I. HDFS

* Hadoop Distributed File System
* Stored in separate blocks i.e. named *blk\_1,* etc
  + Example: storing 150MB will lead to 3 separate blocks named *blk\_1 to blk\_3* containing 64MB, 64MB, 22MB respectively .
* Data nodes *(DN)*
* Name node *(NN)*

II. HDFS

* Problems:
  + Network failure
  + Disk failure on DN
  + Disk failure on NN
* Not problems:
  + Not all DN used
  + Block sizes differ

III. Data Redundancy

* To solve this problems, hadoop creates 3 copies of data to different data nodes (quite random)
* Name node is intelligent to determine these 3 copies, at pag nasirang isa irerecreate niya ulit para maging tatlo

IV. Data Redundancy

* What happens if NN fails?
  + May be inaccessible for a time (i.e. network failure)
  + Lost forever (harddisk failure)

V. Name node High availability

* May copy rin yung NN na nasa network. Or tinatawag na “Standby”

VI. HDFS demo

* All commands is prefixed with ***“hadoop fs”***
* ***hadoop fs -ls*** (lists all available commands)
* ***hadoop fs -put <filename i.e. purchases.txt>*** (uploads file to HDFS)
* ***hadoop fs -<linux command>*** (commands almost identical to linux commands, i.e. mv, mkdir, rm, etc)
* ***hadoop fs -put purchases.txt <group i.e. myinput>*** (for cleaner)

VII. MapReduce

* instead of processing file sequentially, it process data by parallel

VIII. Real word Example

IX. Hash tables

* With key -> value

X. Problems:

* i.e. Million of sales to process:
  + Long processing time
  + May run out of memory

XI. Distributed Work

Illustration: May sales per stores

* Mappers (i.e. 3 in parallel): icoconsolidate nila sa index cards yung sales per store na nakaassign sa kanila, tapos ipipile nila ung same stores sa isang group
* Reducers: taga “reduce” nung nai-pile na ng mappers. I.e. itotal nila lahat ng nakapile. Algorithm is alphabetical order (sorted)

XII. Summary of MapReduce

* Key value ang store.

XIII. Sort final results

XIV. Sort final results

* Possible by using only one reducer or adding an extra step

XV. Multiple reducers

* How is the task divided?

XVI. Multiple reducers

* We don’t know how the task is divided. Pwedeng ung isang reducer walang makuha at all

XVII. Daemons of MapReduce

* Daemons:
  + Task tracers
  + Job Tracker

XVII. Running a job

* Originally written in Java
* But we can use Python (thanks to Hadoop Streaming feature)
* These two should be present in your directory
* mapper.py
* reducer.py
* ***hadoop jar <hadoop file library?>***
* ***jar -mapper mapper.py -reducer reducer.py -file mapper.py -file reducer.py -input <yung name nung i.e. group mo, i.e. myinput> -output <output name/directory, i.e. joboutput>***
* ***hadoop -fs -ls joboutput***
* ***hadoop -fs -cat joboutput/part-00000 | less***
* ***hadoop -fs -get joboutput/part-00000 <my local file in system>*** (copy to local disk)

XVIII. Simplifying things

* They added alias (cloudera) sa VM na idodownload. Para shortcut:
* ***hs mapper.py reducer.py myinput joboutput***
* Job will not begin pag may existing na na directory i.e. *joboutput,* para iwas overwrite

XIX. A different Application

* Log processing (i.e. how many times a page was hit)

XX. Other problems

* How are you going to use and write MapReduce in your own problem?

XXI. Virtual Machine

* Cloudera provided VM (pseudo mode muna, single machine lang. Tas may answers and solutions dun)
* Read instruction notes (required subscription amps)

XXII. Conclusion