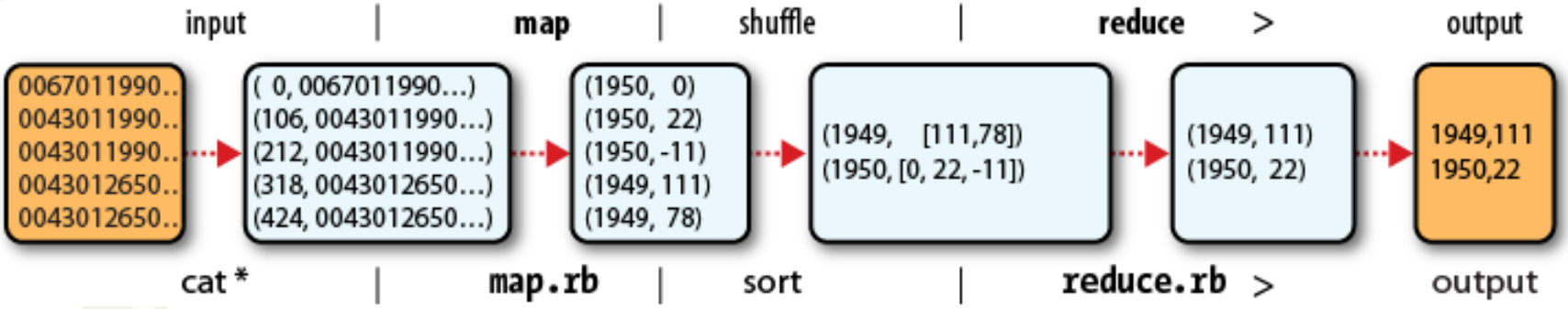
(1949, [111, 78])
(1950, [0, 22, −11])

資料分析 – MapReduce

- **Reduce - Output: <Year, Max>**

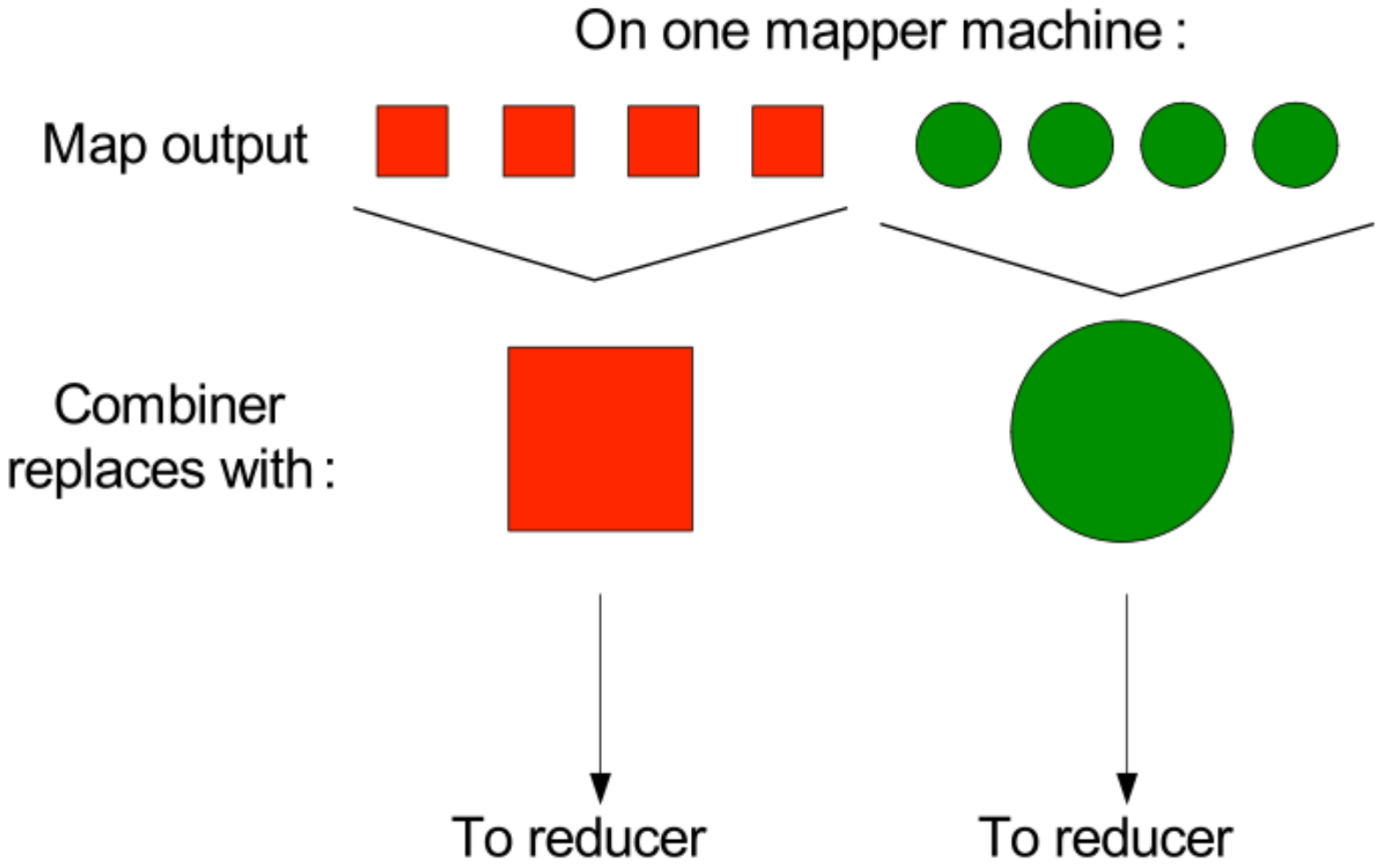
  (1949, 111)
  (1950, 22)

- **Logical Flow**



- ***費時：6 min on EC2 High-CPU Extra Large Instance***

Combiner


On one mapper machine :
Map output
Combiner
replaces with :
To reducer
To reducer

Ref.

- Hadoop 開發訓練課程 - CCHD 課程講義

- Hadoop - http://hadoop.apache.org/

MapReduce 程式設計

楊順發

shunfa@nchc.narl.org.tw

自由軟體實驗室

國家高速網路與計算中心

Outline

- 概念
- 程式基本框架及執行步驟方法
- 範例一:
  - Hadoop 的Hello World => Word Count
  - 說明
  - 動手做
- 範例二:
  - 進階版 Word Count
  - 說明
  - 動手做

程式基本框架

Map 區

Reduce 區

設定區

```
Class MR{
    Class Mapper …{
    }
    Class Reducer …{
    }
    main(){
        JobConf conf = new JobConf( MR.class );
        conf.setMapperClass(Mapper.class);
        conf.setReduceClass(Reducer.class);
        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new
    Path
        JobClient.runJob(conf);
}}
```

Map 程式碼

Reduce 程式碼

其他的設定參數程式碼

Class Mapper(0.20)

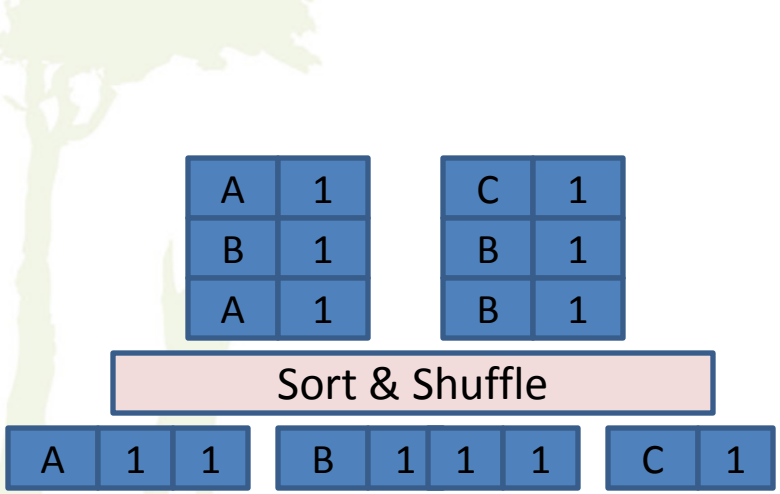
```
1  class MyMap extends MapReduceBase
   implements Mapper < INPUT KEY , INPUT VALUE , OUTPUT KEY , OUTPUT VALUE >
2
3  // 全域變數區
4  public void map ( INPUT KEY key, INPUT VALUE value, Context context)
   throws IOException, InterruptedException
           {
5          // 區域變數與程式邏輯區
6          context.write( NewKey, NewValue);
7          }
8  }
9
```

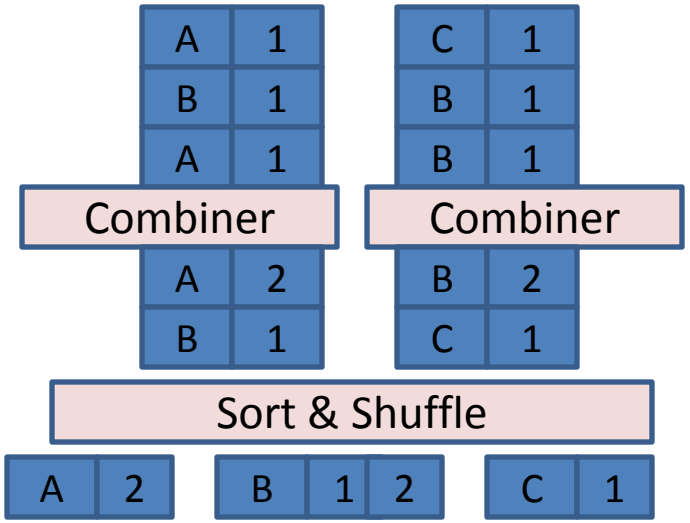
Class Reducer(0.20)

```
1  class MyRed extends MapReduceBase
   implements Reducer < INPUT KEY , INPUT VALUE , OUTPUT KEY , OUTPUT VALUE >
2
3  // 全域變數區
4  public void reduce ( INPUT KEY key, Iterator< INPUT VALUE > values,
           Context context) throws IOException, InterruptedException
           {
5          // 區域變數與程式邏輯區
6          output.collect( NewKey, NewValue);
7          }
8  }
9
```

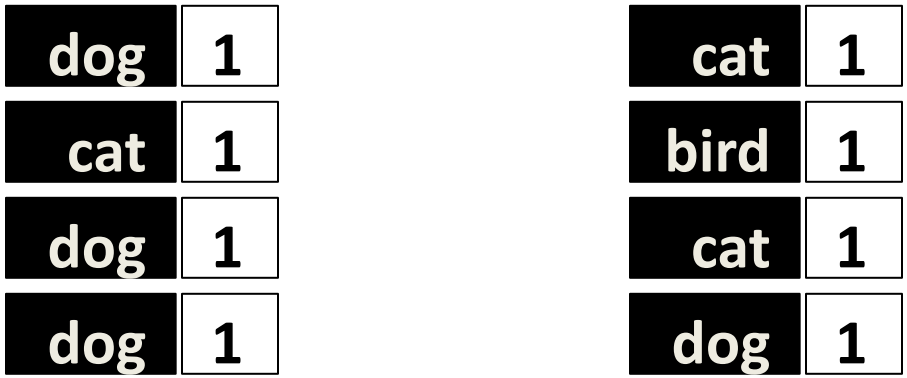
Class Combiner

- 指定一個combiner，它負責對中間過程的輸出進行聚集，這會有助於降低從Mapper 到 Reducer數據傳輸量。
  - 引用Reducer
  - JobConf.setCombinerClass(Class)





No Combiner



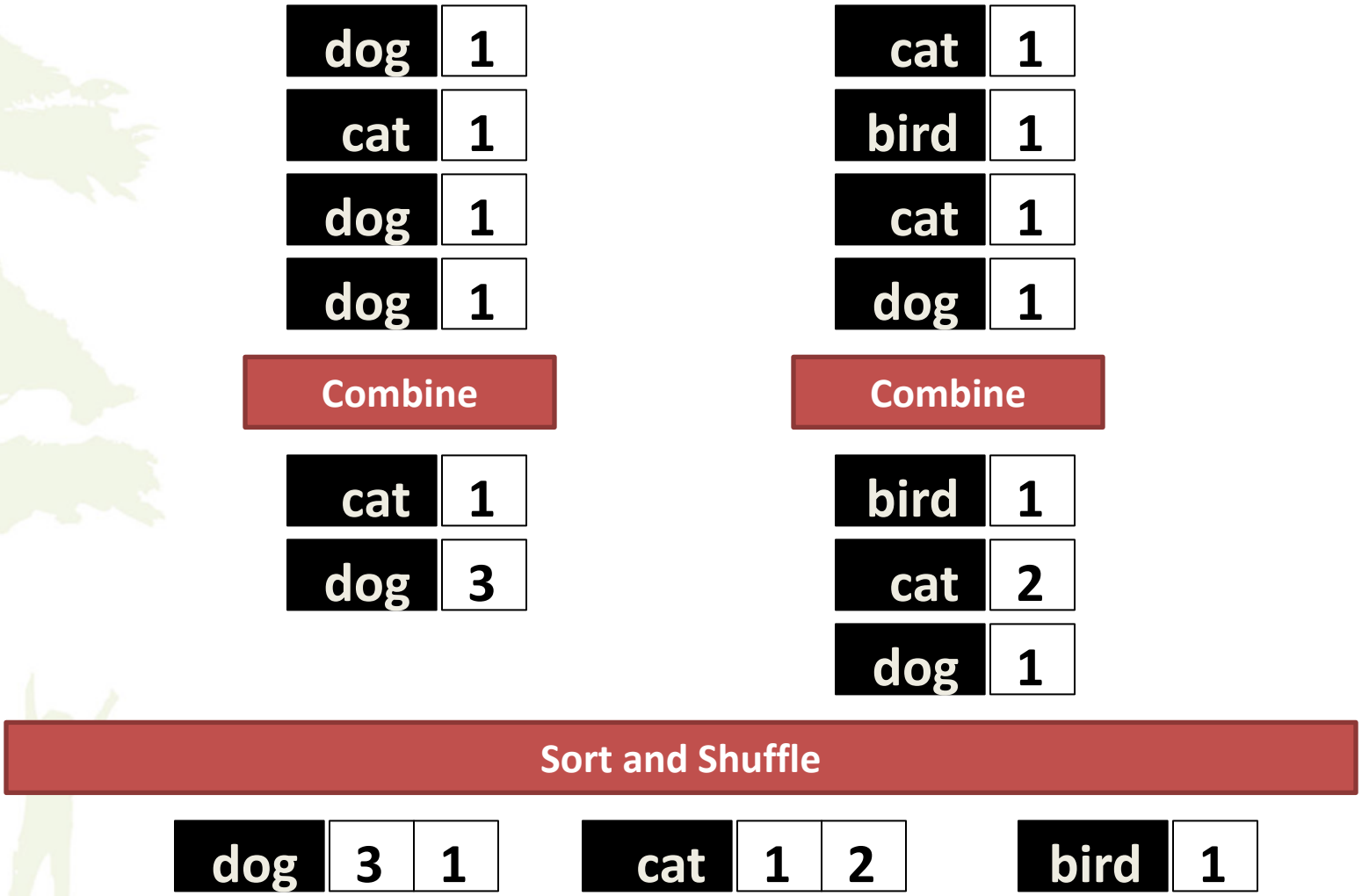
Sort and Shuffle

Reduce class

Reduce method (key, values)

```
for each v in values
        sum ← sum + v
        EMIT(key, sum)
```

Added Combiner



HelloHadoop.java

```
package org.nchc.hadoop;
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class HelloHadoop {
  static public class HelloMapper extends
      Mapper<LongWritable, Text, LongWritable, Text> {
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
      context.write((LongWritable) key, (Text) value);
    }
  }

  static public class HelloReducer extends
      Reducer<LongWritable, Text, LongWritable, Text> {
    public void reduce(LongWritable key, Iterable<Text> values,
        Context context) throws IOException, InterruptedException {
      Text val = new Text();
      for (Text str : values) {
        val.set(str.toString());
      }
      context.write(key, val);
    }
  }

  public static void main(String[] args) throws IOException,
      InterruptedException, ClassNotFoundException {
    Configuration conf = new Configuration();
    Job job = new Job(conf, "Hadoop Hello World");
    job.setJarByClass(HelloHadoop.class);
    FileInputFormat.setInputPaths(job, "/user/hadooper/input_exText");
    FileOutputFormat.setOutputPath(job, new
Path("/user/hadooper/output_helloHadoop"));
    job.setMapperClass(HelloMapper.class);
    job.setReducerClass(HelloReducer.class);
    job.waitForCompletion(true);

  }
}
```

編譯

- 目標：產生Hadoop的jar執行檔

1. 編譯
   - javac △ -classpath △ hadoop-*-core.jar △ -d △ MyJavaDir △ MyCode.java
2. 封裝
   - jar △ -cvf △ MyJar.jar △ -C △ MyJavaDir △ .
3. 執行
   - bin/hadoop △ jar △ MyJar.jar △ MyCode △ arg0 △ arg1 ……

測試HelloHadoop

- 置入運算文件檔

```
$ cd /opt/hadoop/
$ mkdir input_exText
$ vim input_exText /text
$ bin/hadoop fs –put input_exText input_exText
```

Text
Hello
Hadoop
HDFS
MapReduce

- 執行

```
$ cd /opt/hadoop/
$ bin/hadoop jar helloHadoop.jar org.nchc.hadoop.HelloHadoop
```

Options without Java

- 雖然Hadoop框架是用Java實作，但Map/Reduce應用程序則不一定要用Java來寫
- Hadoop Streaming ：
  - 執行作業的工具，使用者可以用其他語言（如：PHP）套用到Hadoop的mapper和reducer
- Hadoop Pipes：C++ API

補充

- **Mapper 數量**
  - Mapper的數量由Input的大小決定。Input資料被Hadoop切割成幾份就會有多少個Mapper。
  - 可以用JobConf物件的setNumMapTasks(int)來提示Hadoop關於Mapper的數量。但決定權仍由Hadoop控制。
- **Reducer數量**
  - 可以用JobConf物件的JobConf.setNumReduceTasks(int)來直接控制Reducer的數量。決定權可由使用者控制 。
  - 增加Reducer的數量會讓系統的負荷增加，但增加Reducer數量也可以讓整個MapReduce的Map與Reduce的工作平衡有好的成效，同時也增加容錯性。

程式設計開發工具介紹

楊順發
shunfa@nchc.narl.org.tw
自由軟體實驗室
國家高速網路與計算中心

參考文件

- **Eclipse 3.3.2 + Hadoop 0.20.2**
  - http://trac.nchc.org.tw/cloud/wiki/NCHCCloudCourse100928_2_IDE

Hadoop應用 - Crawlzilla 搜尋引擎安裝與實作

楊順發
shunfa@nchc.narl.org.tw
自由軟體實驗室
國家高速網路與計算中心

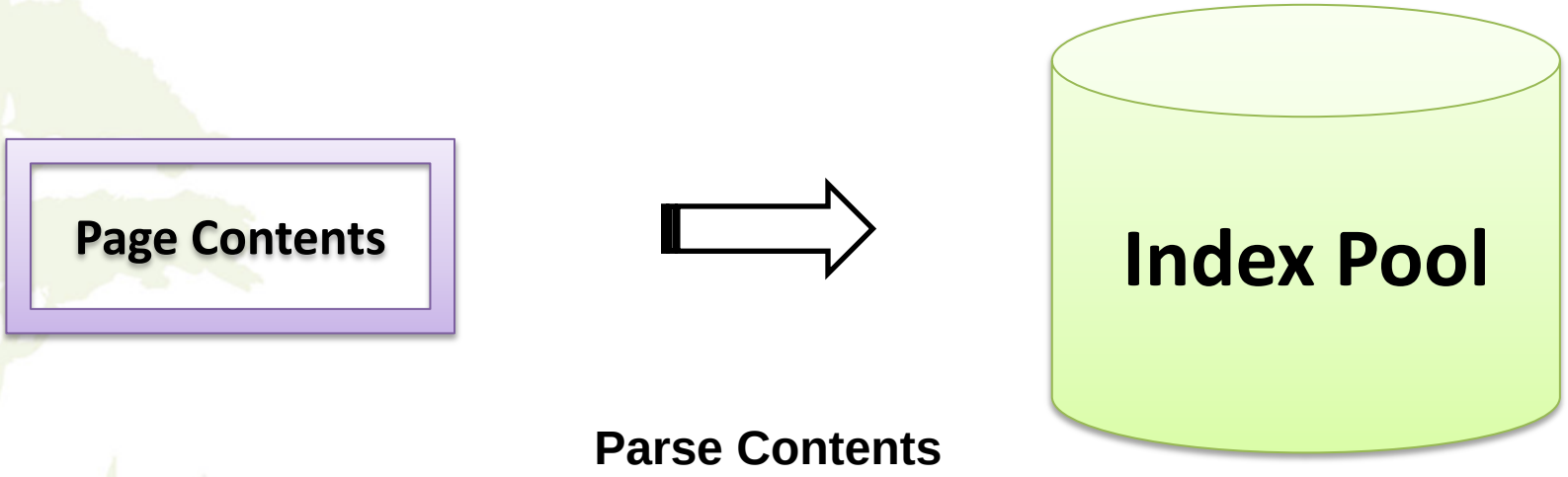
搜尋引擎運作原理 – Phase1

- **Crawling the Web**

Crawler visits the web pages of the links

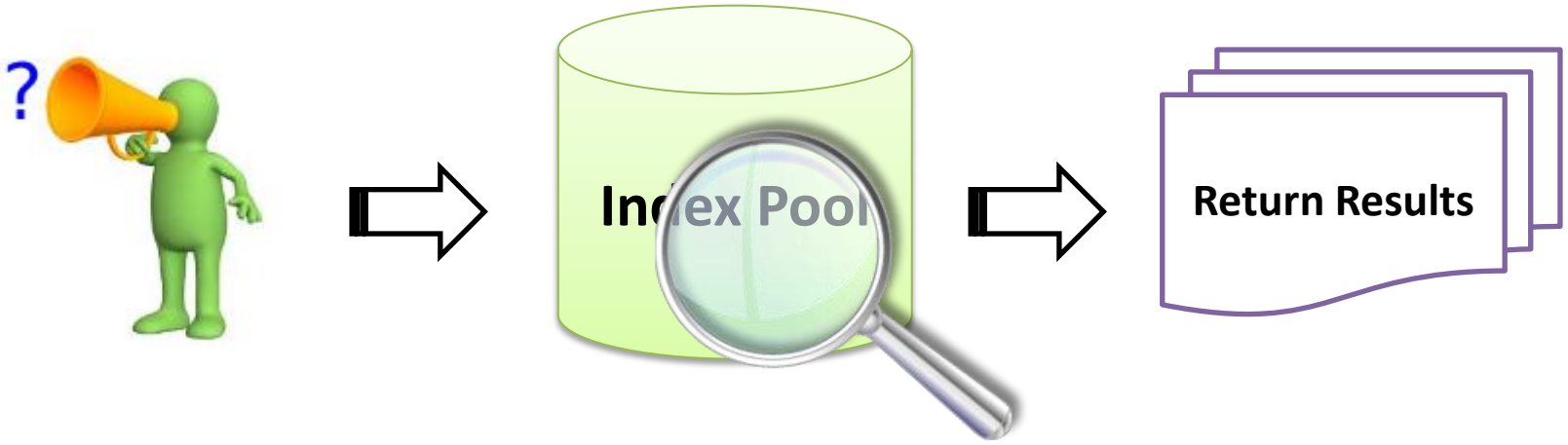
搜尋引擎運作原理 – Phase2

- **Building the Index Pool**



搜尋引擎運作原理 – Phase3

- **Serving Queries**



User Sent a Query **Search from Index Pool**

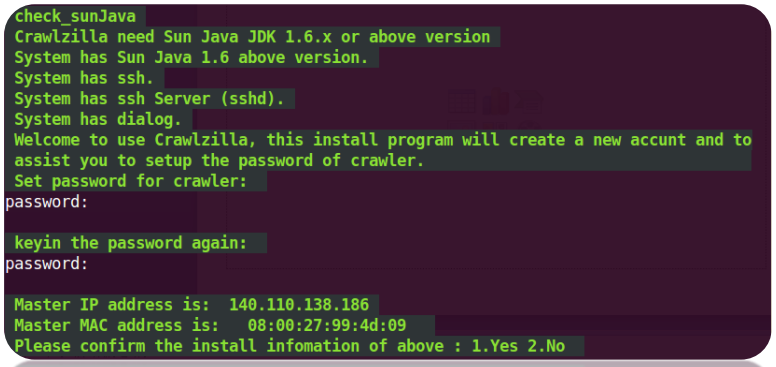
What is Crawlzilla?

- Crawlzilla 簡介
  - 於2009推出實驗版
  - Crawlzilla 於2010更名並延續實驗版開發更多新功能
  - 提供簡單安裝及操作管理介面，輕鬆建立搜尋引擎的套件工具
  - 提供索引資料庫瀏覽功能，搜尋引擎資料庫資訊一目了然

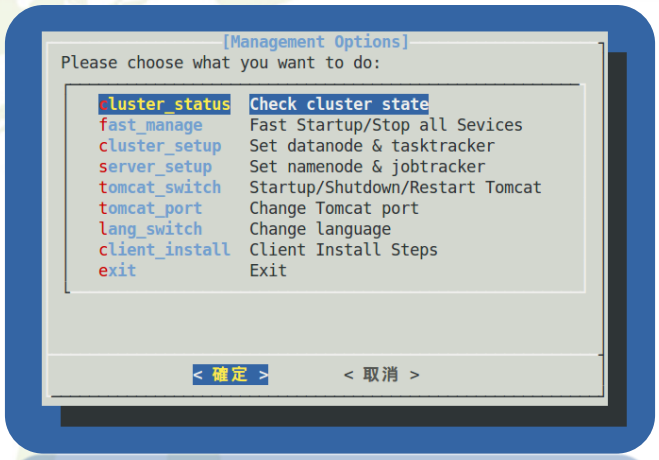
Why Crawlzilla?

- 開放式搜尋引擎不適用於企業內部網站
- 使用Opensource建立搜尋引擎的技術門檻太高
- 叢集環境架設不易
- 使用Crawlzilla優點
  - Opensource專案，使用者可依自己的需求修改源始碼
  - 使用簡單，可輕鬆建立叢集環境
  - 友善的操作環境，節省適應系統時間
  - 支援中文分詞，提高搜尋精準度

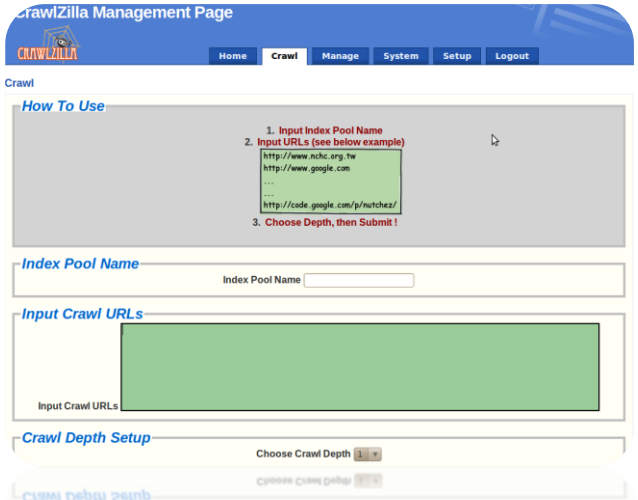
Crawlzilla 操作介面特色



(1) Easy to Deploy Crawling Cluster Environment



(2) Easy to Manage

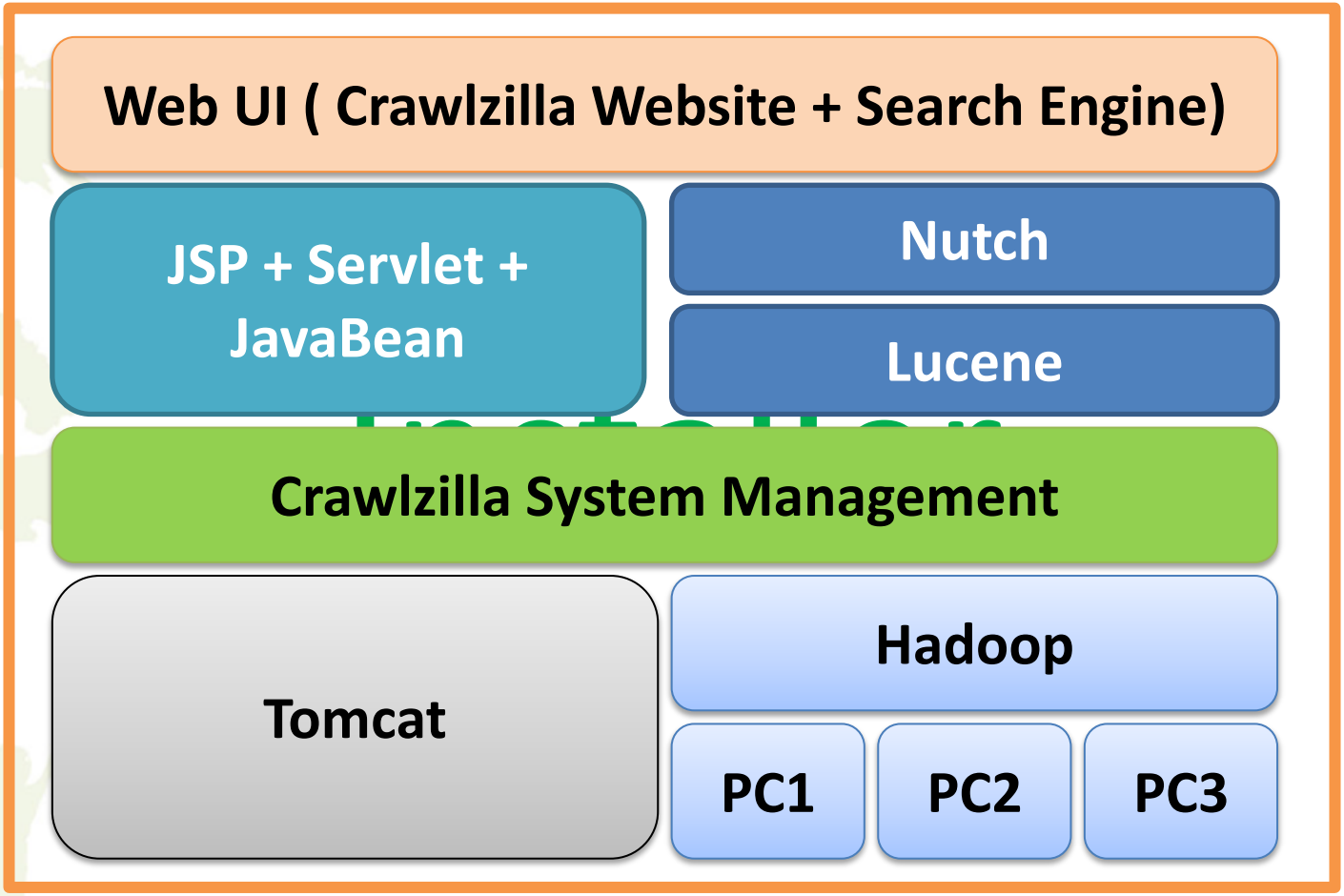


(3) Easy to Use

Crawlzilla 系統功能

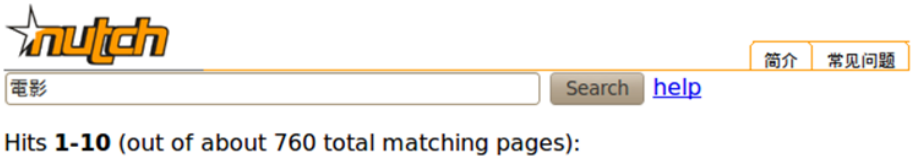
- 支援叢集運算及顧全安全性
- 支援中文分詞功能
- 支援多工網頁爬取
- 支援多重搜尋引擎
- 即時瀏覽資料庫資訊
- 解決中文亂碼及中文支援
- 支援多國語言
- 網頁管理

系統架構


Web UI (Crawlzilla Website + Search Engine)
JSP + Servlet + JavaBean
Nutch
Lucene
Crawlzilla System Management
Tomcat
Hadoop
PC1
PC2
PC3

搜尋引擎加入中文分詞功能

- 索引資料庫會以中文字詞為基本單位建立索引
- 加入中文分詞針對同一網站爬取進行搜尋
  - 搜尋引擎無中文分詞功能時，搜尋關鍵字 - 電影
    - *760* 筆搜尋結果



  - 搜尋引擎加入中文分詞功能時，搜尋關鍵字 - 電影
    - *43* 筆搜尋結果
    - 可提高搜尋的精準度

Crawlzilla - 叢集環境需求

- 如果你覺得...
  - 一台電腦無法滿足你的運算需求
  - 閒置電腦太多
  - 解：讓多台電腦分工運算
- 但是...
  - 架設叢集環境很麻煩!?
  - 解：Crawlzilla 提供叢集安裝模式，只要三分鐘即可建立叢集式搜尋引擎!!!

Resources

- Crawlzilla @ Google Code Project Hosting (中文說明頁)
  - http://code.google.com/p/crawlzilla/
- Crawlzilla @ SourceForge(英文說明頁)
  - http://sourceforge.net/p/crawlzilla/home/
- Crawlzilla User Group @ Google
  - http://groups.google.com/group/crawlzilla-user
- NCHC Cloud Computing Research Group
  - http://trac.nchc.org.tw/cloud

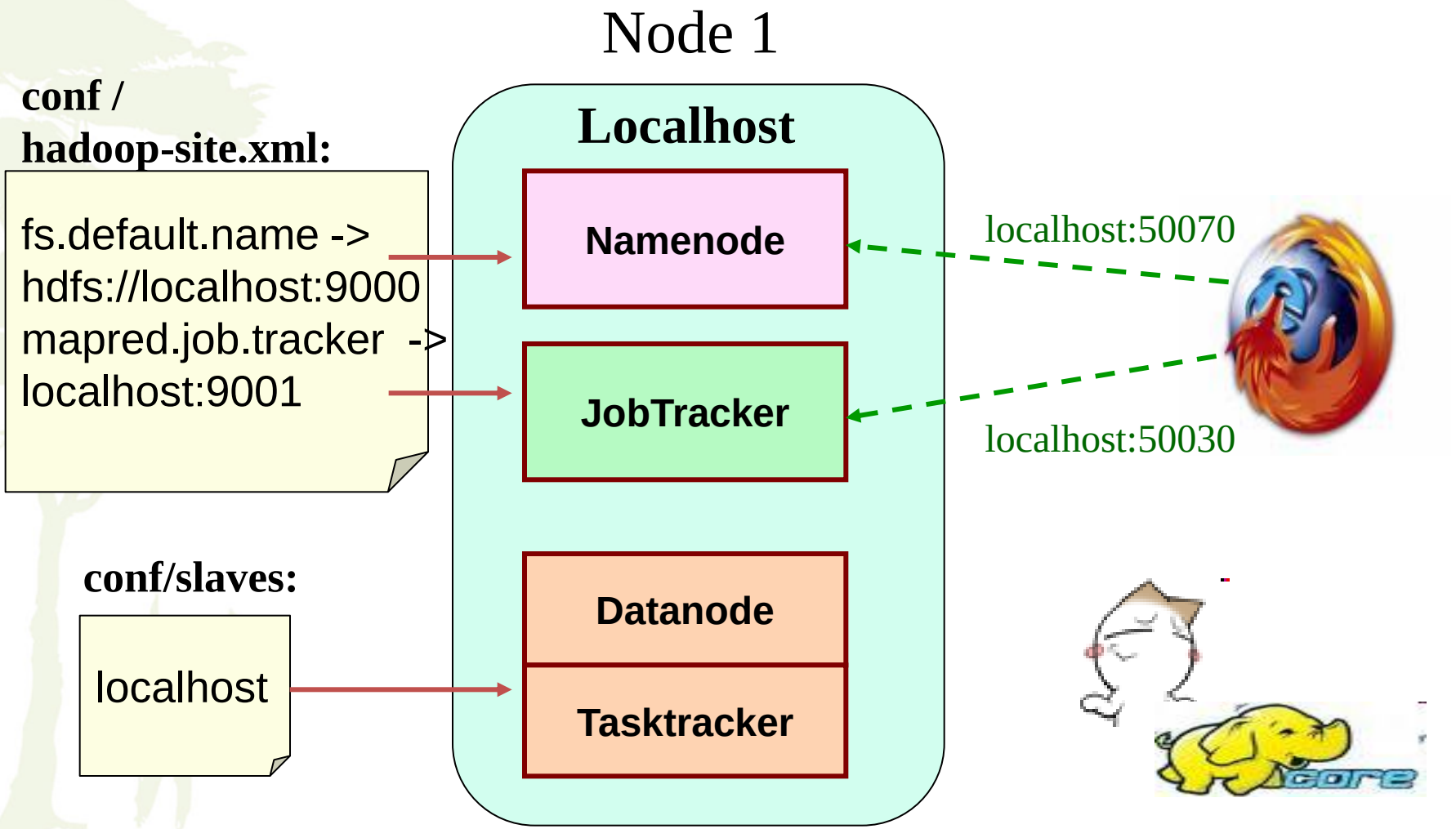
Hadoop 叢集安裝設定解析

楊順發
shunfa@nchc.narl.org.tw
自由軟體實驗室
國家高速網路與計算中心

Hadoop 單機設定與啟動

- **step 1. 設定登入免密碼**
- **step 2. 安裝java**
- **step 3. 下載安裝Hadoop**
- **step 4.1 設定 hadoop-env.sh**
  - export JAVA_HOME=/usr/lib/jvm/java-6-sun
- **step 4.2 設定 hadoop-site.xml**
  - 設定Namenode-> hdfs://localhost:9000
  - 設定Jobtracker -> localhost:9001
- **step 5.1 格式化HDFS**
  - bin/hadoop namenode -format
- **step 5.2 啟動Hadoop**
  - bin/start-all.sh
- **step 6. 完成！檢查運作狀態**
  - Job admin http://localhost:50030/
  - HDFS http://localhost:50070/

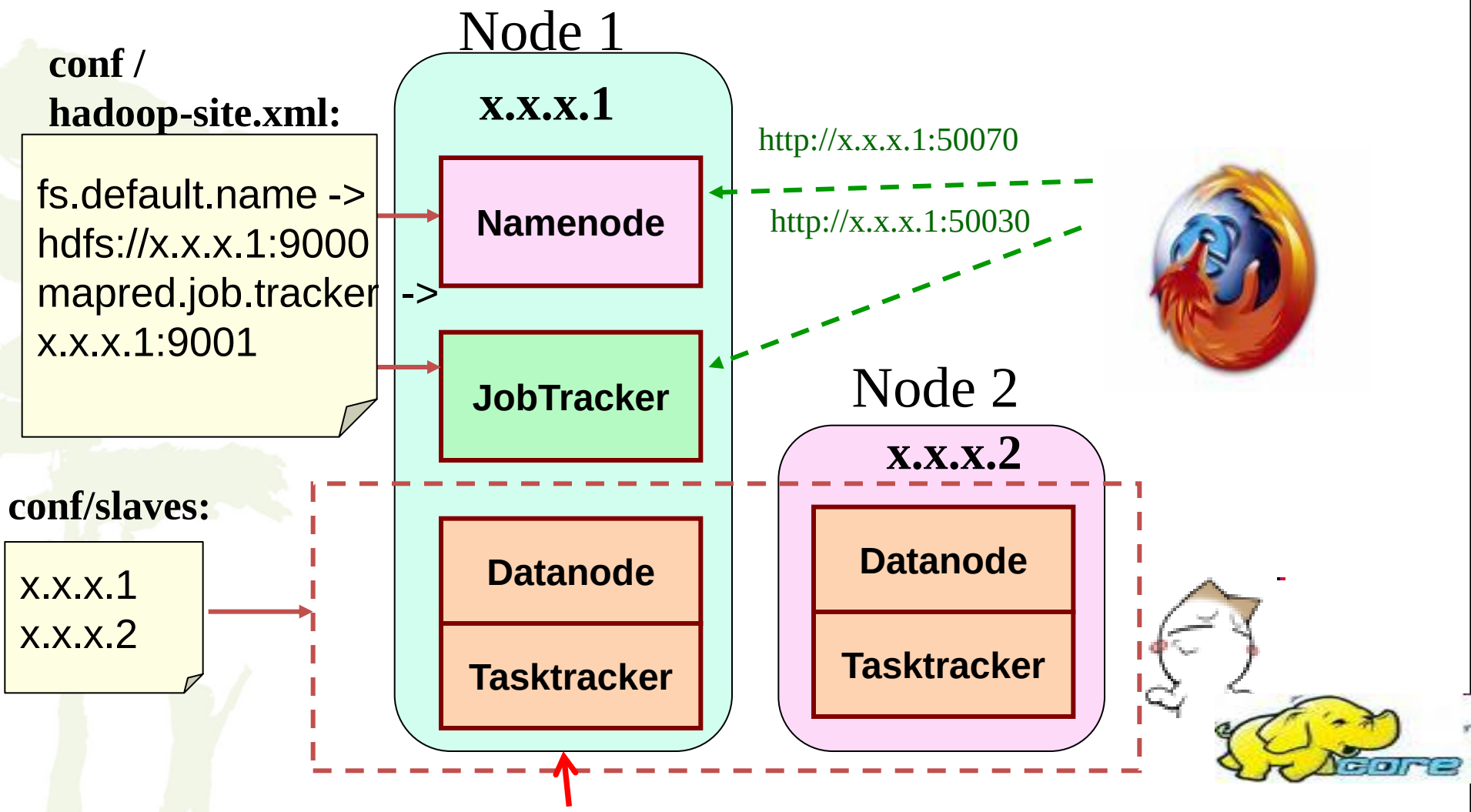
Hadoop 單機環境示意圖



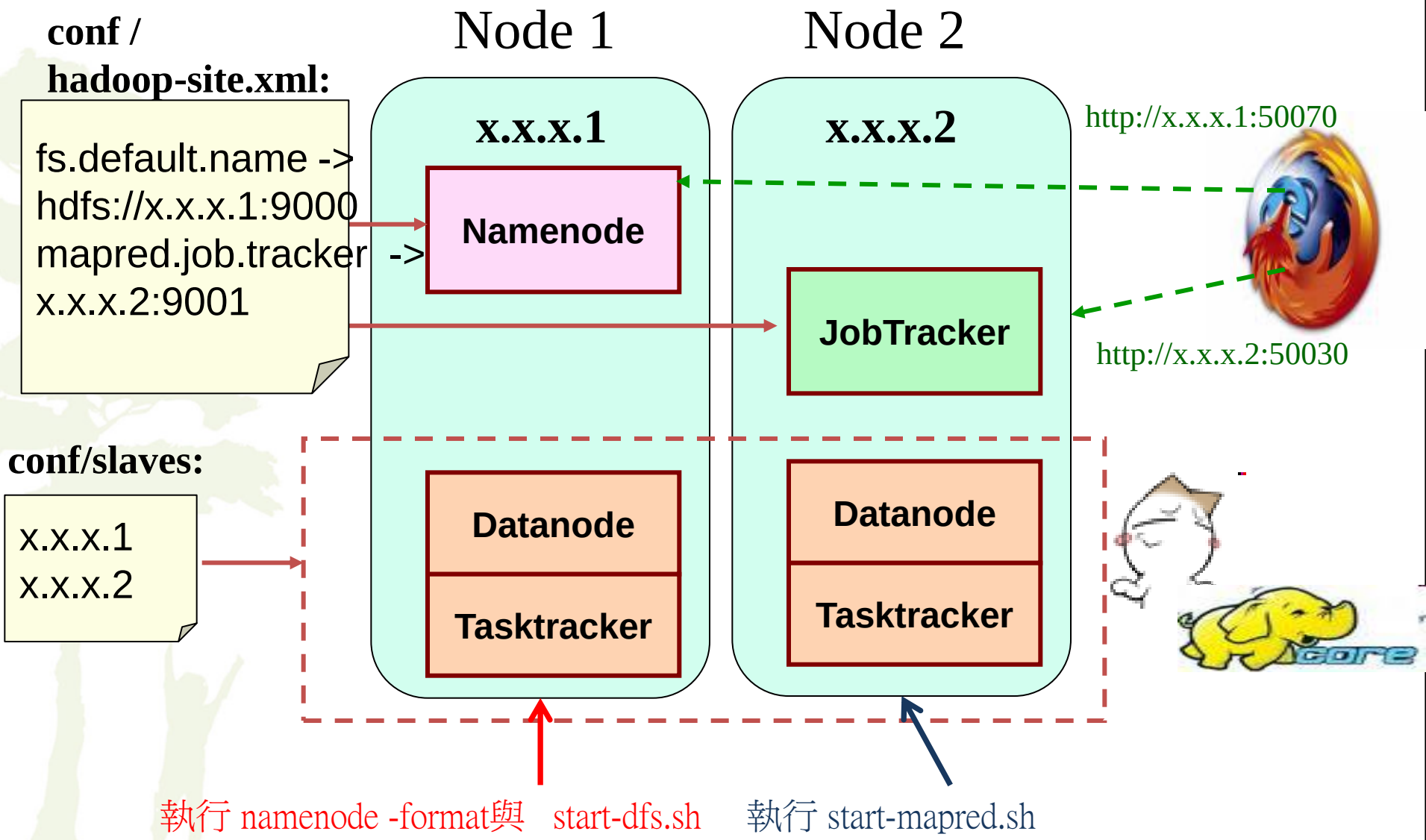
Hadoop 叢集設定與啟動

- **step 1. 設定登入免密碼**
- **step 2. 安裝java**
- **step 3. 下載安裝Hadoop**
- **step 4.1 設定 hadoop-env.sh**
  - export JAVA_HOME=/usr/lib/jvm/java-6-sun
- **step 4.2 設定 hadoop-site.xml**
  - 設定Namenode-> hdfs://x.x.x.1:9000
  - 設定Jobtracker -> x.x.x.2:9001
- **step 4.3 設定slaves 檔**
- **step 4.4 將叢集內的電腦Hadoop都做一樣的配置**
- **step 5.1 格式化HDFS**
  - bin/hadoop namenode -format
- **step 5.2 啟動Hadoop**
  - nodeN執行： bin/start-dfs.sh ； nodeJ執行： bin/start-mapred.sh
- **step 6. 完成！檢查運作狀態**
  - Job admin http://x.x.x.2:50030/ HDFS http://x.x.x.1:50070/

情況一


Node 1
x.x.x.1
conf / hadoop-site.xml:
fs.default.name -> hdfs://x.x.x.1:9000
mapred.job.tracker -> x.x.x.1:9001
Namenode
JobTracker
Datanode
Tasktracker
http://x.x.x.1:50070
http://x.x.x.1:50030
Node 2
x.x.x.2
Datanode
Tasktracker
conf/slaves:
x.x.x.1
x.x.x.2
執行 namenode －format 與 start-all.sh

情況二


conf / hadoop-site.xml:
fs.default.name -> hdfs://x.x.x.1:9000
mapred.job.tracker -> x.x.x.2:9001
Node 1
Node 2
x.x.x.1
x.x.x.2
Namenode
JobTracker
Datanode
Tasktracker
Datanode
Tasktracker
http://x.x.x.1:50070
http://x.x.x.2:50030
conf/slaves:
x.x.x.1
x.x.x.2
執行 namenode -format與 start-dfs.sh
執行 start-mapred.sh

情況三

conf / hadoop.site.xml:

fs-default.name -> hdfs://x.x.x.1:9000
mapred.job.tracker -> x.x.x.1:9001

Node 1

x.x.x.1

Namenode

JobTracker

http://x.x.x.1:50070

http://x.x.x.1:50030

conf/slaves:

x.x.x.2
…..
x.x.x.n

Node 2

x.x.x.2

Datanode

Tasktracker

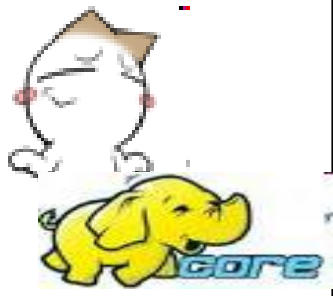
…

Node N

x.x.x.n

Datanode

Tasktracker



情況四

conf /
hadoop-site.xml:

mapred.job.tracker->
x.x.x.2:9001
fs.default.name ->
hdfs://x.x.x.1:9000

JobTracker

Namenode

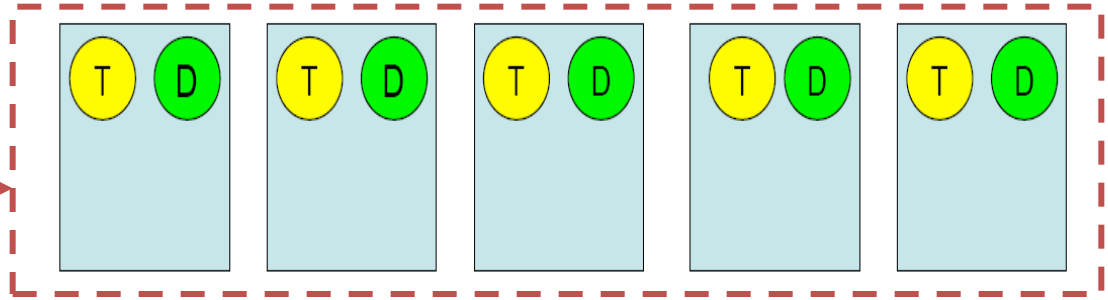
HTTP Monitoring UI

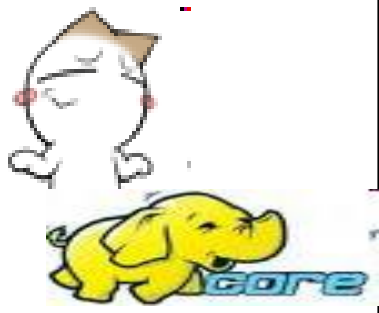
http://x.x.x.2:50030

http://x.x.x.1:50070

conf/slaves:

x.x.x.3
…….
x.x.x.n





參考：線上文件位址

- **Hadoop 叢集安裝**
  - http://trac.nchc.org.tw/cloud/wiki/Hadoop_Lab7

- **Hadoop 叢集進階用法**
  - http://trac.nchc.org.tw/cloud/wiki/Hadoop_Lab8

總結

楊順發
shunfa@nchc.narl.org.tw
自由軟體實驗室
國家高速網路與計算中心

Big Data Programming using Map/Reduce-HDFS in Hadoop

- **Big Data**
- **Hadoop**
  - MapReduce
  - HDFS, 副本策略, Heartbeat…
  - Namenode, DataNode
  - Jobtracker, Tasktracker
  - Programming
  - Cluster

Resources

- **NCHC Cloud Trac**
  - http://trac.nchc.org.tw/cloud
- **Hadoop 實驗叢集**
  - http://hadoop.nchc.org.tw/
- **台灣 Hadoop 技術討論區**
  - http://forum.hadoop.tw/
- **臉書粉絲團**
  - https://www.facebook.com/groups/hadoop.tw

Ref.

- **Apache™ Hadoop®!**
  - http://hadoop.apache.org/
- **hdfs-federation-hadoop-summit2011**
  - http://www.slideshare.net/huguk/hdfs-federation-hadoop-summit2011
- **Hadoop: The Definitive Guide**
  - http://www.amazon.com/Hadoop-Definitive-Guide-Tom-White/dp/0596521979
- **Hadoop in Action**
  - http://manning.com/lam/