Hadoop on Beacon:
An Introduction

Junqi Yin and Pragnesh Patel

NICS Seminar, Feb. 20, 2015

Outline

- Intro to Hadoop
- Hadoop architecture on Beacon
- WordCount example – “Hello World” in Hadoop
- HiBench sort example – Intel Hadoop benchmark
- Hive example – Data warehouse based on Hadoop

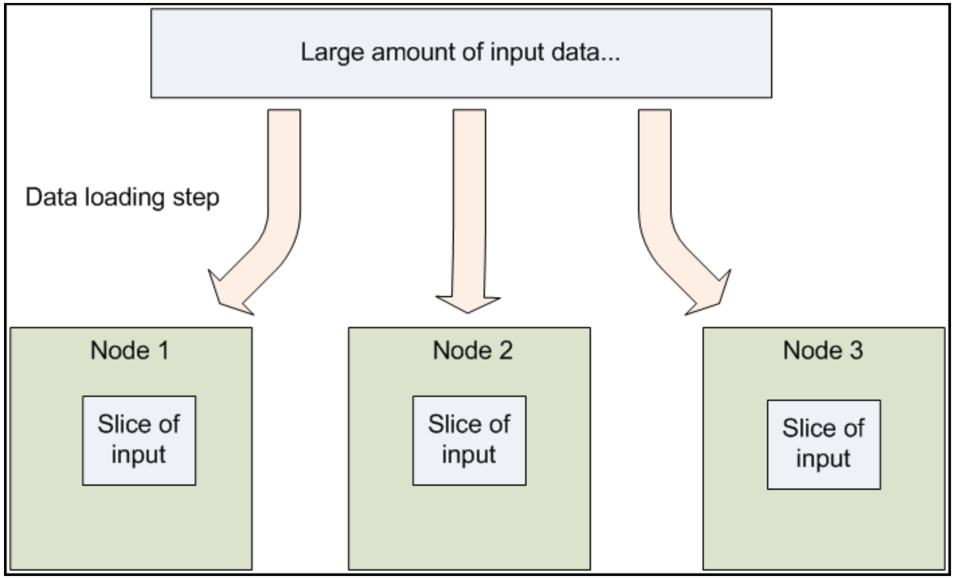
Intro to Hadoop

- Hadoop, a Apacha Software Foundation project, is designed to efficiently process large volumes of information by connecting many commodity computers together to work in parallel.
- Hadoop mainly offers two things:

1. HDFS (Hadoop Distributed File System)
2. MapReduce framework

Intro to Hadoop

- HDFS is structured similarly to a regular Unix filesystem except that data storage is distributed across several machines

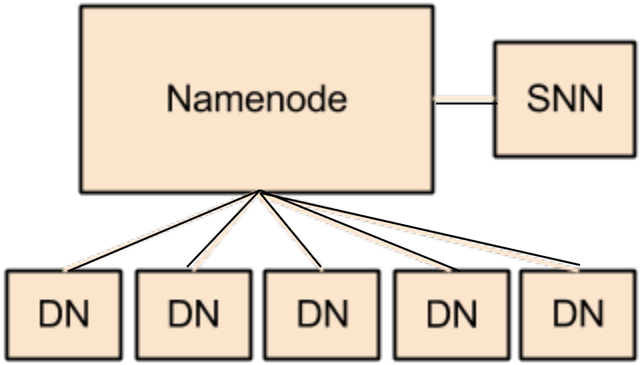


- HDFS contains:

Datanode – where data actually stores

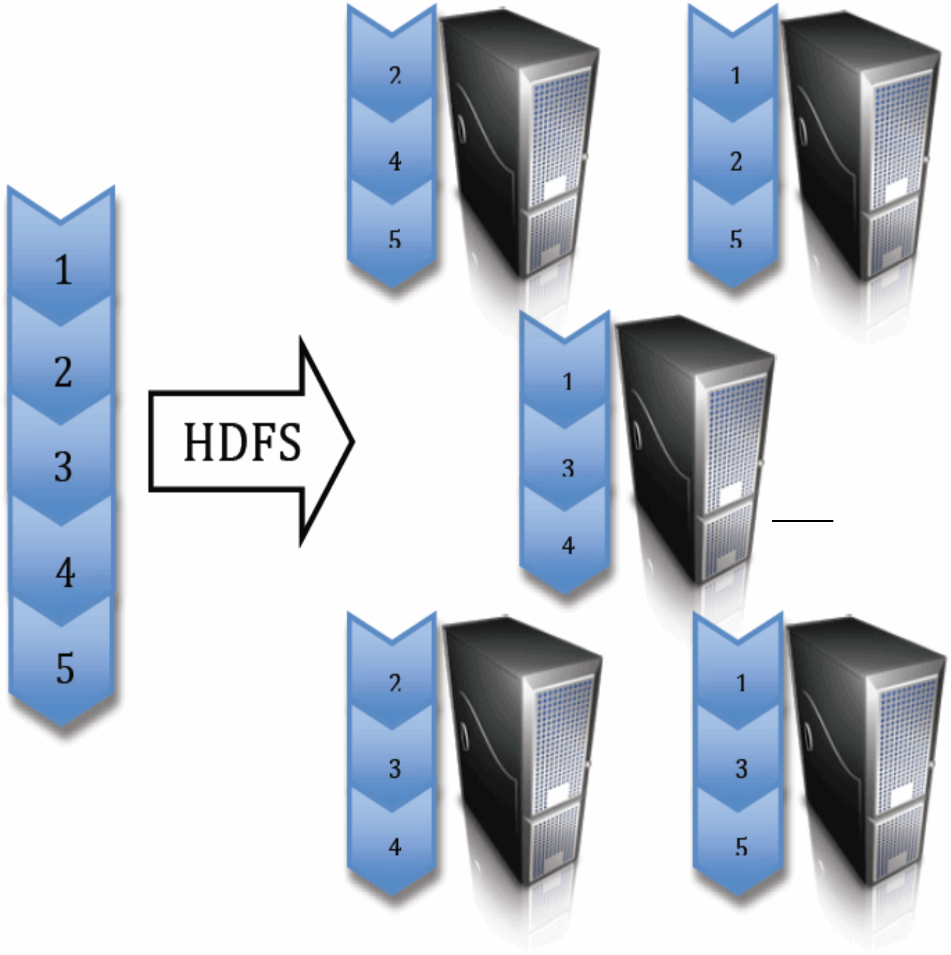
Namenode – controls meta data

Secondary Namenode – keeps edit logs, filesystem image, etc



Intro to Hadoop

- A given file is broken down into blocks (default=64MB), then blocks are replicated across cluster (default=3)



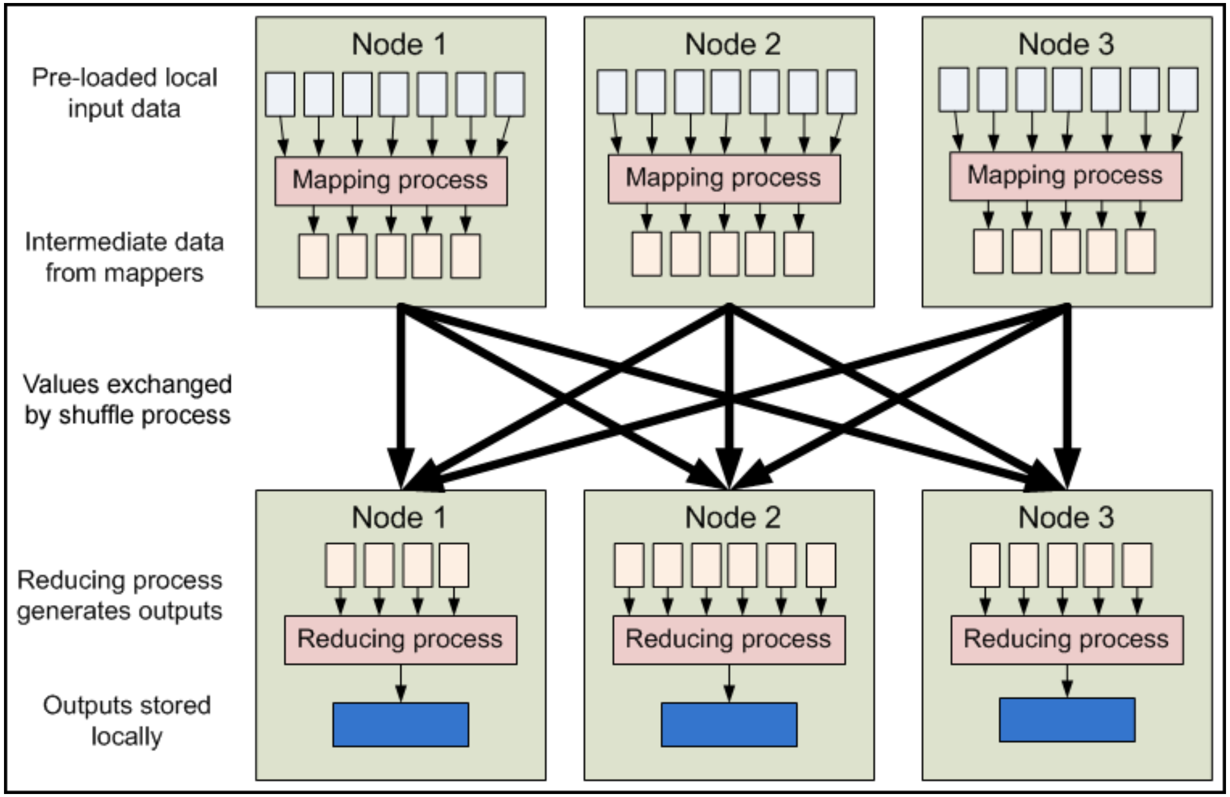
- Optimized for

1. Throughput
2. Scalability
3. Fault tolerant

Intro to Hadoop

- **MapReduce framework:**

1. **APIs for writing MapReduce programs**
2. **Services for managing the execution of these programs**

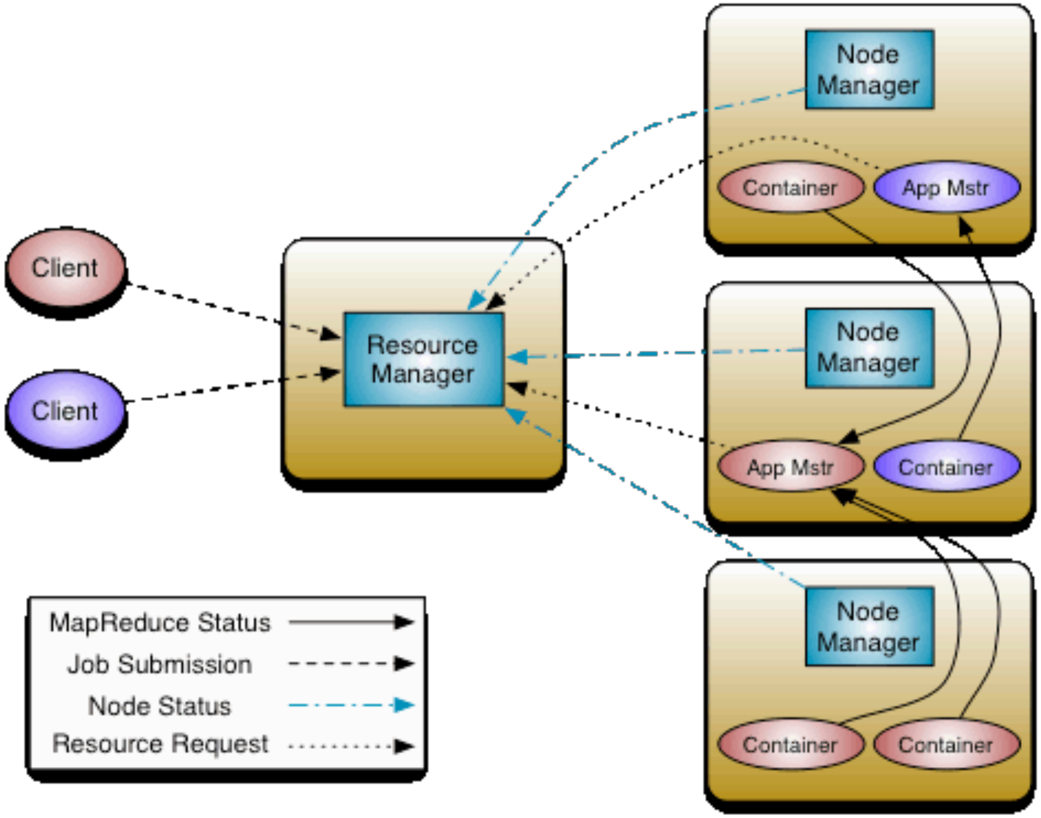


- **Mapper perform a transformation**
- **Reducer perform an aggregation**

Intro to Hadoop

- **MapReduce 2.0 (YARN):**

1. **Resource manager includes job scheduler and application manager**



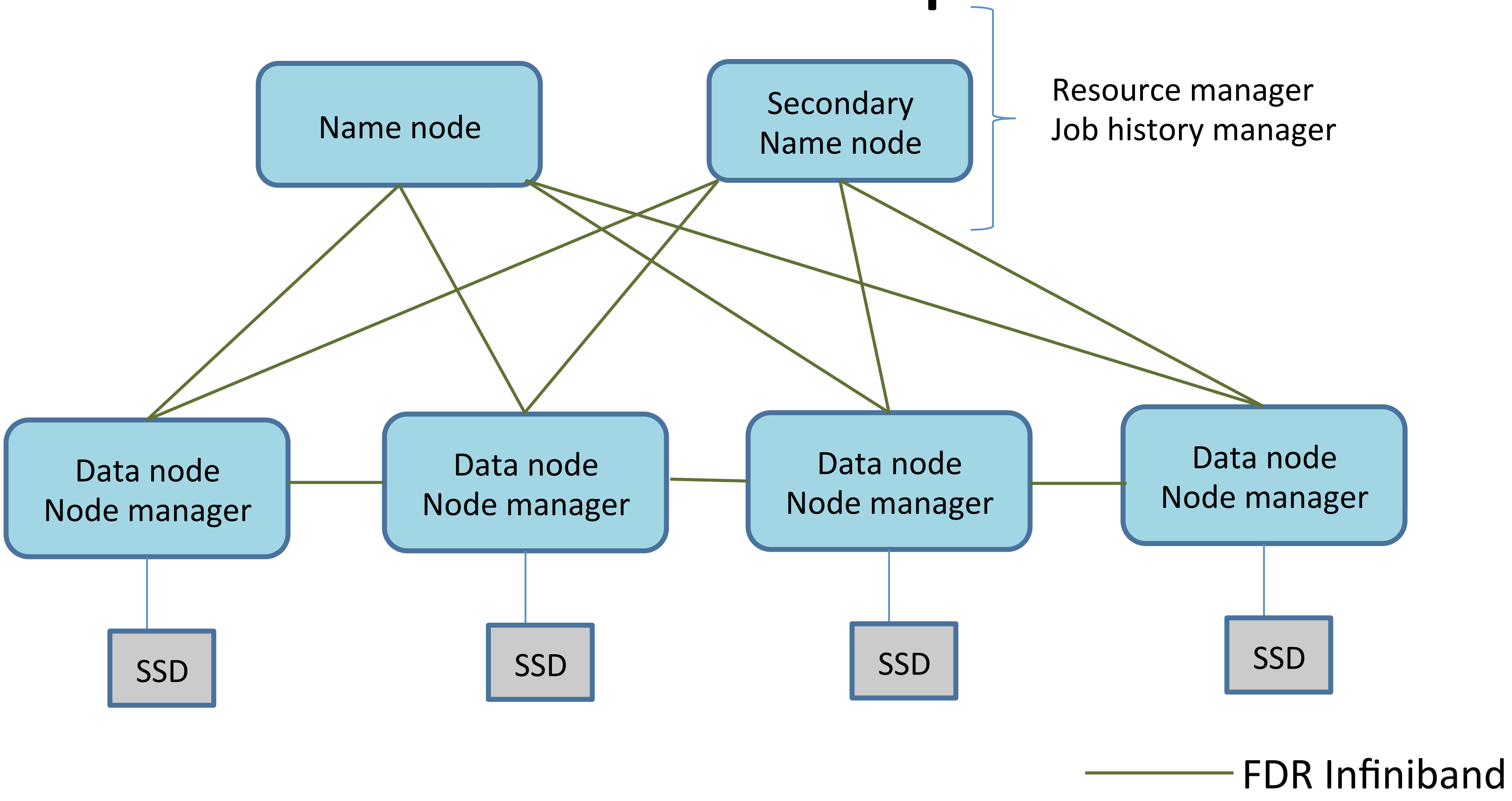
2. NodeManager is the per-machine agent who is responsible for containers and reports to ResourceManager

Intro to Hadoop

- Why Hadoop

1. Cost effective: tie smaller and more reasonably priced machines together into a single compute cluster
2. Flat scalability: orders of magnitude of growth can be managed with little re-work required
3. Reliability: fault tolerance in HDFS and MapReduce jobs

Beacon Hadoop Architecture
6 nodes example



Beacon Hadoop Configuration

- $HADOOP_CONF_DIR: core-site.xml, hdfs-site.xml, yarn-site.xml, mapred-site.xml
- By default, generated from the template files at $HADOOP_HOME/etc/hadoop/template/optimized
- User can customize configuration by

```
export HADOOP_CONF_DIR=/path/to/your/configuration
```

after loading the Hadoop module

Beacon Hadoop Configuration

- **Some important parameters:**

core-site.xml	Value	Notes
fs.defaultFS	hdfs://beaconxxx-ib0:8020	*Other than hdfs, e.g. file:/// path/to/ramdisk or lustre*
io.file.buffer.size	131072	Size of read/write buffer

hdfs-site.xml	Value	Notes
dfs.namenode.name.dir	file:///tmp/xxx/hadoop/hdfs/nn	Path on the local filesystem to NameNode directory.
dfs.datanode.data.dir	file:///tmp/xxx/hadoop/hdfs/dn	Path on the local filesystem to DataNode directory.
dfs.blocksize	134217728	HDFS blocksize of 128MB for large file-systems.
dfs.namenode.handler.count	100	More NameNode server threads to handle large number of DataNodes.
dfs.replication	1	Number of duplicates

Beacon Hadoop Configuration

- **Some important parameters:**

yarn-site.xml	Value	Notes
yarn.scheduler.minimum-allocation-mb	2048	Minimum limit of memory to allocate to each container
yarn.scheduler.maximum-allocation-mb	8196	The maximum allocation for every container request at the RM, in MBs.
yarn.nodemanager.local-dirs	file:///tmp/xxx/local	paths on the local filesystem where intermediate data is written
yarn.nodemanager.log-dirs	file:///tmp/xxx/logs	paths on the local filesystem where logs are written.
yarn.nodemanager.resource.memory-mb	245760	Amount of physical memory, in MB, that can be allocated for containers.
yarn.nodemanager.resource.cpu-vcores	30	Number of CPU cores that can be allocated for containers

Beacon Hadoop Configuration

- **Some important parameters:**

mapred-site.xml	Value	Notes
mapreduce.framework.name	yarn	Execution framework set to Hadoop YARN
mapreduce.map.memory.mb	2048	Larger resource limit for maps
mapreduce.map.java.opts	-Xmx1638m	Larger heap-size for child jvms of maps
mapreduce.reduce.memory.mb	4096	Larger resource limit for reduces.
mapreduce.task.io.sort.mb	1066	Higher memory-limit while sorting data for efficiency.
mapreduce.job.reduce.slowstart.completedmaps	0.95	Control when reduce jobs start
mapreduce.jobs.maps	64	The default number of map tasks per job
mapreduce.jobs.reduces	64	The default number of reduce tasks per job

Beacon Hadoop Usage

```
[jqyin@beacon-login2 ~]$ module help hadoop/2.5.0

----------- Module Specific Help for 'hadoop/2.5.0' ---------------

Sets up environment to use Hadoop 2.5.0.
Usage:
module load hadoop/2.5.0
cluster_start (setup hadoop cluster with at least 3 nodes)
copy your data to HDFS
run hadoop application with your data
copy results back from HDFS
cluster_stop (shutdown hadoop cluster)
******************************************
IMPORTANT:
By default, HDFS is set up on local SSD,
the data on which will be purged once job exits
******************************************
```

WordCount example: “hello world” in Hadoop

- Source code (WordCount.java) at /lustre/medusa/jqyin/hadoop/WordCount.java
- Steps to run on Beacon:

1. qsub -I -l nodes=3   # request a interactive job
2. module load hadoop/2.5.0   #load the module
3. cluster_start   # launch hadoop services

WordCount example: “hello world” in Hadoop

- Steps to run on Beacon:

```
4. hadoop com.sun.tools.javac.Main WordCount.java
# compile the java code
5. jar cf wc.jar WordCount*.class  # create jar file
6. hadoop fs -mkdir -p /user/wordcount/input
# create input folder on hdfs
7. cat   << _EOF_ > file0
Hello World Bye World
_EOF_
```

WordCount example: “hello world” in Hadoop

- Steps to run on Beacon:

```
7. cat   << _EOF_ > file1
Hello Hadoop Goodbye Hadoop
_EOF_
8. hadoop  fs -put  file* /user/wordcount/input
# copy files to hdfs
9. hadoop jar wc.jar WordCount /user/wordcount/input /
   user/wordcount/output
# run the application
```

WordCount example: “hello world” in Hadoop

- Check result:

```
hadoop fs -cat /user/wordcount/output/part-r-*
Bye 1
Goodbye 1
Hadoop 2
Hello 2
World 2
```

WordCount example:
“hello world” in Hadoop

- **Mapper**

```
public void map(Object key, Text value, Context context ) throws
IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
     }
}
```

- **Output: file0 <Hello, 1> <World,1> <Bye,1> <World,1>**

  file1 <Hello,1> <Hadoop,1> <Goodbye, 1> <Hadoop,1>

WordCount example: “hello world” in Hadoop

- **Reducer**

```
public void reduce(Text key, Iterable<IntWritable> values,
Context context ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
        sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
}
```

- **Output:**

<Bye, 1> <Goodbye, 1> <Hadoop, 2> <Hello, 2> <World, 2>

Running Hadoop on Beacon

HiBench sort example

```
#PBS -A your-account-number
#PBS -j oe
#PBS -l nodes=6
#PBS -l walltime=1:00:00

#load the Hadoop module
module load hadoop/2.5.0

#start the Hadoop cluster with one name node,
#one secondary name node plus resource manager and job history #manager,
four data nodes plus node managers
#this command will also setup the directories on HDFS for Hadoop #and Hive
cluster_start

#Run Hadoop application: HiBench sort
$HADOOP_HOME/HiBench/HiBench.sh sort
cluster_stop
```

Running Hadoop on Beacon

Hive example

```
#PBS -A your-account-number
#PBS -j oe
#PBS -l nodes=6
#PBS -l walltime=1:00:00

#load the Hadoop module
module load hadoop/2.5.0

cluster_start

#Run Hadoop application: Hive
wget http://files.grouplens.org/
datasets/movielens/ml-100k.zip
unzip ml-100k.zip
cd ml-100k
```

```
cat << _EOF_ > hive-script.sql
CREATE TABLE u_data (
  userid INT,
  movieid INT,
  rating INT,
  unixtime STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH './u.data'
OVERWRITE INTO TABLE u_data;

SELECT COUNT(*) FROM u_data;
_EOF_

hive -f hive-script.sql

#stop hadoop cluster
cluster_stop
```

Reference

- http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
- https://developer.yahoo.com/hadoop/tutorial/
- http://hortonworks.com/wp-content/uploads/2014/02/RHEL_Big_Data_HDP-Reference_Architechure_FINAL.pdf