

**Name: Dharmesh Jain and Shrinivas S
Kulkarni**

**Title: Introduction to Hadoop and BigInsights
Programming Overview**



Agenda

- **Hadoop Overview**

- Why Hadoop?
- Hadoop distributed file system
- Map Reduce engine

- **JAQL Overview**

- Need for high level languages
- Core JAQL operators

- **BigInsights programming**

- BigInsights Apps
 - Application catalog
 - Develop your own BigInsights Applications
- Development tools
- BigSheets

Hardware improvements through the years...

- **CPU Speeds:**

- 1990 - 44 MIPS at 40 MHz
- 2000 - 3,561 MIPS at 1.2 GHz
- 2010 - 147,600 MIPS at 3.3 GHz

- **RAM Memory**

- 1990 – 640K conventional memory (256K extended memory recommended)
- 2000 – 64MB memory
- 2010 - 8-32GB (and more)

- **Disk Capacity**

- 1990 – 20MB
- 2000 - 1GB
- 2010 – 1TB

- **Disk Latency (speed of reads and writes) – not much improvement in last 7-10 years, currently around 70 – 80MB / sec**

How long it will take to read 1TB of data?

- **1TB (at 80Mb / sec):**
 - 1 disk - 3.4 hours
 - 10 disks - 20 min
 - 100 disks - 2 min
 - 1000 disks - 12 sec
- **Parallel Data Processing is the answer!**

Parallel computing is not new

- **HPC and Grid computing**

- Move data to computation- Network bandwidth becomes a bottleneck; compute nodes idle
- Good for compute intensive jobs
- Exchanging data requires synchronization– very tricky
- Scalability is programmer's responsibility
 - Will require change in job implementation

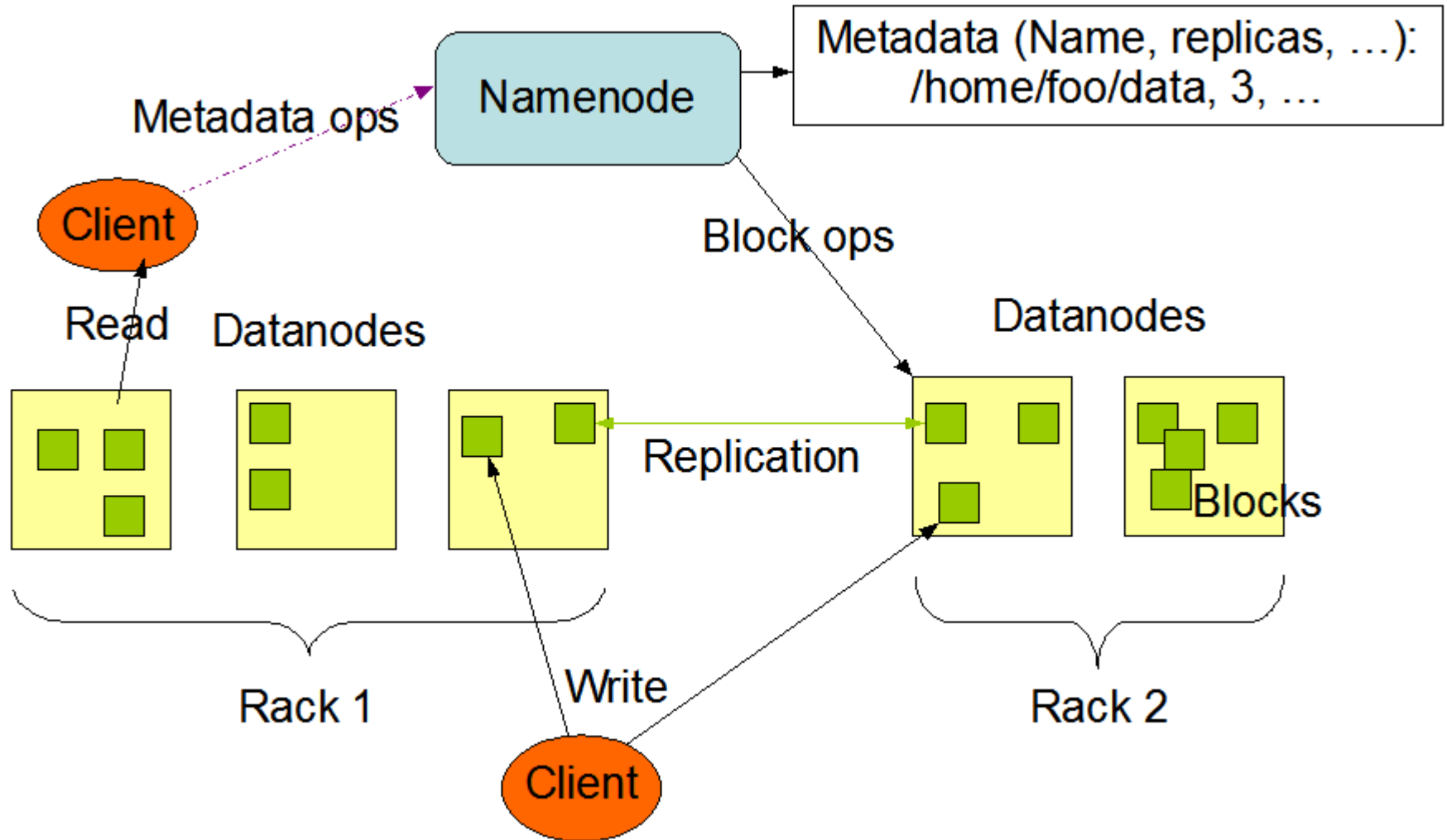
- **Hadoop approach**

- Move computation to data- conserves network bandwidth
- Shared nothing Architecture- no dependencies between tasks
- Communication between nodes in frameworks responsibility
- Designed for scalability
 - Adding increased load to a system should not cause outright failure, but a graceful decline
 - Increasing resources should support a proportional increase in load capacity
 - Without modifying the job implementation

Apache *hadoop*

- **A scalable fault-tolerant distributed system for data storage and processing (open source under the Apache license).**
- **Inspired by Google technologies**
 - MapReduce
 - Google file system
- **Originally built to address scalability problems of Nutch, an open source Web search technology**
 - Developed by Douglass Read *Cutting* (Doug cutting)
- **Core Hadoop has two main systems:**
 - **Hadoop Distributed File System:** self-healing high-bandwidth clustered storage.
 - **MapReduce:** distributed fault-tolerant resource management and scheduling coupled with a scalable data programming abstraction.

HDFS Architecture – Master/Slaves



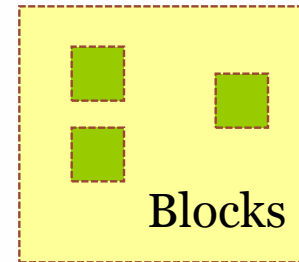
NameNode

Master node
NameNode

- **Manages the file system namespace**
 - Maintains file system tree and meta data for all files/directories in the tree
 - **Single point of failure.** Name node loss renders file system inaccessible
- **Centralizes and manages file system metadata in memory**
 - Maps blocks to DataNodes, filenames, etc
 - Metadata size limited to available RAM of NameNode.
 - Bias toward modest number of large files, not large number of small files (where metadata can grow too sizeable)
 - NameNode will crash if it runs out of RAM
- **Runs on a master node**
 - Coordinates access to DataNodes but data never goes on NameNode
 - Hadoop V1 has no built-in failover mechanism for NameNode

DataNode (Slave)

DataNode



- **Files on HDFS are chopped into blocks and stored on DataNodes**
 - Size of blocks is configurable
 - Different blocks from the same file are stored on different DataNodes if possible
- **Serves read and write requests to clients**
- **Performs block creation, deletions, and replication as instructed by NameNode**
 - Replication factor is configurable
- **One instance of DataNode per slave node is recommended in real deployment**

HDFS Data Blocks



- **Much larger than traditional file system blocks**
 - 64MB by default. Increase to 128MB for very large files.
 - If chunk of file is smaller than HDFS block size, only needed space is used
- **Trade-off: block size and MapReduce parallelism**
 - Map tasks in MapReduce normally operate on one block at a time
 - so if you have too few tasks (fewer than nodes in the cluster), your jobs will run slower than they could otherwise
 - Minimize the cost of seeks
- **Advantages of HDFS's data block approach**
 - Simplifies replication, providing fault tolerance and reliability
- **Each block replicated across 3 DataNodes (by default)**
 - 1st replica placed on same node as client
 - 2nd replica placed on different rack from 1st rack
 - 3rd replica placed on same rack as 2nd rack, but on a different node

Map Reduce 101

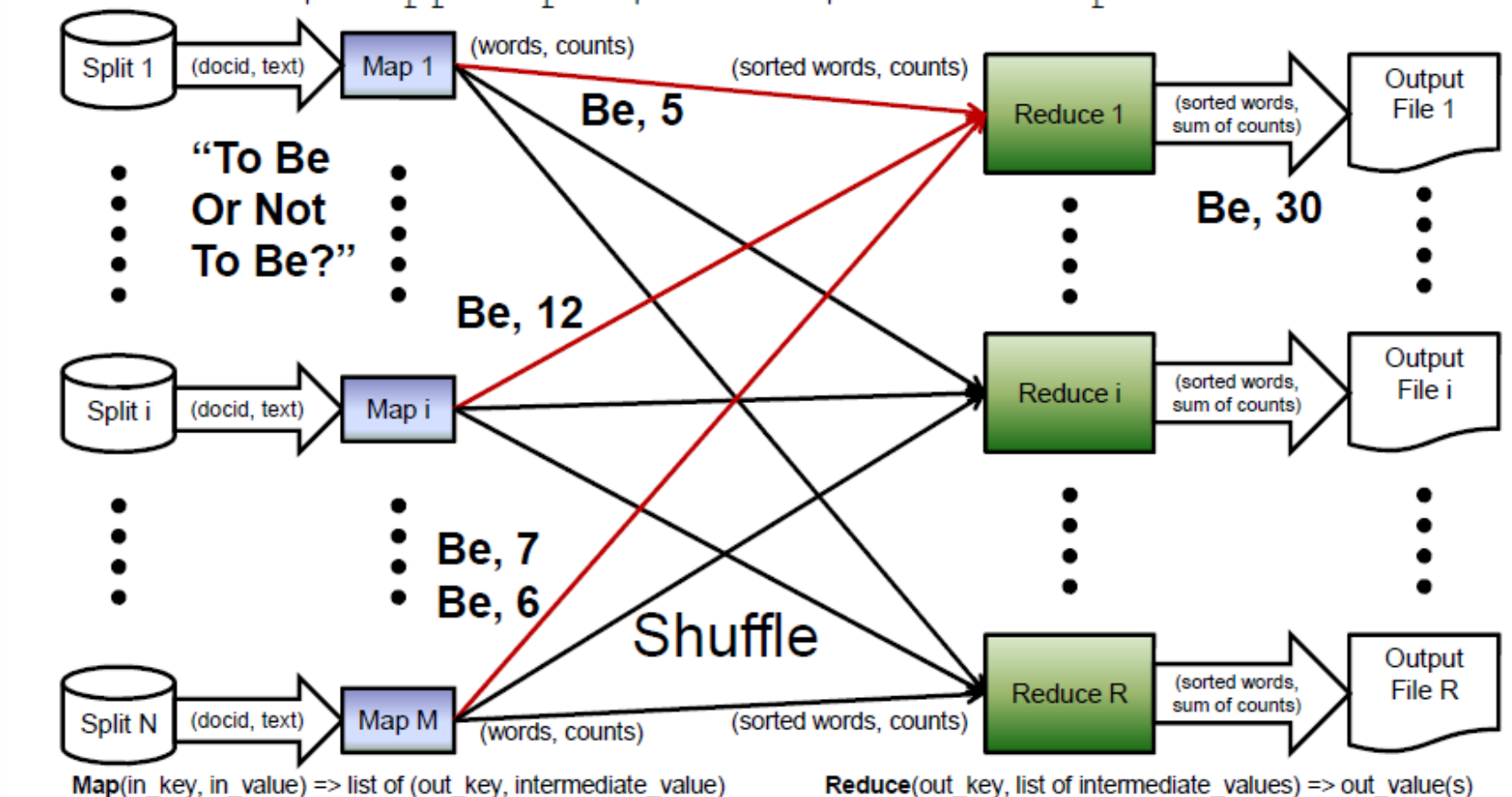
- Originated in functional programming but common in many languages
- **Example of Map function:**
 - **square** $x = x * x$
 - **map square** [1,2,3,4,5] will return [1,4,9,16,25]
- **Notice that I can process “map square” in parallel:**
 - **map square** [1,2,3] -> [1,4,9]
 - **map square** [4,5] -> [16,25]
- **Example of Reduce function:**
 - **MAX** (1, 2, 3, 10,15, 20) -> 20
 - **SUM** (1, 7, 10) -> 18
- **In Hadoop REDUCE function always takes MAP function as an input**
 - REDUCE phase is optional, for some jobs no reducing is required

Hadoop MapReduce engine

- **Framework which enables writing applications to process multi-terabyte of data in-parallel on large clusters (thousands of nodes) of commodity hardware**
- **A clean abstraction for programmers**
 - No need to deal with internals of large scale computing
 - Implement just Mapper and Reducer functions- most of the times
 - Implement in the language you comfortable with
 - Java (assembly language for Hadoop)
 - With hadoop streaming, you can run any shell utility as mapper and reducer
 - Hadoop pipes to support implementation of mapper and reducer in C++.
- **Automatic parallelization & distribution**
 - Divides the job into tasks (map and reduce task)
 - Schedules submitted jobs
 - Schedules tasks as close to data as possible
 - Monitors task progress
- **Fault-tolerance**
 - Re-execute failed or slow task instances.

Data flow in a map reduce job

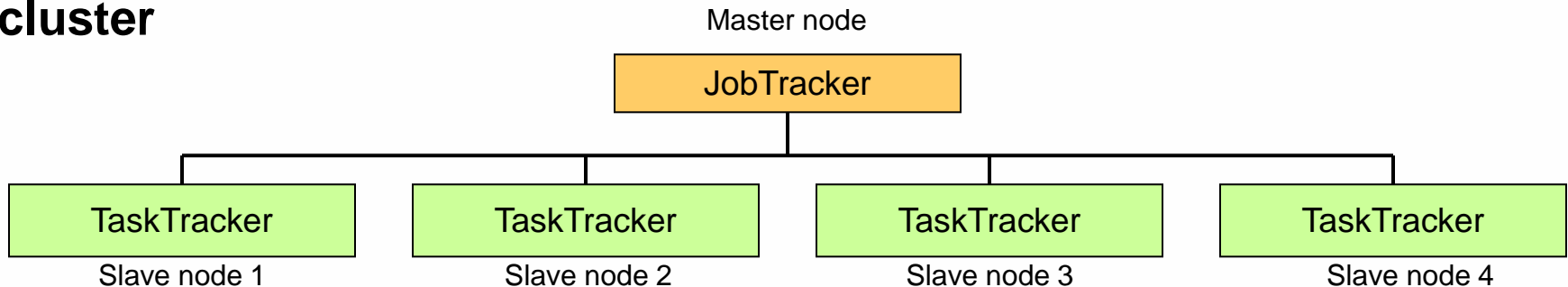
```
cat *.txt | mapper.pl | sort | reducer.pl > out.txt
```



MapReduce Architecture- master/slave

- **Single master (JobTracker) controls job execution on multiple slaves (TaskTrackers).**
- **JobTracker**
 - Accepts MapReduce jobs submitted by clients
 - Pushes *map* and *reduce* tasks out to TaskTracker nodes
 - Keeps the work as physically close to data as possible
 - Monitors tasks and TaskTracker status
- **TaskTracker**
 - Runs map and reduce tasks; Reports status to JobTracker
 - Manages storage and transmission of intermediate output

cluster



Word count Mapper

```
public static class WordCountMapper extends MapReduceBase
    implements Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value,
        OutputCollector<Text, IntWritable> output,
        Reporter reporter) throws IOException {
        String line = value.toString();
        StringTokenizer itr = new StringTokenizer(line);
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            output.collect(word, one);
        }
    }
}
```

Word count Reducer

```
public static class WordCountReducer extends MapReduceBase
    implements Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values,
        OutputCollector<Text, IntWritable> output,
        Reporter reporter) throws IOException {
        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
    }
}
```


Prepare and Submit job

```
public class WordCountJob {
    public static void main(String[] args) throws Exception{
        JobConf conf = new JobConf(WordCount.class);

        // specify input and output dirs
        FileInputFormat.addInputPath(conf, new Path("input"));
        FileOutputFormat.addOutputPath(conf, new Path("output"));
        // specify output types
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        //InputFormat and OutputFormat
        conf.setInputFormat(TextInputFormat.class);

        conf.setOutputFormat(TextOutputFormat.class);

        conf.setMapperClass(WordCountMapper.class); // specify a mapper
        conf.setReducerClass(WordCountReducer.class); // specify a reducer
        conf.setCombinerClass(WordCountReducer.class);

        conf.setNumberOfReducer(2); //Number of reducer
        JobClient.runJob(conf); // Submit the job to Job Tracker
    }
}
```

Need for higher level languages

- Although the Hadoop framework is implemented in Java, **MapReduce applications do not need to be written in Java**
- To abstract complexities of Hadoop programming model, a few application development languages have emerged that build on top of Hadoop:
 - Pig
 - Hive
 - Jaql



Jaql

Word count in JAQL

- `read(lines("/WordCount/input")) -> expand tokenizer($)`
 - > `group by w = $ into { word:w ,frequency: count($)}`
 - > `sort by [$.frequency]`
 - > `write(file("/WordCount/output/wordcount_result"));`
- Anatomy of Word count
 - `$doc = read(lines("/jaqlsession/data/file1"));`
 - `$alltokens = $doc -> expand tokenizer($);`
 - `$unsortedresult = $alltokens -> group by w = $ into { word:w , frequency: count($)};`
 - `$sortedresult = $unsortedresult -> sort by [$.frequency] ;`
 - `$sortedresult -> filter $.frequency > 5;`
 - `$stopWordList = ["the", "Hello", "Work"];`
 - `$sortedresult -> filter not ($.word in $stopWordList);`
 - `$unsortedresult -> top 3 by [$.frequency desc];`
 - `$sortedresult -> write(file("/opt/ibm/JaqlSession/sortedWordCount"));`

Core operators – TRANSFORM

- The transform operator allows you to manipulate the values in an array ("project" in DBMS vernacular)
 - An expression is applied to each element in the array
 - The result of the expression is the next element in the output array

```
jaql> recs = [ {a: 1, b: 4}, {a: 2, b: 5}, {a: -1, b: 4} ];  
jaql> recs -> transform $.a + $.b;  
[ 5, 7, 3 ]
```

```
jaql> recs -> transform { sum: $.a + $.b };  
[ { sum: 5 }, { sum: 7 }, { sum: 3 } ]
```

- The `each` clause can be applied if you don't like `$`

```
jaql> recs -> transform each rec { sum: rec.a + rec.b };  
[ { sum: 5 }, { sum: 7 }, { sum: 3 } ]
```

Other important core operators

- **EXPAND**
 - The expand operator flattens nested arrays
- **FILTER**
 - filter allows you to selectively filter out array entries
- **GROUP**
 - Performs SQL-style GROUP BY against a single input
- **JOIN**
 - The join operator joins two or more arrays
- **SORT and TOP**
 - The sort operator allows sorting of arrays
 - The top operator returns the first k rows of its input array

Why I love JAQL?

- **There are other well-known languages (Hive, Pig, etc.) Why another?**
- **Hive**
 - Good for "flat", structured data
 - Has Java UDF/UDA's
 - Familiar SQL syntax
- **Pig**
 - Better for moderately complex, nested data
 - Has Java UDF/UDA's
 - Used for simple scripts
- **JAQL: A JSON Query Language**
 - Elegantly handles deeply nested data (e.g. text analytics).
 - Has Java UDF/UDA's
 - Native Jaql functions and aggregates
 - Modules and functions allow for larger, more complex projects
 - Seamlessly integrates Jaql and SQL syntax

Developing and sharing jobs is not easy!

- **Typical Workflow**
 - Import data into the cluster
 - Analyze the imported data
 - Java MR jobs ; JAQL; Hive; Pig
 - Write back the analysis result in cluster
 - Optionally, export the results out
- **Some jobs needs to be run periodically**
 - Indexing
- **Big data jobs are long running**
 - You have to monitor them; to track progress
- **Packaging the workflow**
 - So that it is consumable

Applications Catalog (Web Console)

- Browse available applications
- Deploy published applications (administrators only)
- Launch (or schedule for launch) a deployed application
- Monitor job (application) execution status

The screenshot displays the IBM Applications Catalog Web Console. On the left, a sidebar titled 'Applications' shows a grid of application icons: HIVE, JAQL, PIG, and Boardreader. The main panel shows the details for the 'Ad hoc Jaql query' application. The 'Name' is 'Ad hoc Jaql query' and the 'Description' is 'The Ad hoc Jaql Query application runs a custom query entered in the UI to analyze data.' Below the description, the 'Execution' section shows the 'Execution Name' as 'JaqlTest' with a 'Run' button. The 'Parameters' section shows a text area containing '\$test = [1,2,3];' and '\$test;'. At the bottom, the 'Application History' table shows a single execution record for 'JaqlTest' with a status of 'Completed', 100% progress, and an elapsed time of 21 seconds.

Status	Execution Name	Progress	Start Time	Elapsed Time	Details
Completed	JaqlTest	100%	Nov 22, 2011 8:59:07 PM	21(sec)	

Running Applications

- Import & Export Data
 - Database & Files
- Web and Social
- Analyze and Query
 - Predictive Analytics
 - Text Analytics
 - SQL/Hive, Jaql, Pig, HBase

Applications

BoardReader Pig sample Crawler Database Import

Tera Gen-Sort Hive sample Word Count Database Export

Simple Jaql Application Distributed Copy Hadoop Streaming Word Count Sample Ad hoc Hive query

Name: Ad hoc Hive query

Description: You can use the ad hoc Hive query to create your own customized queries.

Execution Name: Default Execution

Hive query:

```
CREATE EXTERNAL TABLE student (
  NAME STRING,
  AGE INT,
  GPA FLOAT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION '/tmp/mystudents';
```

Status	Execution Name	Progress
✓	Default Execution	100%

Quickly drag and drop to create new Apps

Execute Applications | Chain Applications

Applications



A vertical palette of application icons for chain building. The icons include: Boardreader, Data Sample, JAQL, Data Subset, Database Export, Database Import, Distributed File Copy, Web Crawler, and Word Count. The 'Data Sample' icon is currently selected and highlighted with a purple border.



Application Name: Data Sampling Analysis

Application Description: Application transfers data from ftp to hdfs. Generates a sample of it and exports it into a database.

Workflow Diagram: A sequence of four steps connected by arrows. The first three steps are 'Distributed File Copy', 'Data Sample', and 'Database Export', each with a red 'X' in the top right corner. The fourth step is a dashed box containing a green circular arrow icon, representing a loop or continuation.

Next button

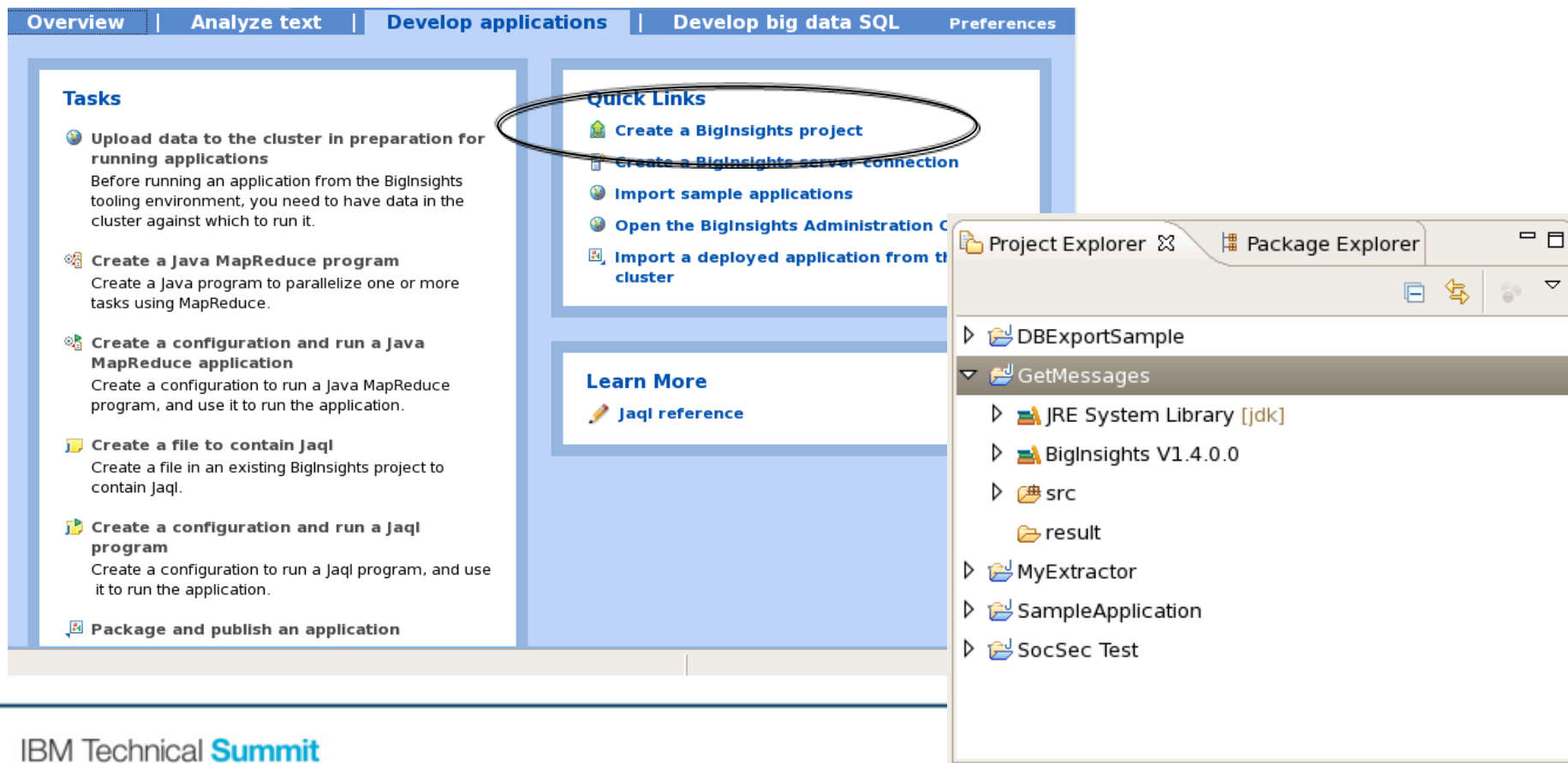
Overview of Application Development Lifecycle

- **Configure your Eclipse environment (one-time set up)**
- **Develop your application using InfoSphere BigInsights tools**
- **Test your application**
- **Package and publish your application**
- **Deploy your application on the cluster**

The screenshot displays the 'Develop applications' tab in the InfoSphere BigInsights interface. The navigation bar at the top includes 'Overview', 'Analyze text', 'Develop applications', 'Develop big data SQL', and 'Preferences'. The main content area is divided into two columns. The left column, titled 'Tasks', lists several actions with icons: 'Upload data to the cluster in preparation for running applications', 'Create a Java MapReduce program', 'Create a configuration and run a Java MapReduce application', 'Create a file to contain Jaql', 'Create a configuration and run a Jaql program', and 'Package and publish an application'. The right column, titled 'Quick Links', contains five links: 'Create a BigInsights project', 'Create a BigInsights server connection', 'Import sample applications', 'Open the BigInsights Administration Console', and 'Import a deployed application from the cluster'. Below the quick links is a 'Learn More' section with a link to 'Jaql reference'.

Develop your application – Scripting example

- Open the BigInsights perspective in Eclipse
- Create a BigInsights project



Develop your application – Scripting example (cont'd)

- Create a Jaql file within your BigInsights project

The screenshot shows the 'Develop applications' tab in the BigInsights interface. The 'Tasks' section on the left contains several instructions, with the task 'Create a file to contain Jaql' circled in red. The 'Quick Links' section on the right includes 'Create a BigInsights project', 'Create a BigInsights server co', 'Import sample applications', 'Open the BigInsights Administr', and 'Import a deployed application cluster'. Below these is a 'Learn More' section with a 'Jaql reference' link. On the right side, a file explorer dialog is open, showing the 'GetMessages' folder selected in a tree view. The 'File name' field contains 'MyJaql'. The dialog also has 'Advanced >>' and 'Cancel' buttons, and a 'Finish' button at the bottom right.

Test your application

- Create a BigInsights server connection
- Define a configuration for running your application
- Run your application from Eclipse

The image shows two overlapping dialog boxes in the Eclipse IDE. The left dialog, titled "Create a new BigInsights server", is used for setting up a connection to a BigInsights web console. It contains fields for URL (http://myserver.ibm.com:8080), Server name (TestServer), User ID (myID), and Password (masked with dots). There is a "Save password" checkbox and a "Test connection" button. The right dialog, titled "Edit configuration and launch", is used for defining the application's configuration. It shows the "JAQL Main Settings" tab with fields for Project (GetMessages), JAQL File (Myjaql.jaql), and JAQL Search Path (\${BIGINSIGHTS_MODULES}). There is a "Select a BigInsights server" dropdown menu and buttons for "Add default path", "Add", and "Remove". Both dialogs have "Cancel" and "Finish" (or "Apply" and "Revert") buttons at the bottom.

Publish your application to the console App catalog

- Package and publish your application using the InfoSphere BigInsights Eclipse Task Launcher
- Specify application name, input parms, workflow requirements, etc.

The image displays two overlapping windows from the 'BigInsights Application Publish' dialog. The left window is titled 'Specify Application' and shows the 'Application' step selected in a sequence of five steps. It includes radio buttons for 'Create New Application' (selected) and 'Replace Existing Application'. The 'Name' field contains 'GetMessages', and the 'Description' field contains 'Retrieve 15 most recent Twitter messages about IBM Watson'. The 'Icon' field shows a file path and a 'Browse' button. The 'Preview Icon' field shows a globe icon. The 'Categories' field is empty.

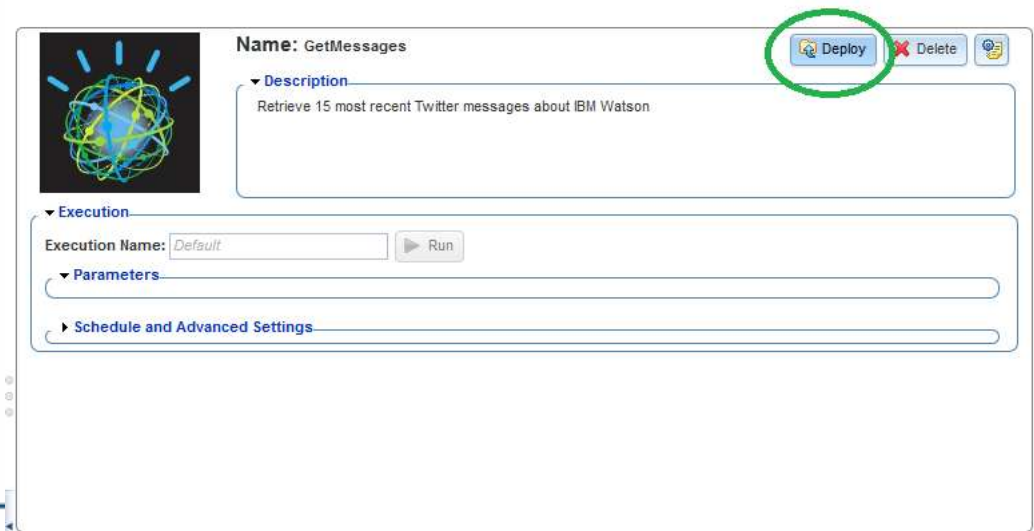
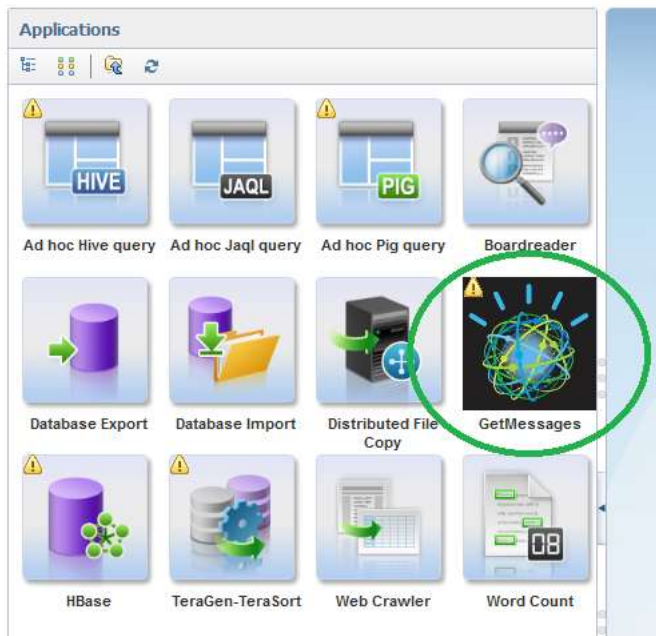
The right window is titled 'Specify Workflow' and shows the 'Workflow' step selected. It includes a radio button for 'Select an existing workflow.xml file.' and a 'Browse...' button. Below that, the 'Create a new single action workflow.xml file.' option is selected. The 'Workflow' section has a 'Type' dropdown set to 'jaql'. The 'Properties' section contains a table:

Name	Value	Variable
script	MyJaql jaql	false

Buttons for 'New...', 'Edit...', and 'Remove' are visible on the right side of the workflow table.

Deploy your application on the cluster

- Access the Applications tab of the Web console
- Locate your new app and click “deploy”
- Optionally, “run” the application after it’s been deployed



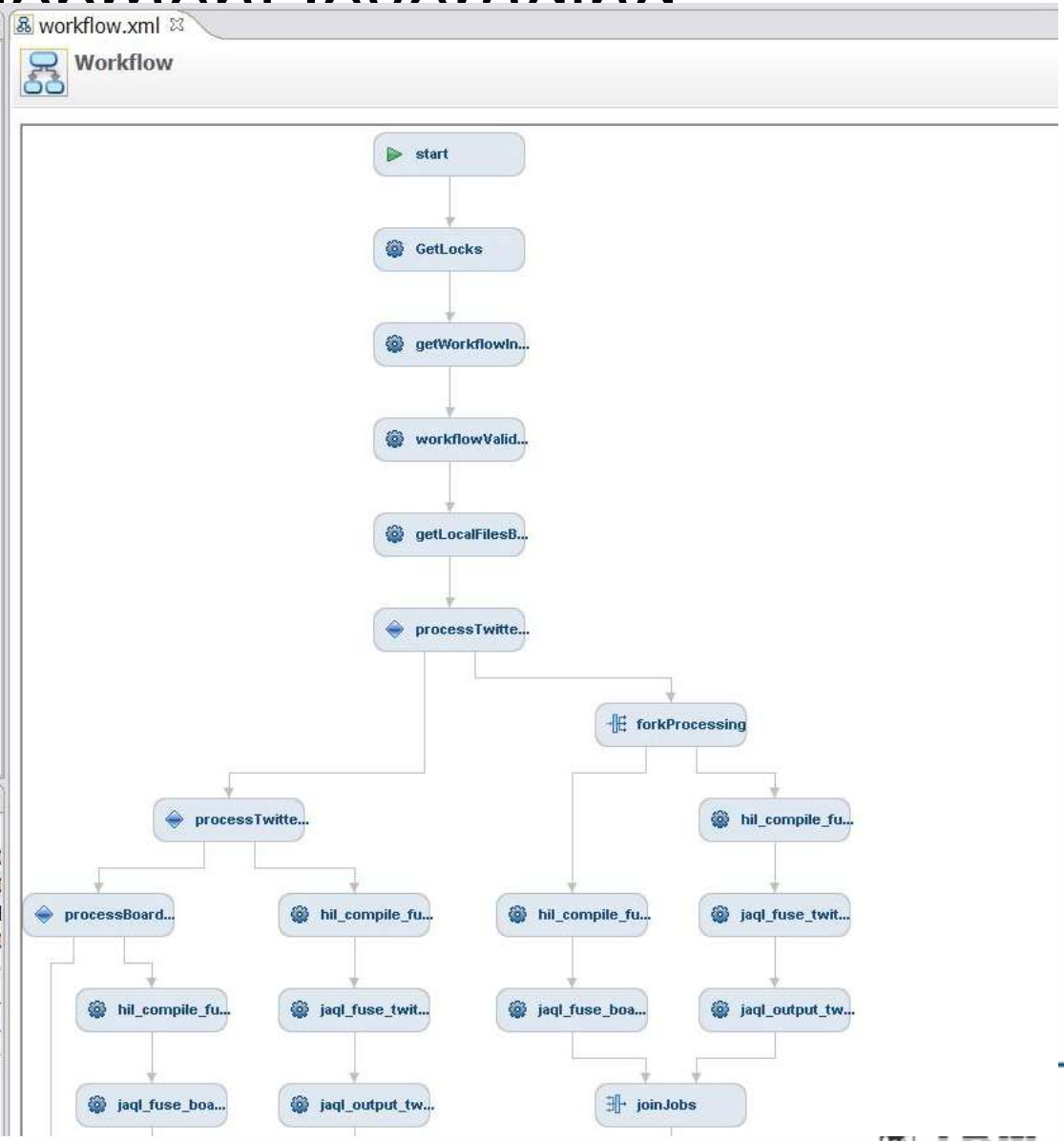
Workflow development leveraging

Workflow

Workflow Elements
Add workflow elements.

+ - ✕ ⬆ ⬇

- ▶ start
- ⚙ GetLocks
- ⚙ getWorkflowInfo
- ⚙ workflowValidation
- ⚙ getLocalFilesByJobID
- ⬠ processTwitterAndBoardReader
- ⬠ processTwitterDecision
- ⬠ processBoardReaderDecision
- ⚙ forkProcessing
- ⚙ hil_compile_fuse_twitter
- ⚙ hil_compile_fuse_boardreader
- ⚙ jaql_fuse_twitter
- ⚙ jaql_fuse_boardreader
- ⚙ jaql_output_twitter_prettyPrint
- ⚙ joinJobs
- ⚙ hil_compile_fuse_twitter2
- ⚙ hil_compile_fuse_boardreader2



Develop, run, and test SQL

The screenshot shows a SQL IDE window with a script editor and a results pane. The script editor contains the following SQL code:

```
DROP TABLE EMPLOYEE2;  
  
CREATE TABLE EMPLOYEE2 (EMPNO INT, NAME STRING, AGE INT)  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ','  
COLLECTION ITEMS TERMINATED BY '|'  
MAP KEYS TERMINATED BY ':'  
LINES TERMINATED BY '\n';  
  
load hive DATA LOCAL INPATH '/home/wenjie/employee2.data' OVERWRITE INTO TABLE EMPLOYEE2;  
  
SELECT * FROM EMPLOYEE2 /*+accessMode='local'+*/;
```

The results pane shows a table with 8 rows and 3 columns: empno, name, and age. The data is as follows:

empno	name	age
1	uttam	10
2	bert	12
3	scott	12
4	mei-mei	11
5	deepa	9
6	arun	10
7	ravi	9
8	marcel	9

The results pane also includes a log of operations with their status and dates:

Status	Operation	Date	Connectio...
✓	Succeed select * from employee3 /*+accessMode='local'+*/	9/6/12 1:54 ...	New BigSQL ...
✓	Succeed Script1.sql	9/6/12 1:57 ...	New BigSQL ...
✓	Suc DROP TABLE EMPLOYEE2	9/6/12 1:57 ...	New BigSQL ...
✓	Suc CREATE TABLE EMPLOYEE2(EMPNO INT, NAME STRIN...	9/6/12 1:57 ...	New BigSQL ...
✓	Suc load hive DATA LOCAL INPATH '/home/wenjie/employe...	9/6/12 1:57 ...	New BigSQL ...
✓	Suc SELECT * FROM EMPLOYEE2 /*+accessMode='local'+*/	9/6/12 1:57 ...	New BigSQL ...

At the bottom of the results pane, it says "Total 8 records shown" and "Displayed 2 of 13 results: 2 succeeded, ... terminated, 0 warning, 0 critical error".

Develop, run, and test Pig

The screenshot displays the Eclipse IDE interface for developing and testing Pig scripts. The main editor shows a Pig script named 'tutorial.pig' with the following code:

```
REGISTER ./tutorial.jar;

raw = LOAD 'excite-small.log' USING PigStorage('\t') AS (user, time, query);

clean1 = FILTER raw BY org.apache.pig.tutorial.NonURLDetector(query);

clean2 = FOREACH clean1 GENERATE user, time, org.apache.pig.tutorial.ToLower(query) as query;

houred = FOREACH clean2 GENERATE user, org.apache.pig.tutorial.ExtractHour(time) as hour, query;

ngramed1 = FOREACH houred GENERATE user, hour, flatten(org.apache.pig.tutorial.NGramGenerator(query)) as ngram;

ngramed2 = DISTINCT ngramed1;

hour_frequency1 = GROUP ngramed2 BY (ngram, hour);

hour_frequency2 = FOREACH hour_frequency1 GENERATE flatten($0), COUNT($1) as count;

uniq_frequency1 = GROUP hour_frequency2 BY group::ngram;

uniq_frequency2 = FOREACH uniq_frequency1 GENERATE flatten($0), flatten(org.apache.pig.tutorial.ScoreGenerator($1));

uniq_frequency3 = FOREACH uniq_frequency2 GENERATE $1 as hour, $0 as ngram, $2 as score, $3 as count, $4 as mean;

filtered_uniq_frequency = FILTER uniq_frequency3 BY score > 2.0;

ordered_uniq_frequency = ORDER filtered_uniq_frequency BY hour, score;

STORE ordered_uniq_frequency INTO 'script1-local-results.txt' USING PigStorage();
```

The bottom-left pane shows the 'BigInsights Servers' view with a context menu open over the 'svhdev03.svl.ibm.com - svhdev' server. The menu options include: Update, Delete, Test connection, Browse Files, Show jobs, Open JAQL Shell, Open Pig Shell, and Open HBase Shell.

The bottom-right pane shows the 'Problems' and 'Console' views. The console output indicates that the Pig script was executed successfully:

```
pigshell [Pig] C:\Program Files (x86)\IBM\Java60\bin\javaw.exe (Sep 6, 2012 11:10:58 AM)
2012-09-06 11:10:59,505 [main] INFO org.apache.pig.Main - Apache Pig version 0.10.0 (r: unknown) compiled Aug 08 2012, 01:3
2012-09-06 11:10:59,506 [main] INFO org.apache.pig.Main - Logging error messages to: C:\Users\IBM_ADMIN\runtime-New_configu
2012-09-06 11:10:59,787 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop f
2012-09-06 11:11:00,093 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to map-redu
grunt>
```

Develop, run, and test HBase

The screenshot displays the Eclipse IDE environment for developing and testing HBase. The main editor shows the `HBaseTest.java` file with the following code:

```
package myPackage;

import java.util.Random;

public class HBaseTest {

    public static void main(String[] args) throws Exception {

        String [] pages = {"", "page1.html", "page2.html", "page3.html"};

        Configuration hbaseConfig = HBaseConfiguration.create();
        HTable htable = new HTable(hbaseConfig, "myTable");
        htable.setAutoFlush(false);
        htable.setWriteBufferSize(1024 * 1024 * 12);

        int totalRecords = 100000;
        int maxID = totalRecords / 1000;
        Random rand = new Random();
        System.out.println("Importing " + totalRecords + " records ....");
        for (int i=0; i < totalRecords; i++)
        {
            int userID = rand.nextInt(maxID) + 1;
            byte [] rowKey = Bytes.add(Bytes.toBytes(userID), Bytes.toBytes(i));

            create "myTable", "value"
            create "myTable", "value"
            0 row(s) in 3.0780 seconds

            scan "myTable", {LIMIT => 5}
            scan "myTable", {LIMIT => 5}
            ROW COLUMN=CELL
            \x00\x00\x00\x01\x00\x00\x00\x00 column=value:page, timestamp=1346971047192, value=page2.html
            \x00\x00\x00\x01\x00\x00\x00\x00 column=value:page, timestamp=1346971047192, value=page3.html
            \x00\x00\x00\x01\x00\x00\x00\x00 column=value:page, timestamp=1346971047192, value=
            \x00\x00\x00\x01\x00\x00\x00\x01\x03 column=value:page, timestamp=1346971047192, value=page2.html
            \x00\x00\x00\x01\x00\x00\x00\x01\x08 column=value:page, timestamp=1346971047192, value=page2.html
            5 row(s) in 0.1720 seconds
```

The **Run Configurations** dialog is open, showing the configuration for running the `HBaseTest` application. The **Main class** is set to `myPackage.HBaseTest`.

The **Problems** and **Console** views are visible at the bottom. The console shows the output of the `hbase shell` command, including the creation of the `myTable` table and the execution of `scan` and `scan` commands, resulting in the following output:

```
create "myTable", "value"
create "myTable", "value"
0 row(s) in 3.0780 seconds

scan "myTable", {LIMIT => 5}
scan "myTable", {LIMIT => 5}
ROW COLUMN=CELL
\x00\x00\x00\x01\x00\x00\x00\x00 column=value:page, timestamp=1346971047192, value=page2.html
\x00\x00\x00\x01\x00\x00\x00\x00 column=value:page, timestamp=1346971047192, value=page3.html
\x00\x00\x00\x01\x00\x00\x00\x00 column=value:page, timestamp=1346971047192, value=
\x00\x00\x00\x01\x00\x00\x00\x01\x03 column=value:page, timestamp=1346971047192, value=page2.html
\x00\x00\x00\x01\x00\x00\x00\x01\x08 column=value:page, timestamp=1346971047192, value=page2.html
5 row(s) in 0.1720 seconds
```


Develop, run, and test Jaql

- Features
 - Syntax highlighting
 - Execute all or selected pieces
 - Explain output
 - Launch Jaqlshell
 - Publish as Application to BI server

Jaql execution/explain is not yet supported on Windows

```
test.jaql demo.jaql test2.jaql x
data = [ 1, 3, 100, 2 ];
sum(data);

recs = [
  {a: 1, b: 4},
  {a: 2, b: 5},
  {a: -1, b: 4}
];

recs -> transform {sum: $.a + $.b};
```

```
test2.jaql
books -> write(hdfs('BOOKS'));

//runs map reduce job
read(hdfs('books'))
-> transform {$.publisher, $.title};

//runs map reduce job
read(hdfs('books'))
-> filter exists($.reviews)
, $.title};
```

```
Problems Console SQL Results Error Log
jaqlshell [JAQL] /home/hadoop/java/ibm-java-i386-60/bin/javaw (Nov 14, 2011 9:41:56 PM)

Initializing Jaql - < version: 0.5.2; build time: November 03, 2011, 20:28:54; built for hadoop: 0.20 >

jaql>
```

BigSheets - Spreadsheet-style Analytic Tool

- Model “big data” collected from various sources as *collections* (tabular structures)
- Filter and enrich content with built-in functions
- Combine data in different collections
- Visualize results through spreadsheets, charts
- Export data into common formats (if desired)

The screenshot displays the BigSheets web interface. The top navigation bar includes 'Welcome Joe the LOB', 'Settings', and 'Logout'. The main content area is divided into several sections:

- UKParliament Configuration:** Shows 'HTML Crawler: UKParliament', 'Crawl depth: 10', and 'Maximum # of Pages: 3,000,000'. It includes a 'Recommendations' section with a slider for 'We recommend using 10 machines' and an 'Estimated run time: 8 days'. Below this is a 'Where would you like to run the crawl?' section with fields for 'Access Key', 'Secret Access Key', and 'M2 AMI ID'. A 'Run' button is present.
- Where to collect the data & how much:** A text overlay on the configuration screen.
- Dial in scalability based on needs of business:** A text overlay on the recommendations section.
- Choose PaaS from on-premise to off premise...run job:** A text overlay on the 'Run' button.
- Retrieved Collection for Analysis:** A list of collections including 'Open Calais Entities', 'Bill Pages', 'Members of Parliament', 'Members of Parliament & Party', and 'People'. Each collection has a brief description and a progress indicator.
- Progress dashboards during on going retrieval:** A text overlay on the 'Retrieved Collection for Analysis' section.
- People & Bills:** A sidebar on the right showing a table of people with columns for 'PersonName', 'Count', and 'Rank'. The table lists names like 'Ben Bradshaw', 'Simon, V', 'L. Bridgman, V', 'R. Jull', and 'B. Reevie, V'.

No programming knowledge needed!

Spreadsheet-style Data Analysis and Discovery

TestTwitter Demo

Save Exit

fx | Fit column(s) | Undo | Redo

	A	B	C	D
	id	name	screen_name	time_zone
1	71453118591807488	黑猫	kuro1209neko	null
2	71453118591803392	Iesha Young	All_DolledUp	Central Time (US & Canada)
3	71453118566629376	H E	new_helen	Mumbai
4	71453118591811584	ELECTRIC ✓	ELECTRICHAIR	Mountain Time (US & Canada)
5	71453118587600896	Jadore	mrJadore	Eastern Time (US & Canada)
6	71453118562443264	Leah Maria Savage	LilMonsterLeah	London
7	71453118600183808	RoCckii AURiEIEIE	un4seenBEAUTI	Mountain Time (US & Canada)
8	71453118587609088	Stephanie Rousell	Sofistic&dLibra	Central Time (US & Canada)
9	71453118587604992	Hiday Tata Karyos	hiday_karyos	Pacific Time (US & Canada)
10	71453118562451456	Arturo Alvarado	archivalero	Mountain Time (US & Canada)
11	71453118575026176	Saney Coetzee	SaneyCoetzee	Greenland
12	71453118579212288	Andini Anggraini Ms	andinnims	Alaska
13	71453118591811585	ibrahim jum3a	5nfoos	null
14	71453122765135872	Amp&E.L.F. SJ Only13	AmpChoKyuMin	Bangkok
			sfatlung	null
			IchaReutenia	Pacific Time (US & Canada)
			_beerpongCHAMP	Central Time (US & Canada)
			greithaalie	Pacific Time (US & Canada)
			JamesMaslow_RP	Quito
			_shimo	Hawaii
			Lorns_Maseko	null

Select a type of sheet:

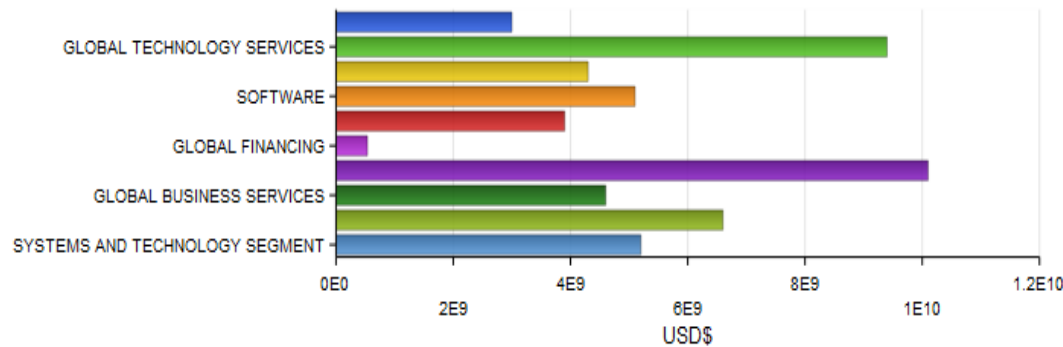
- Filter
- Macro
- Load
- Pivot
- Combine
- Union
- Limit
- Distinct
- Copy
- Formula

Add sheets | TestTwitter | Ready

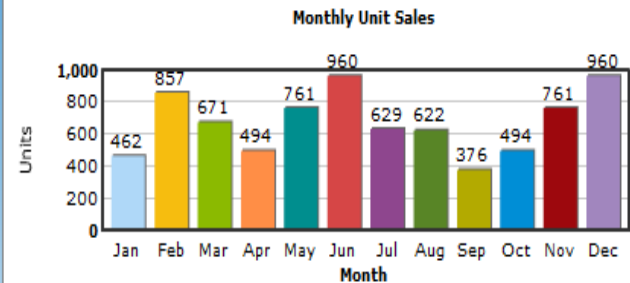
Visualize results in the customizable dashboard

Data Sampling ▾ Revenue Analysis +

REVENUE BY REGION



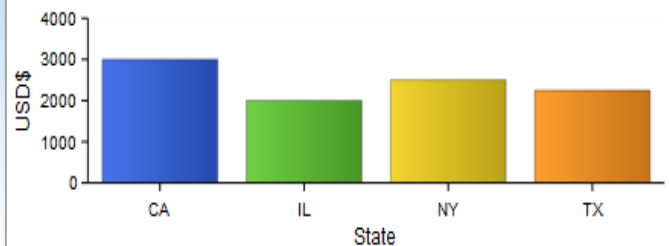
MONTHLY SALES



REVENUE

Region	Revenue_billions_of_dollars	Year	Quarter
SYSTEMS AND TECHNOLOGY SEGMENT	3900000000	2009	Q3
GLOBAL FINANCING	536000000	2009	Q3
GLOBAL TECHNOLOGY SERVICES	10100000000	2009	Q4
GLOBAL BUSINESS SERVICES	4600000000	2009	Q4

STATE INFO



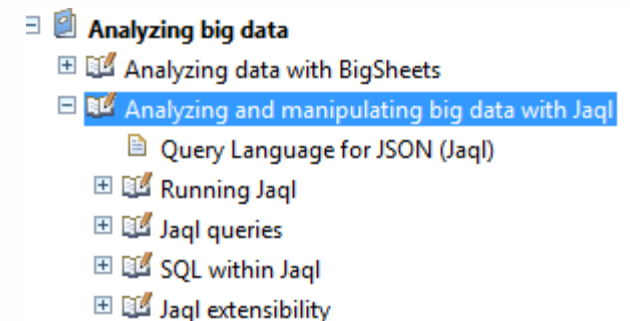
Getting Started

- In the Class Room
 - IBM Education
 - Big Data University
 - <https://bigdatauniversity.com>
- On Your Cluster
 - BigInsights 2.1 QuickStart Edition VM
 - BigInsights 2.1 Basic Edition
- Stay current
 - BigData on developerWorks
 - <http://ibm.co/bigdatadev>
 - <http://tinyurl.com/biginsights>
 - Links to demos, papers, forum, downloads, etc.
- Stay connected with IBM Big Data
 - <http://ibmbigdatahub.com>



JAQL Resources

- IBM Big Data Youtube Channel – [JAQL](#) playlist
 - Youtube.com/ibmBigData
- developerWorks articles
 - [“Query social media and structured data with InfoSphere BigInsights”](#)
 - [“Developing, publishing, and deploying your first Big Data application with InfoSphere BigInsights”](#)
- BigInsights 2.1 QuickStart Edition VM
 - JAQL Tutorial
- Big Data University
 - Learn core Hadoop
 - Let me know if you are interested in helping?
- [Information Center JAQL Docs](#)



IBM big data • IBM big data • IBM big data

THINK

BIG

BIG

IBM big data • IBM big data • IBM big data

IBM big data • IBM big data

IBM big data • IBM big data