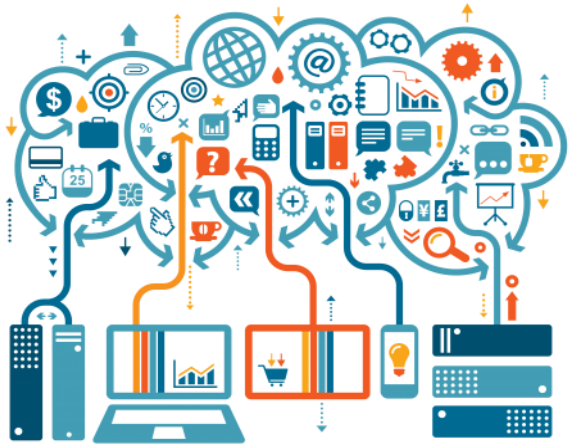
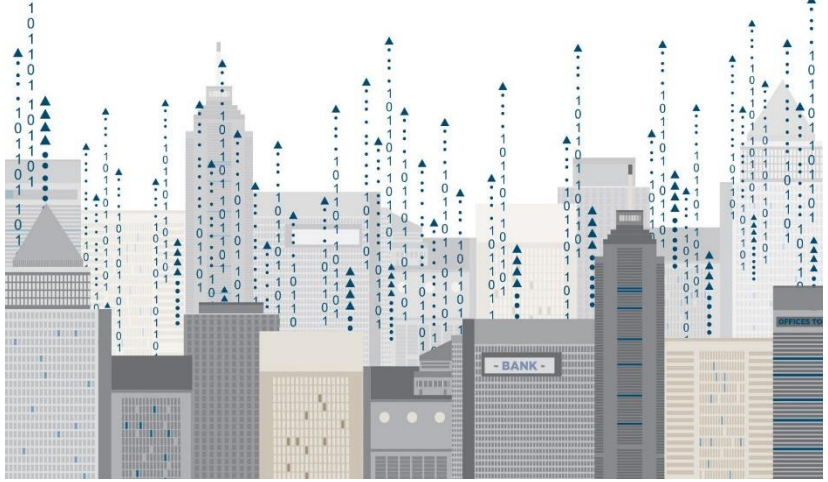




Big Data Analytics

Fall 2025

Lecture 6

```
==> error.log <==
[Fri Aug 08 04:13:33 2008] [error] [client 220.205.23.174] access to /d.exe failed for 220.205.23.174, reason: Client exceeded
[Fri Aug 08 04:13:34 2008] [error] [client 220.205.23.174] access to /d.exe failed for 220.205.23.174, reason: Client exceeded
[Fri Aug 08 04:13:39 2008] [error] [client 220.205.23.174] access to /d.exe failed for 220.205.23.174, reason: Client exceeded
[Fri Aug 08 04:13:40 2008] [error] [client 220.205.23.174] access to /d.exe failed for 220.205.23.174, reason: Client exceeded
[Fri Aug 08 04:13:46 2008] [error] [client 220.205.23.174] access to /d.exe failed for 220.205.23.174, reason: Client exceeded
[Fri Aug 08 04:32:08 2008] [error] [client 89.238.65.54] File does not exist: /errors.php
[Fri Aug 08 09:01:10 2008] [error] [client 80.25.230.39] File does not exist: /errors.php
[Fri Aug 08 09:48:09 2008] [error] [client 209.51.141.74] File does not exist: /errors.php
[Fri Aug 08 10:36:58 2008] [error] [client 209.51.141.74] File does not exist: /errors.php
[Fri Aug 08 12:18:24 2008] [error] [client 87.106.191.136] File does not exist: /errors.php
[Fri Aug 08 12:35:52 2008] [error] [client 85.214.97.225] File does not exist: /errors.php
[Fri Aug 08 15:02:35 2008] [error] [client 201.235.176.120] request failed: error reading the headers
[Fri Aug 08 15:04:58 2008] [error] [client 122.161.56.36] client denied by server configuration: /.htpasswd

==> access.log <==
131.203.76.50 - - [08/Aug/2008:19:01:19 -0700] "GET /favicon.ico HTTP/1.1" 200 42627 "-" "Mozilla/5.0 (Windows; U; Windows NT !
74.6.8.121 - - [08/Aug/2008:19:01:21 -0700] "GET / HTTP/1.0" 200 14123 "-" "Mozilla/5.0 (compatible; Yahoo! Slurp; http://help
131.203.76.50 - - [08/Aug/2008:19:01:22 -0700] "GET /favicon.ico HTTP/1.1" 200 42627 "-" "Mozilla/5.0 (Windows; U; Windows NT !
64.91.220.28 - - [08/Aug/2008:19:01:27 -0700] "GET /feed/atom/ HTTP/1.1" 302 569 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1;
64.91.220.28 - - [08/Aug/2008:19:01:28 -0700] "GET /feed/atom/ HTTP/1.1" 302 569 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1;
66.150.96.121 - - [08/Aug/2008:19:01:28 -0700] "GET /feed/ HTTP/1.1" 200 301 "-" "FeedBurner/1.0 (http://www.FeedBurner.com)"
64.91.220.28 - - [08/Aug/2008:19:01:28 -0700] "GET /feed/atom/ HTTP/1.1" 302 568 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1;
64.91.220.28 - - [08/Aug/2008:19:01:29 -0700] "GET /feed/atom/ HTTP/1.1" 302 568 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1;
64.91.220.28 - - [08/Aug/2008:19:01:29 -0700] "GET /feed/atom/ HTTP/1.1" 302 568 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1;
74.6.8.121 - - [08/Aug/2008:19:01:38 -0700] "GET /htaccess/mod_rewrite-tips-and-tricks.rdf HTTP/1.0" 200 8500 "-" "Mozilla/5.0
21.162.164.205 - - [08/Aug/2008:19:01:41 -0700] "POST /wp-admin/admin-ajax.php HTTP/1.1" 200 460 "-" "Mozilla/5.0 (Windows; U;
```

```
root@ubuntu:/var/log/apache2# cat access.log

192.168.0.104 - - [29/Aug/2017:10:04:59 -0700] "GET /dvwa HTTP/1.1" 301 574 "-" "Mozilla/5.0 (Windows NT 10.0; Win6
HTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:04:59 -0700] "GET /dvwa/ HTTP/1.1" 302 553 "-" "Mozilla/5.0 (Windows NT 10.0; Wir
KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:04:59 -0700] "GET /dvwa/login.php HTTP/1.1" 200 1086 "-" "Mozilla/5.0 (Windows NT
t/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:00 -0700] "GET /dvwa/dvwa/css/login.css HTTP/1.1" 200 740 "http://192.168.0.10
0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:00 -0700] "GET /dvwa/dvwa/images/login_logo.png HTTP/1.1" 200 9374 "http://192
ozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:00 -0700] "GET /favicon.ico HTTP/1.1" 404 502 "http://192.168.0.102/dvwa/login
T 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:33 -0700] "GET /bWAPP HTTP/1.1" 301 576 "-" "Mozilla/5.0 (Windows NT 10.0; Win
KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:33 -0700] "GET /bWAPP/ HTTP/1.1" 302 248 "-" "Mozilla/5.0 (Windows NT 10.0; Wi
(KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:33 -0700] "GET /bWAPP/portal.php HTTP/1.1" 302 384 "-" "Mozilla/5.0 (Windows N
it/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:33 -0700] "GET /bWAPP/login.php HTTP/1.1" 200 1783 "-" "Mozilla/5.0 (Windows N
it/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:33 -0700] "GET /bWAPP/stylesheets/stylesheet.css HTTP/1.1" 200 2088 "http://19
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.3
192.168.0.104 - - [29/Aug/2017:10:05:33 -0700] "GET /bWAPP/images/owasp.png HTTP/1.1" 200 17274 "http://192.168.0.1
5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:33 -0700] "GET /bWAPP/images/zap.png HTTP/1.1" 200 17843 "http://192.168.0.102
0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:33 -0700] "GET /bWAPP/images/netsparker.png HTTP/1.1" 200 2173 "http://192.168
lla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36"
192.168.0.104 - - [29/Aug/2017:10:05:33 -0700] "GET /bWAPP/images/mk.png HTTP/1.1" 200 11512 "http://192.168.0.102/
```

R15

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	EEID	Full Name	Job Title	Department	Business Unit	Gender	Ethnicity	Age	Hire Date	Annual Salar	Bonus %	Country	City	Exit Date
2	E02387	Emily Davis	Sr. Manger	IT	Research & Development	Female	Black	55	4/8/2016	$141,604	15%	United States	Seattle	10/16/2021
3	E04105	Theodore Dinh	Technical Architect	IT	Manufacturing	Male	Asian	59	11/29/1997	$99,975	0%	China	Chongqing	
4	E02572	Luna Sanders	Director	Finance	Speciality Products	Female	Caucasian	50	10/26/2006	$163,099	20%	United States	Chicago	
5	E02832	Penelope Jordan	Computer Systems Manager	IT	Manufacturing	Female	Caucasian	26	9/27/2019	$84,913	7%	United States	Chicago	
6	E01639	Austin Vo	Sr. Analyst	Finance	Manufacturing	Male	Asian	55	11/20/1995	$95,409	0%	United States	Phoenix	
7	E00644	Joshua Gupta	Account Representative	Sales	Corporate	Male	Asian	57	1/24/2017	$50,994	0%	China	Chongqing	
8	E01550	Ruby Barnes	Manager	IT	Corporate	Female	Caucasian	27	7/1/2020	$119,746	10%	United States	Phoenix	
9	E04332	Luke Martin	Analyst	Finance	Manufacturing	Male	Black	25	5/16/2020	$41,336	0%	United States	Miami	5/20/2021
10	E04533	Easton Bailey	Manager	Accounting	Manufacturing	Male	Caucasian	29	1/25/2019	$113,527	6%	United States	Austin	
11	E03838	Madeline Walker	Sr. Analyst	Finance	Speciality Products	Female	Caucasian	34	6/13/2018	$77,203	0%	United States	Chicago	
12	E00591	Savannah Ali	Sr. Manger	Human Resources	Manufacturing	Female	Asian	36	2/11/2009	$157,333	15%	United States	Miami	
13	E03344	Camila Rogers	Controls Engineer	Engineering	Speciality Products	Female	Caucasian	27	10/21/2021	$109,851	0%	United States	Seattle	
14	E00530	Eli Jones	Manager	Human Resources	Manufacturing	Male	Caucasian	59	3/14/1999	$105,086	9%	United States	Austin	
15	E04239	Everleigh Ng	Sr. Manger	Finance	Research & Development	Female	Asian	51	6/10/2021	$146,742	10%	China	Shanghai	
16	E03496	Robert Yang	Sr. Analyst	Accounting	Speciality Products	Male	Asian	31	11/4/2017	$97,078	0%	United States	Austin	3/9/2020
17	E00549	Isabella Xi	Vice President	Marketing	Research & Development	Female	Asian	41	3/13/2013	$249,270	30%	United States	Seattle	
18	E00163	Bella Powell	Director	Finance	Research & Development	Female	Black	65	3/4/2002	$175,837	20%	United States	Phoenix	
19	E00884	Camila Silva	Sr. Manger	Marketing	Speciality Products	Female	Latino	64	12/1/2003	$154,828	13%	United States	Seattle	
20	E04116	David Barnes	Director	IT	Corporate	Male	Caucasian	64	11/3/2013	$186,503	24%	United States	Columbus	
21	E04625	Adam Dang	Director	Sales	Research & Development	Male	Asian	45	7/9/2002	$166,331	18%	China	Chongqing	
22	E03680	Elias Alvarado	Sr. Manger	IT	Manufacturing	Male	Latino	56	1/9/2012	$146,140	10%	Brazil	Manaus	
23	E04732	Eva Rivera	Director	Sales	Manufacturing	Female	Latino	36	4/2/2021	$151,703	21%	United States	Miami	
24	E03484	Logan Rivera	Director	IT	Research & Development	Male	Latino	59	5/24/2002	$172,787	28%	Brazil	Rio de Janerio	
25	E00671	Leonardo Dixon	Analyst	Sales	Speciality Products	Male	Caucasian	37	9/5/2019	$49,998	0%	United States	Seattle	
26	E02071	Mateo Her	Vice President	Sales	Speciality Products	Male	Asian	44	3/2/2014	$207,172	31%	China	Chongqing	
27	E02206	Jose Henderson	Director	Human Resources	Speciality Products	Male	Black	41	4/17/2015	$152,239	23%	United States	Columbus	
28	E04545	Abigail Mejia	Quality Engineer	Engineering	Corporate	Female	Latino	56	2/5/2005	$98,581	0%	Brazil	Rio de Janerio	
29	E00154	Wyatt Chin	Vice President	Engineering	Speciality Products	Male	Asian	43	6/7/2004	$246,231	31%	United States	Seattle	
30	E03343	Carson Lu	Engineering Manager	Engineering	Speciality Products	Male	Asian	64	12/4/1996	$99,354	12%	China	Beijing	
31	E00304	Dylan Choi	Vice President	IT	Corporate	Male	Asian	63	5/11/2012	$231,141	34%	China	Beijing	
32	E02594	Ezekiel Kumar	IT Coordinator	IT	Research & Development	Male	Asian	28	6/25/2017	$54,775	0%	United States	Columbus	

Hadoop to the rescue

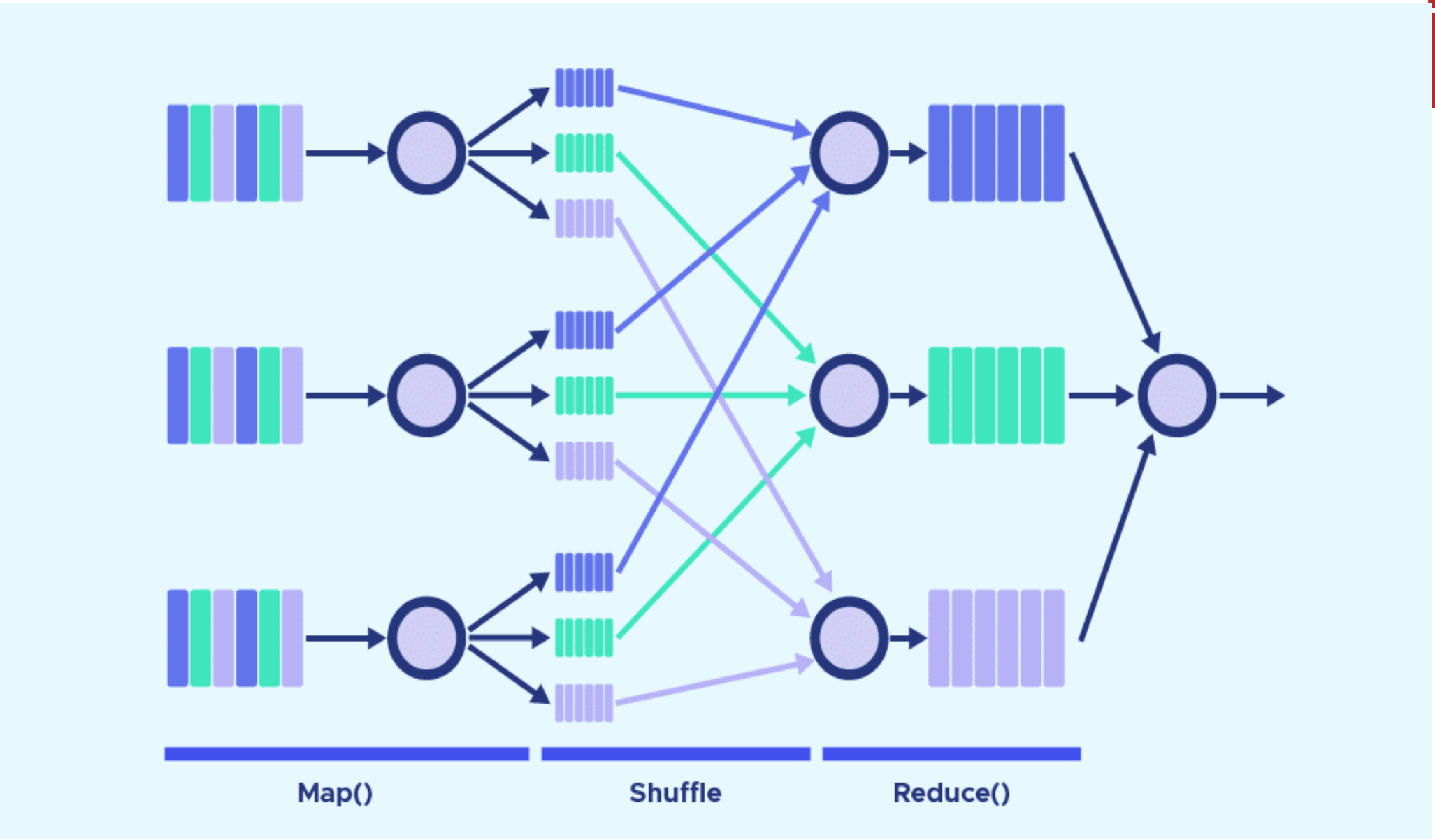
- Semi-structured data: looser, and though there may be a schema, it is often ignored, so it may be used only as a guide to the structure of the data: for example, a spreadsheet
- Unstructured data: does not have any particular internal structure
- Hadoop works well on unstructured or semi-structured data: interpret the data at processing time (schema-on-read).
- Provides flexibility and avoids the costly data loading phase of an RDBMS, since in Hadoop it is just a file copy

Storage Blues

- Normalization: problems for Hadoop | makes reading a nonlocal operation - assumes possible to perform streaming reads and writes.
- Web server log: not normalized
- The client hostnames are specified in full each time, even though the same client may appear many times
- Due to this, logfiles of all kinds are particularly well suited to analysis with Hadoop.

MR power

- MapReduce scales linearly with the size of the data
- Data is partitioned, and the functional primitives (like map and reduce) can work in parallel on separate partitions.
- If you double the size of the input data, a job will run twice as slowly.
- But if you also double the size of the cluster, a job will run as fast as the original one.
- This is not generally true of SQL queries.

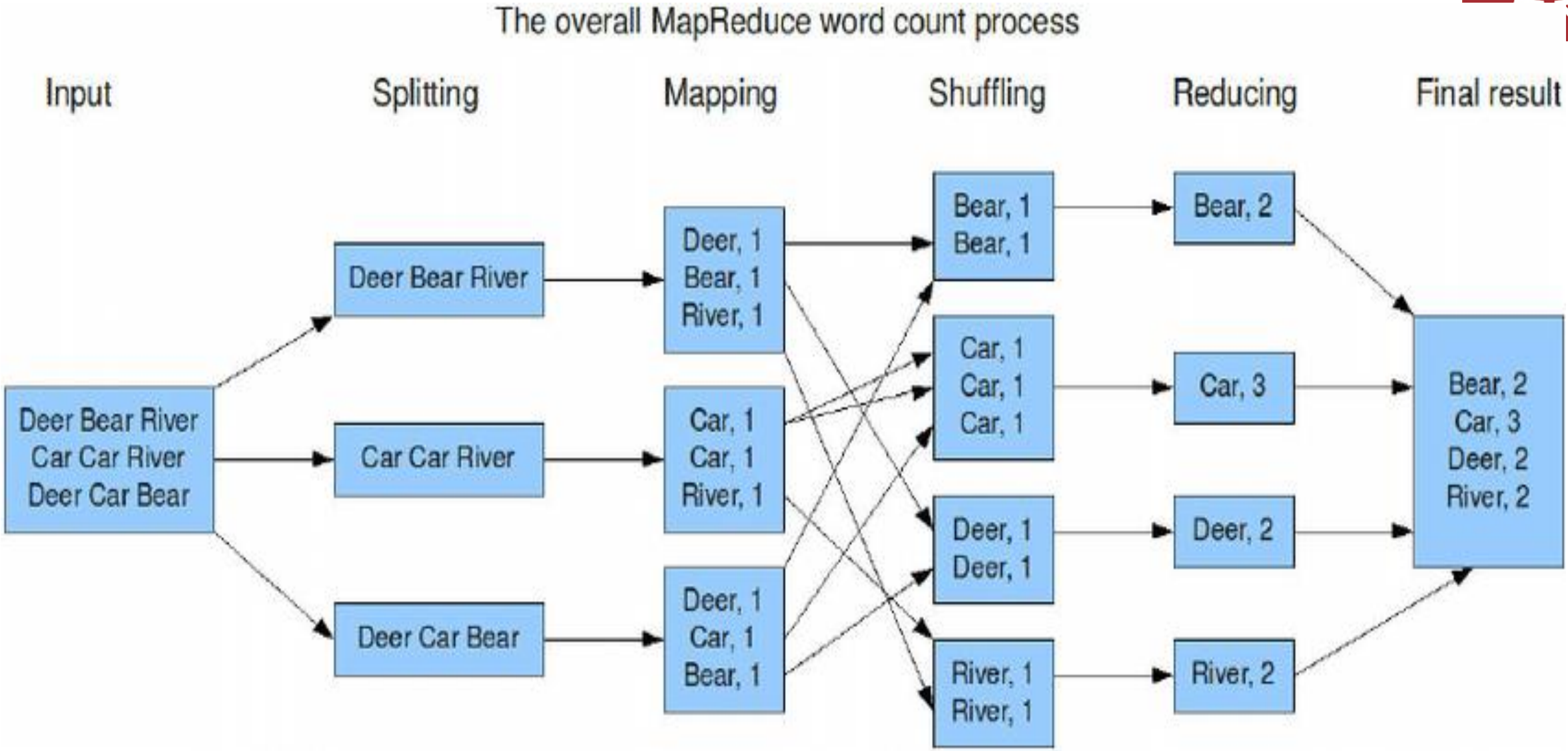

Map()
Shuffle
Reduce()

Power of MR

- Coordination in distributed computation a challenge: how to handle partial failure and still make progress?
- MapReduce spare the programmer from having to think about failure: detect failed tasks and reschedule replacements on healthy machines
- MapReduce can do this b/c its a shared-nothing architecture: tasks have no dependence on one other.
- Programmer: order of tasks doesn’t matter.

MapReduce

- MapReduce is a programming model for processing.
- Very simple
- Hadoop can run MapReduce programs written in various languages
- Our program mines weather data - Weather sensors collect data every hour at many locations across the globe and gather a large volume of log data, which is a good candidate for analysis with MapReduce because we want to process all the data, and the data is semi-structured and record-oriented.
- The data we will use is from the National Climatic Data Center, or NCDC.


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

```
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
C
N
010000    # visibility distance (meters)
1         # quality code
N
9
-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
```

```
0057
332130
99999
19500101
0300
4
+51317
+028783
FM-12
+0171
99999
V020
320
1
N
0072
1
00450
1
C
N
010000
1
N
9
-0128
1
-0139
1
10268
1
```

```
1  0029029070999991901010106004+64333+023450FM-12+000599999V0202701N015919999999N0000001N9-00781+99999102001ADDGF10899
   199999999999999999
2  0029029070999991901010113004+64333+023450FM-12+000599999V0202901N008219999999N0000001N9-00721+99999102001ADDGF10499
   199999999999999999
3  0029029070999991901010120004+64333+023450FM-12+000599999V0209991C000019999999N0000001N9-00941+99999102001ADDGF10899
   199999999999999999
4  0029029070999991901010206004+64333+023450FM-12+000599999V0201801N008219999999N0000001N9-00611+99999101831ADDGF10899
   199999999999999999
5  0029029070999991901010213004+64333+023450FM-12+000599999V0201801N009819999999N0000001N9-00561+99999101761ADDGF10899
   199999999999999999
6  0029029070999991901010220004+64333+023450FM-12+000599999V0201801N009819999999N0000001N9-00281+99999101751ADDGF10899
   199999999999999999
7  0029029070999991901010306004+64333+023450FM-12+000599999V0202001N009819999999N0000001N9-00671+99999101701ADDGF10699
   199999999999999999
8  0029029070999991901010313004+64333+023450FM-12+000599999V0202301N011819999999N0000001N9-00331+99999101741ADDGF10899
   199999999999999999
9  0029029070999991901010320004+64333+023450FM-12+000599999V0202301N011819999999N0000001N9-00281+99999101741ADDGF10899
   199999999999999999
10 0029029070999991901010406004+64333+023450FM-12+000599999V0209991C000019999999N0000001N9-00331+99999102311ADDGF10899
   199999999999999999
11 0029029070999991901010413004+64333+023450FM-12+000599999V0202301N008219999999N0000001N9-00441+99999102261ADDGF10899
   199999999999999999
12 0029029070999991901010420004+64333+023450FM-12+000599999V0202001N011819999999N0000001N9-00391+99999102231ADDGF10899
   199999999999999999
13 0029029070999991901010506004+64333+023450FM-12+000599999V0202701N004119999999N0000001N9+00001+99999101821ADDGF10499
```

```
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
C
N
010000   # visibility distance (meters)
1        # quality code
N
9
-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
```

Datafiles are organized by date and weather station

What's the highest recorded global temperature for each year in the dataset?

The classic tool for processing line-oriented data is awk

```
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
C
N
010000   # visibility distance (meters)
1        # quality code
N
9
-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
```

```
#!/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
```

```
% ./max_temperature.sh
1901 317
1902 244
1903 289
1904 256
1905 283
...
```

The need for MapReduce

- To speed up, we need to run parts of the program in parallel
- We could process different years in different processes
- Dividing the work into equal-size pieces isn't always easy
- Better: Split the input into fixed-size chunks and assign each chunk to a process
- Combining the results from independent processes may require further processing.
  - Result of each year independent of other years - combined by concatenating all the results and sorting by year.
  - If using the fixed-size chunk approach, the combination is more delicate. For this example, data for a particular year will typically be split into several chunks

The need for MapReduce

- You are still limited by the processing capacity of a single machine.
- If the best time you can achieve is 20 minutes with the number of processors you have, then that’s it.
- You can’t make it go faster

MapReduce

- Breaking the processing into two phases: the map phase and the reduce phase.
- Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer - Specifies two functions: the map function and the reduce function.
- The input to our map phase is the raw NCDC data:
  - We pull out the year and the air temperature, because these are the only fields we are interested in.
  - Map is a data preparation phase - setting up the data so that reduce function can do its work on it: finding the maximum temperature for each year.
- The map function is also a good place to drop bad records: here we filter out temperatures that are missing, suspect, or erroneous.

To visualize the way the map works, consider the following sample lines of input data (some unused columns have been dropped to fit the page, indicated by ellipses):

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

These lines are presented to the map function as the key-value pairs:

```
(0, 0067011990999991950051507004...9999999N9+00001+99999999999...)
(106, 0043011990999991950051512004...9999999N9+00221+99999999999...)
(212, 0043011990999991950051518004...9999999N9-00111+99999999999...)
(318, 0043012650999991949032412004...0500001N9+01111+99999999999...)
(424, 0043012650999991949032418004...0500001N9+00781+99999999999...)
```

The keys are the line offsets within the file, which we ignore in our map function. The map function merely extracts the year and the air temperature (indicated in bold text), and emits them as its output (the temperature values have been interpreted as integers):

```
(1950, 0)
(1950, 22)
(1950, -11)
(1949, 111)
(1949, 78)
```

The output from the map function is processed by the MapReduce framework before being sent to the reduce function. This processing sorts and groups the key-value pairs by key. So, continuing the example, our reduce function sees the following input:

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

Each year appears with a list of all its air temperature readings. All the reduce function has to do now is iterate through the list and pick up the maximum reading:

```
(1949, 111)
(1950, 22)
```

This is the final output: the maximum global temperature recorded in each year.

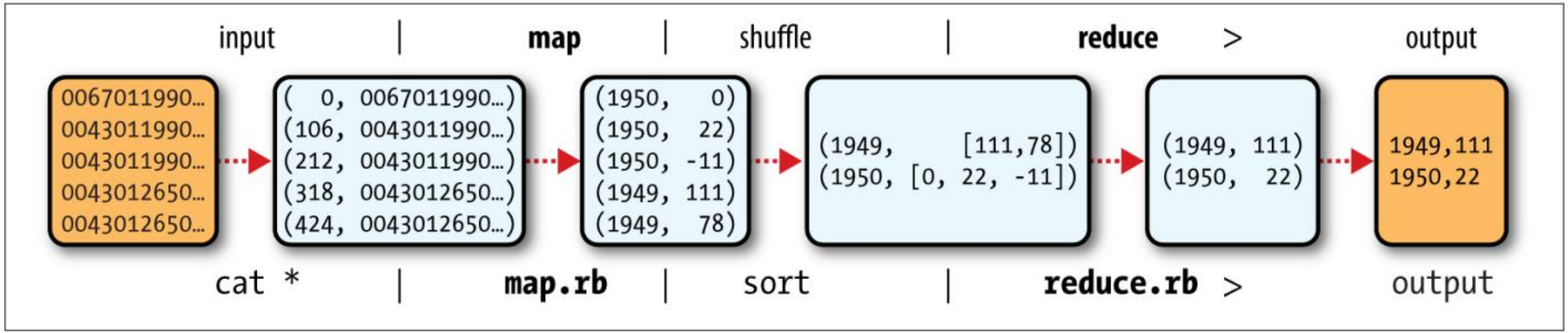


Figure 2-1. MapReduce logical data flow

Example 2-3. Mapper for the maximum temperature example

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MaxTemperatureMapper
    extends Mapper<LongWritable, Text, Text, IntWritable> {

  private static final int MISSING = 9999;

  @Override
  public void map(LongWritable key, Text value, Context context)
      throws IOException, InterruptedException {

    String line = value.toString();
    String year = line.substring(15, 19);
    int airTemperature;
    if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
      airTemperature = Integer.parseInt(line.substring(88, 92));
    } else {
      airTemperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);

    if (airTemperature != MISSING && quality.matches("[01459]")) {
      context.write(new Text(year), new IntWritable(airTemperature));
    }
  }
}
```

Example 2-4. Reducer for the maximum temperature example

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MaxTemperatureReducer
    extends Reducer<Text, IntWritable, Text, IntWritable> {

  @Override
  public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {

    int maxValue = Integer.MIN_VALUE;
    for (IntWritable value : values) {
      maxValue = Math.max(maxValue, value.get());
    }
    context.write(key, new IntWritable(maxValue));
  }
}
```

The input types of the reduce function must match the output types of the map function: Text and IntWritable

Example 2-5. Application to find the maximum temperature in the weather dataset

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MaxTemperature {

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: MaxTemperature <input path> <output path>");
      System.exit(-1);
    }

    Job job = new Job();
    job.setJarByClass(MaxTemperature.class);
    job.setJobName("Max temperature");

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.setMapperClass(MaxTemperatureMapper.class);
    job.setReducerClass(MaxTemperatureReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

MapReduce

- A Job object forms specification of the work and gives you control over how the job is run.
- The job packages the code into a JAR file - distributed around the cluster
  - We simply pass a class in the Job's setJarByClass() method - Hadoop uses this to locate the relevant JAR file
- We then specify the input and output paths.
- Input path: Call the static *addInputPath()* method on *FileInputFormat*, and it can be a single file, a directory, or a file pattern.
- addInputPath(): can be called more than once to use input from multiple paths.

MapReduce

- The output path: specified by the static *setOutput Path()* method on *FileOutputFormat*.
- It specifies a directory where the output files from the reduce function are written.
- The directory shouldn’t exist before running the job because Hadoop will complain and not run the job.
- This precaution is to prevent data loss
- It can be very annoying to accidentally overwrite the output of a long job with that of another.

MapReduce

- We specify the map and reduce types to use via the setMapperClass() and setReducerClass() methods.
- The setOutputKeyClass() and setOutputValueClass() methods control the output types for the reduce function - and must match what the Reduce class produces.
- The map output types default to the same types, so they do not need to be set if the mapper produces the same types as the reducer (as it does in our case).
- However, if they are different, the map output types must be set using the setMapOutputKeyClass() and setMapOutputValueClass() methods.

MapReduce

- The input types are controlled via the input format, which we have not explicitly set because we are using the default TextInputFormat.
- After setting the classes that define the map and reduce functions, we are ready to run the job.
- The waitForCompletion() method on Job submits the job and waits for it to finish –
  - The single argument to the method is a flag indicating whether verbose output is generated.
  - When true, the job writes information about its progress to the console.
- The return value of the waitForCompletion() method is a Boolean indicating success (true) or failure (false), which we translate into the program's exit code of 0 or 1.

```
% export HADOOP_CLASSPATH=hadoop-examples.jar
% hadoop MaxTemperature input/ncdc/sample.txt output
14/09/16 09:48:39 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
14/09/16 09:48:40 WARN mapreduce.JobSubmitter: Hadoop command-line option
parsing not performed. Implement the Tool interface and execute your application
with ToolRunner to remedy this.
14/09/16 09:48:40 INFO input.FileInputFormat: Total input paths to process : 1
14/09/16 09:48:40 INFO mapreduce.JobSubmitter: number of splits:1
14/09/16 09:48:40 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_local26392882_0001
14/09/16 09:48:40 INFO mapreduce.Job: The url to track the job:
http://localhost:8080/
14/09/16 09:48:40 INFO mapreduce.Job: Running job: job_local26392882_0001
14/09/16 09:48:40 INFO mapred.LocalJobRunner: OutputCommitter set in config null
14/09/16 09:48:40 INFO mapred.LocalJobRunner: OutputCommitter is
org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
14/09/16 09:48:40 INFO mapred.LocalJobRunner: Waiting for map tasks
14/09/16 09:48:40 INFO mapred.LocalJobRunner: Starting task:
attempt_local26392882_0001_m_000000_0
14/09/16 09:48:40 INFO mapred.Task:  Using ResourceCalculatorProcessTree : null
14/09/16 09:48:40 INFO mapred.LocalJobRunner:
14/09/16 09:48:40 INFO mapred.Task: Task:attempt_local26392882_0001_m_000000_0
is done. And is in the process of committing
14/09/16 09:48:40 INFO mapred.LocalJobRunner: map
14/09/16 09:48:40 INFO mapred.Task: Task 'attempt_local26392882_0001_m_000000_0'
 done.
14/09/16 09:48:40 INFO mapred.LocalJobRunner: Finishing task:
attempt_local26392882_0001_m_000000_0
14/09/16 09:48:40 INFO mapred.LocalJobRunner: map task executor complete.
14/09/16 09:48:40 INFO mapred.LocalJobRunner: Waiting for reduce tasks
14/09/16 09:48:40 INFO mapred.LocalJobRunner: Starting task:
attempt_local26392882_0001_r_000000_0
14/09/16 09:48:40 INFO mapred.Task:  Using ResourceCalculatorProcessTree : null
14/09/16 09:48:40 INFO mapred.LocalJobRunner: 1 / 1 copied.
14/09/16 09:48:40 INFO mapred.Merger: Merging 1 sorted segments
14/09/16 09:48:40 INFO mapred.Merger: Down to the last merge-pass, with 1
segments left of total size: 50 bytes
14/09/16 09:48:40 INFO mapred.Merger: Merging 1 sorted segments
14/09/16 09:48:40 INFO mapred.Merger: Down to the last merge-pass, with 1
segments left of total size: 50 bytes
14/09/16 09:48:40 INFO mapred.LocalJobRunner: 1 / 1 copied.
14/09/16 09:48:40 INFO mapred.Task: Task:attempt_local26392882_0001_r_000000_0
is done. And is in the process of committing
14/09/16 09:48:40 INFO mapred.LocalJobRunner: 1 / 1 copied.
14/09/16 09:48:40 INFO mapred.Task: Task attempt_local26392882_0001_r_000000_0
```

```
is allowed to commit now
14/09/16 09:48:40 INFO output.FileOutputCommitter: Saved output of task
'attempt...local26392882_0001_r_000000_0' to file:/Users/tom/book-workspace/
hadoop-book/output/_temporary/0/task_local26392882_0001_r_000000
14/09/16 09:48:40 INFO mapred.LocalJobRunner: reduce > reduce
14/09/16 09:48:40 INFO mapred.Task: Task 'attempt_local26392882_0001_r_000000_0'
 done.
14/09/16 09:48:40 INFO mapred.LocalJobRunner: Finishing task:
attempt_local26392882_0001_r_000000_0
14/09/16 09:48:40 INFO mapred.LocalJobRunner: reduce task executor complete.
14/09/16 09:48:41 INFO mapreduce.Job: Job job_local26392882_0001 running in uber
 mode : false
14/09/16 09:48:41 INFO mapreduce.Job:  map 100% reduce 100%
14/09/16 09:48:41 INFO mapreduce.Job: Job job_local26392882_0001 completed
successfully
14/09/16 09:48:41 INFO mapreduce.Job: Counters: 30
    File System Counters
        FILE: Number of bytes read=377168
        FILE: Number of bytes written=828464
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
    Map-Reduce Framework
        Map input records=5
        Map output records=5
        Map output bytes=45
        Map output materialized bytes=61
        Input split bytes=129
        Combine input records=0
        Combine output records=0
        Reduce input groups=2
        Reduce shuffle bytes=61
        Reduce input records=5
        Reduce output records=2
        Spilled Records=10
        Shuffled Maps =1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=39
        Total committed heap usage (bytes)=226754560
    File Input Format Counters
        Bytes Read=529
    File Output Format Counters
        Bytes Written=29
```