HDFS – CONT’D

BASIC FILESYSTEM OPERATIONS

- **hadoop fs -help** – list all the filesystem command
- Make directory

  hadoop fs –mkdir /user/name/testdir
- Copy a file from the local filesystem to HDFS

  hadoop fs –copyFromLocal example.txt /user/name/.

  hadoop fs –put example.txt /user/name/.
- Copy a file from HDFS to local

  hadoop fs –copyToLocal /user/name/example.txt .

  hadoop fs –get /user/name/example.txt .
- List files

  hadoop fs –ls /user/name/

HDFS FILE LISTING

- Create a directory and list its content

```
% hadoop fs -mkdir books
% hadoop fs -ls .
```

```
Found 2 items

drwxr-xr-x –  name  supergroup 0    2009-04-02 22:41 /user/name/books
-rw-r--r--    1  name supergroup  118  2009-04-02 22:29 /user/name/example.txt
```

Replication factor

Replication factor determines how many times the file is replicated

- Empty for directories since the concept of replication does not apply
- Directories are treated as metadata and stored by the namenode, not the datanodes.

4

OTHER HADOOP SUPPORTED FILE SYSTEMS

BEYOND HDFS
HADOOP SUPPORTED FILE SYSTEMS

- Hadoop has an abstract notion of filesystem, of which HDFS is just one implementation

Filesystem	Java Implementation	Description
HDFS	hdfs.DistributedFileSystem	Hadoop's distributed filesystem. HDFS is designed to work efficiently in conjunction with MapReduce.
Local *file*	fs.LocalFileSystem	A filesystem for a locally connected disk with client-side checksums. Use RawLocalFileSystem for a local filesystem with no checksums.
WebHDFS	hdfs.web.WebHdfsFileSystem	A filesystem providing authenticated read/write access to HDFS over HTTP.

HADOOP SUPPORTED FILE SYSTEMS

Filesystem	Java Implementation	Description
FTP	hdfs.ftp.FTPFileSystem	A filesystem backed by a FTP server
S3	fs.s3a.S3AFileSystem	A filesystem backed by Amazon S3
Azure	fs.azure.NativeAzureFileSystem	A filesystem backed by Microsoft Azure
Swift	fs.swift.snative.SwiftNativeFileSystem	A filesystem backed by OpenStack Swift.

REFERENCES

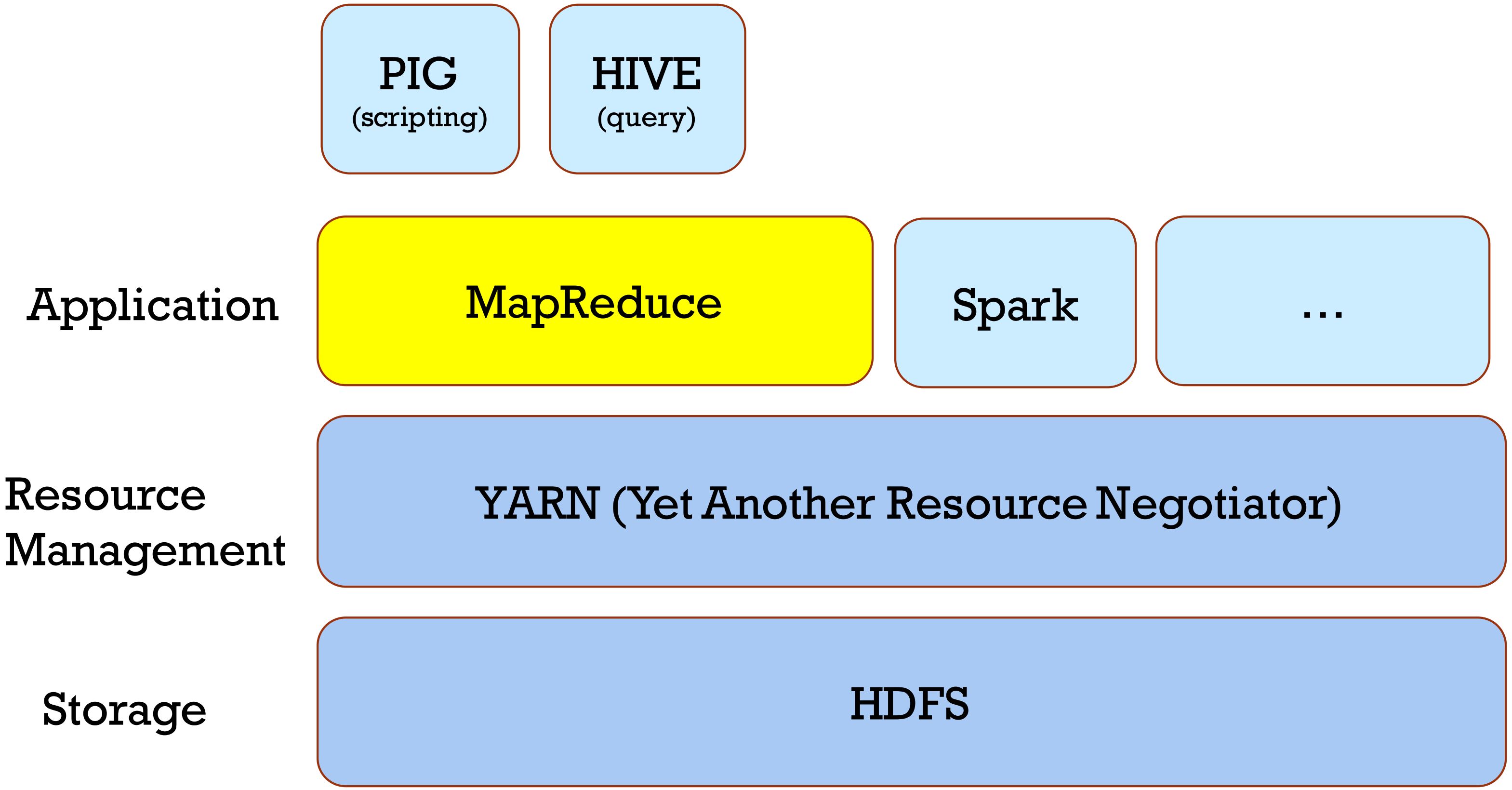
- Hadoop The Definitive Guide 4th Edition, Tom White.

HDFS COMMON INTERVIEW QUESTIONS

- Explain the HDFS Architecture and list the various HDFS daemons in HDFS cluster?
- What is a NameNode in Hadoop?
- What is a DataNode?
- Is Namenode machine same as DataNode machine as in terms of hardware?
- What is the difference between traditional RDBMS and Hadoop?
- What is Secondary NameNode? Is it a substitute or back up node for the NameNode?
- What is checkpointing in Hadoop?
    - Combining Edit Logs and FsImage on the Secondary NameNode

HADOOP MAPREDUCE

HADOOP SOFTWARE ARCHITECTURE



PIG (scripting)

HIVE (query)

Application

MapReduce

Spark

…

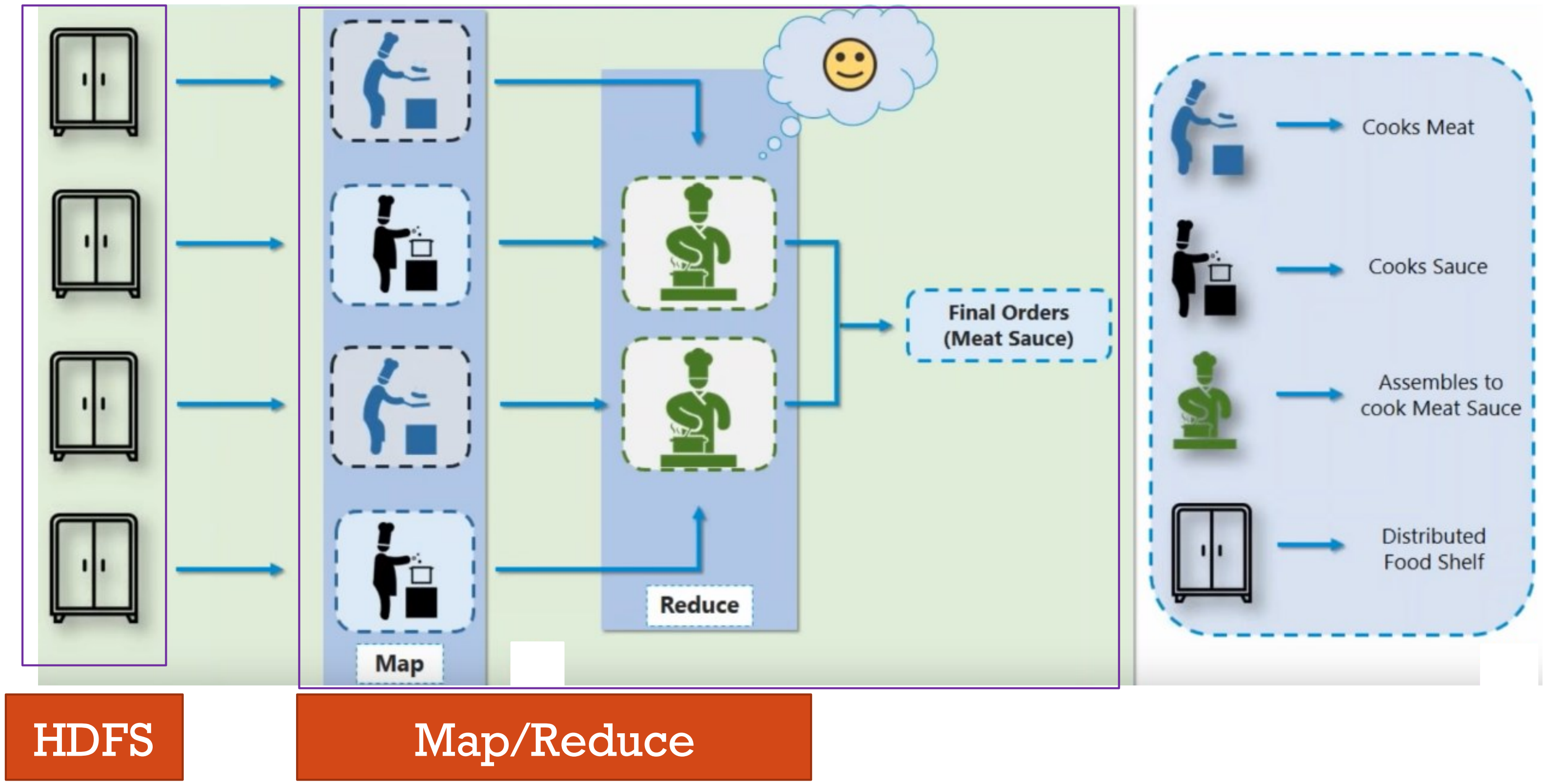
Resource Management

YARN (Yet Another Resource Negotiator)

Storage

HDFS

DO YOU REMEMBER?



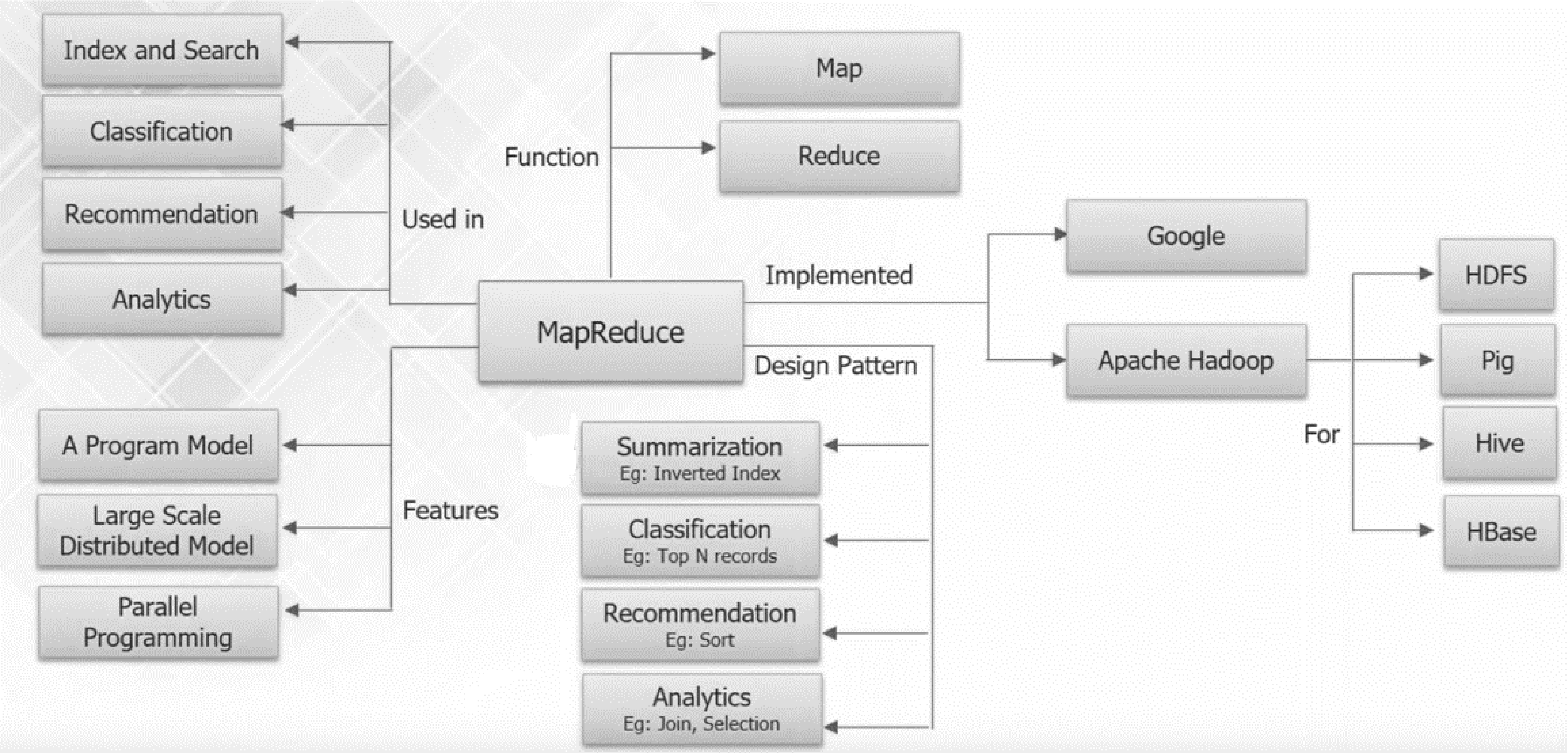
HDFS

Map/Reduce

MAPREDUCE

- A programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a ***Hadoop*** cluster.
- It abstracts the problem from disk reads and writes, transforming it into a computation over sets for ***keys and values***
- It works as a ***batch query processor*** that can run an ad hoc query against your whole data set and get result in a reasonable time

MAPREDUCE - SIMPLIFIED



WORD COUNT EXAMPLE

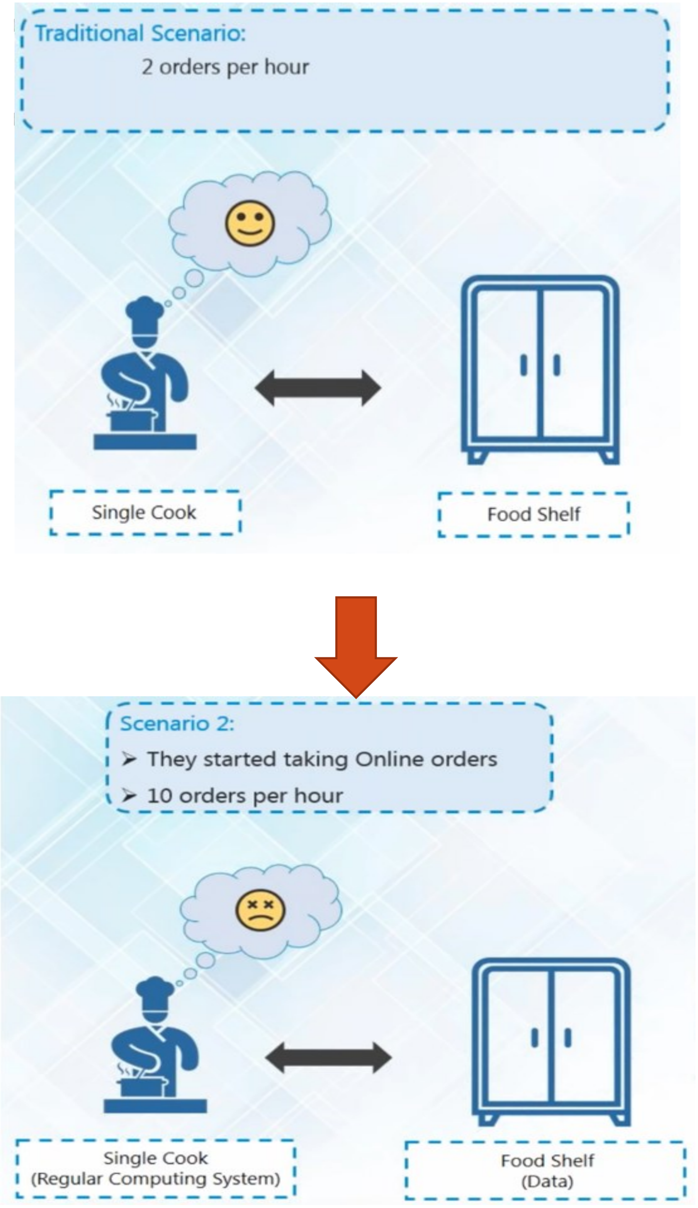
I can not do everything, but still I can do something; and because I cannot do everything, I will not refuse to do something I can do

Word	Count
And	1
Because	1
But	1
Can	4
Do	4
Everything	2
I	5
Not	3
Refuse	1
Something	2
Still	1
To	1
Will	1

WORDCOUNT PROGRAM –

```
Define WordCount as Multiset;
 For Each Document in DocumentSet {
    T = tokenize(document);
    For Each Token in T {
    WordCount[token]++;
   }
 }
Display(WordCount);
```

Program Does NOT Scale for Large Number of Documents



WORDCOUNT PROGRAM – II

- A two-phased program
  - Distribute the work over several machines
  - Combine the outcome from each machine into the final word count
- Phase I – Document Processing
  - Each machine will process a fraction of the document set
- Phase II – Count Aggregation
  - Partial word counts from individual machines are combined into the final word count

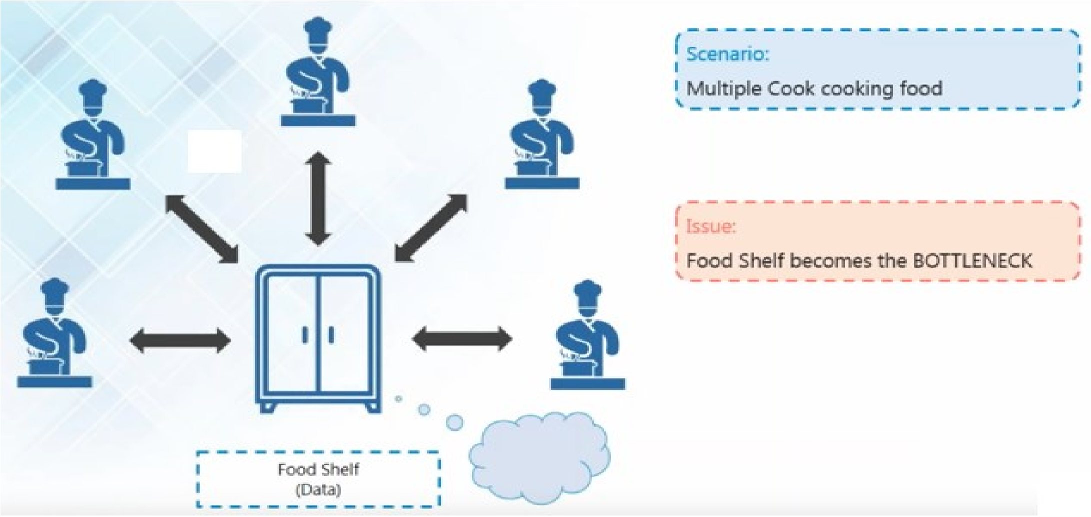
WORDCOUNT PROGRAM – II

Phase I

```
Define WordCount as Multiset;

For Each Document in DocumentSubset {
    T = tokenize(document);
    For Each Token in T {
        WordCount[token]++;

} }

SendToSecondPhase(wordCount);
```

Phase II

```
Define TotalWordCount as Multiset;

for each WordCount Received From
firstPhase {
    MultisetAdd (TotalWordCount,
    WordCount);

}
```



Scenario:
Multiple Cook cooking food

Issue:
Food Shelf becomes the BOTTLENECK

WORDCOUNT PROGRAM II – LIMITATIONS

- The program does not take into consideration the location of the documents
  - Storage server can become a bottleneck, if enough bandwidth is not available
- Storing WordCount and TotalWordCount in the memory is a flaw
  - When processing large document sets, the number of unique words can exceed the RAM capacity
- In Phase II, the aggregation machine becomes the bottleneck

WORDCOUNT PROGRAM
- HADOOP SOLUTION -

- Executing both phases in a distributed fashion on a cluster of machines that can run independently

- To achieve this, functionalities must be added
  - Store data over a cluster of processing machines
  - Partition intermediate data across multiple machines
  - Shuffle the partitions to the appropriate machines

GCP DEMO

- Redeem your coupon as shown in the Tutorials
- Install gsutil https://cloud.google.com/storage/docs/gsutil_install
- Navigate to your GCP Account at https://console.cloud.google.com/
- Create Your first cluster from Dataproc.
- Enjoy HDFS by connecting via SSH or Web Console to your master node