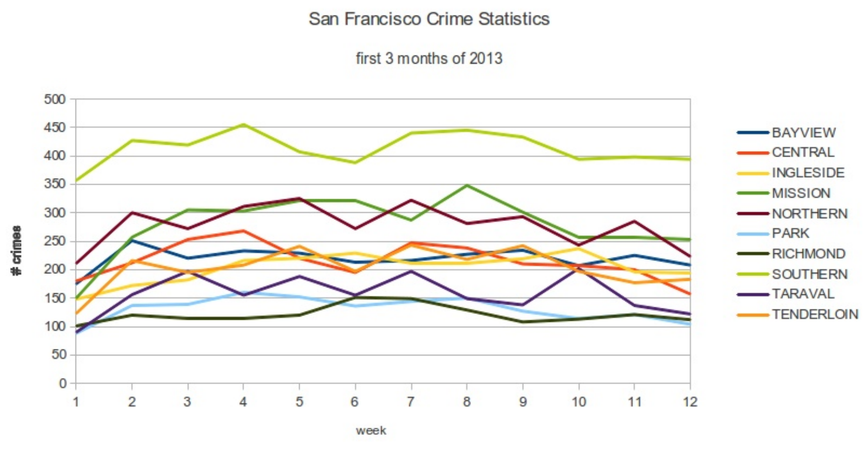
a gentle introduction to

Big Data with Open Source

by Glenn Engstrand

In this paper, I describe how companies on a budget can benefit from big data by using freely available open source software. This is a hands on introduction with accompanying source code in a github repository demonstrating common analysis techniques on crime statistics data for the city of San Francisco.



There's a lot of buzz about big data these days. Most of the coverage is executive where the focus is either on volume, velocity, variety, variability or transactions, interactions, and observations. I thought that it was time to cover the big data topic from a developer's perspective.

Hadoop is a menagerie of sub projects that is very critical to big data.

Nothing is more illustrative than a real world example. I wanted it to be simple enough to follow along for educational purposes but more complicated than the trivial word count example. I wanted the data to be somewhat of interest and small enough to run on a laptop but big enough to point out some gotchas and quirks of data processing. I decided to pick Q1 2013 crime data for San Francisco which logged 26,668 incidents from 34 categories in 10 districts. You can freely download the current data set as a CSV or Comma Separated Values text file from the data.sfgov.org web site.

The first, and most important, open source technology for processing big data is the Apache Hadoop project. Why is Hadoop so important? Prior to Hadoop, if you wanted to process big data, then you had to get a big computer. The cost of owning and operating a mainframe puts it in the reach of utilities, airline companies, and small countries. Hadoop changes all of that. What Hadoop does is break a big job into lots of little jobs that can be run on a cluster of commodity machines. Coupled with cloud computing, this puts big data within reach of SMB and start ups.

Hadoop isn't a single project. Rather it is a menagerie of projects that co-exist together. The heart and soul of Hadoop is the HDFS or Hadoop Distributed File System. What allows you to break up one big job into lots of little jobs is Hadoop Map/Reduce. If you're allergic to the Java programming language, then you can use either pig or hive which sits on top of map/reduce and HDFS. You'll like pig if you are into scripting. Hive will attract you if you are into SQL.

The hive approach to processing our sfcrime statistics would go something like this. First, you must import that CSV file into HDFS.

```
bin/hadoop dfs -put sfcrime.csv /user/hive/warehouse/sfcrime
```

Then you would build a raw hive table from the contents of that file.

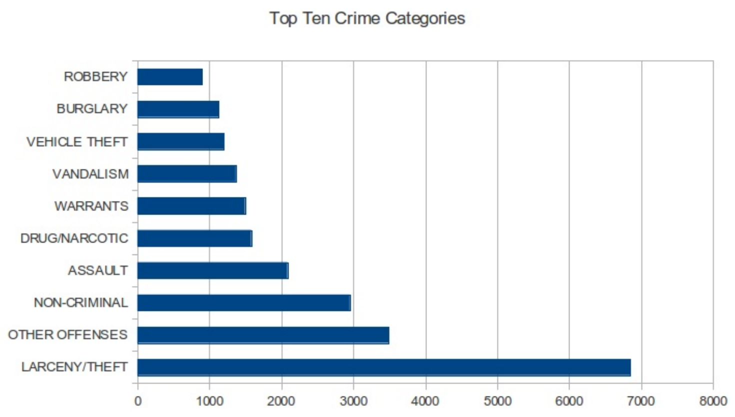
```
create external table sfcrime_raw (IncidntNum string, Category
string, Descript string, DayOfWeek string, Date string, Time
string, PdDistrict string, Resolution string, Address string, X
string, Y string, Location string) row format of delimited fields
terminated by ',' lines terminated by '\n' stored as textfile
location '/user/hive/warehouse/sfcrime';
```

Finally you would group the raw data into the summary report data of interest.

```
insert overwrite local directory 'categoryTotal' select Category,
count(1) from sfcrime_raw group by Category;
```

Hive has its limitations. It can't parse the sfcrime data. Let's dive a little deeper and see what it takes to write Java programs that use the Hadoop map/reduce infrastructure directly.

The simplest Hadoop map/reduce jobs run in two phases, a mapper phase and a reducer phase. The mapper takes the input to the job and outputs key, value pairs to the collector. The infrastructure collates the output from the mapper such that all the values with the same key that were output from the mapper become a collection associated with that key that is input to the reducer. The reducer performs the relevant summary logic and outputs that to its collector which ends up being files in a folder designated for that purpose.



The mapper collects data grouped by Hadoop for the reducer to summarize.

There is a sample project that contains all the corresponding code referred to in this article. You can find all the code here. Code excerpts in the article are simplified for illustrative purposes so you should always go to the project in this repository to really study the code.

There are three sample map/reduce jobs in that project. One generates a report of San Francisco crime totals by category vs week. Another generates San Francisco crime by district vs week. The third job creates a daily activity report necessary to load the San Francisco crime statistics into an OLAP cube. More on that later.

https://github.com/gengstrand/map-reduce-sf-crime

Let's review the district by week job. The mapper parses the raw CSV file and collects intermediate output where the district name is the key and the date that the crime occurred is the value.

```
String[] col = DataFile.getColumns(value.toString());
Text tk = new Text();
tk.set(col[DISTRICT_COLUMN_INDEX]);
Text tv = new Text();
tv.set(col[DATE_COLUMN_INDEX]);
output.collect(tk, tv);
```

The reducer gets called where the district name is the key and the iterator contains all the dates when a crime occurred in that district. The output, that the reducer collects, is a weekly total of these crimes by district which ends up as something similar to a CSV file in the output folder.

```
List<String> incidents = new ArrayList<String>();
while (values.hasNext()) {
   incidents.add(values.next().toString());
}
Collections.sort(incidents);
java.util.Map<Integer, Integer> weekSummary = new HashMap<Integer,
Integer>();
for (String incidentDay : incidents) {
   Date d = getDate(incidentDay);
   Calendar cal = Calendar.getInstance();
   cal.setTime(d);
   int week = cal.get(Calendar.WEEK_OF_MONTH);
   int month = cal.get(Calendar.MONTH);
   int bucket = (month * 5) + week;
   weekSummary.put(bucket, new
Integer(weekSummary.get(bucket).intValue() + 1));
}
StringBuffer rpt = new StringBuffer();
boolean first = true;
for (int week : weekSummary.keySet()) {
   if (first) {
      first = false;
   } else {
      rpt.append(",");
   }
   rpt.append(new Integer(weekSummary.get(week)).toString());
}
String list = rpt.toString();
Text tv = new Text();
tv.set(list);
output.collect(key, tv);
```

It's just that easy. Input looks like this.

```
130245011,BURGLARY,"BURGLARY OF RESIDENCE, UNLAWFUL
ENTRY",Monday,03/25/2013 07:00:00 AM +0000,21:15,BAYVIEW,NONE,1500
Block of MCKINNON AV,-
122.388542742273,37.7353389991176,"(37.7353389991176,
-122.388542742273)"
```

And the output to this job looks like this.

```
BAYVIEW,0,175,251,220,233,155,74,213,216,227,167,67,207,225,177,31
CENTRAL,0,180,212,253,268,156,64,195,247,238,153,57,207,200,145,12
```

The need to summarize raw data into reports is nothing new and there have been plenty of report writers prior to Hadoop. Here is an example, written in awk, that generates the total number of crimes for each category.

```
BEGIN {
    FS=","
}
/^[0-9]/{
    if ($2 in categoryTotal) {
     categoryTotal[$2] += 1
    } else {
     categoryTotal[$2] = 1
    }
}
END {
    for (category in categoryTotal) {
     printf("%s,%d\n", category, categoryTotal[category])
    }
}
```

The true innovation that Hadoop brings isn't just different syntax. It is that Hadoop fully embraces distributed computing without the map/reduce developer having to know much about distributed computing. With report writers like awk, if the data was too big to all fit on one machine, then you're out of luck. With Hadoop, just add more servers. That is why Hadoop is compelling for big data.

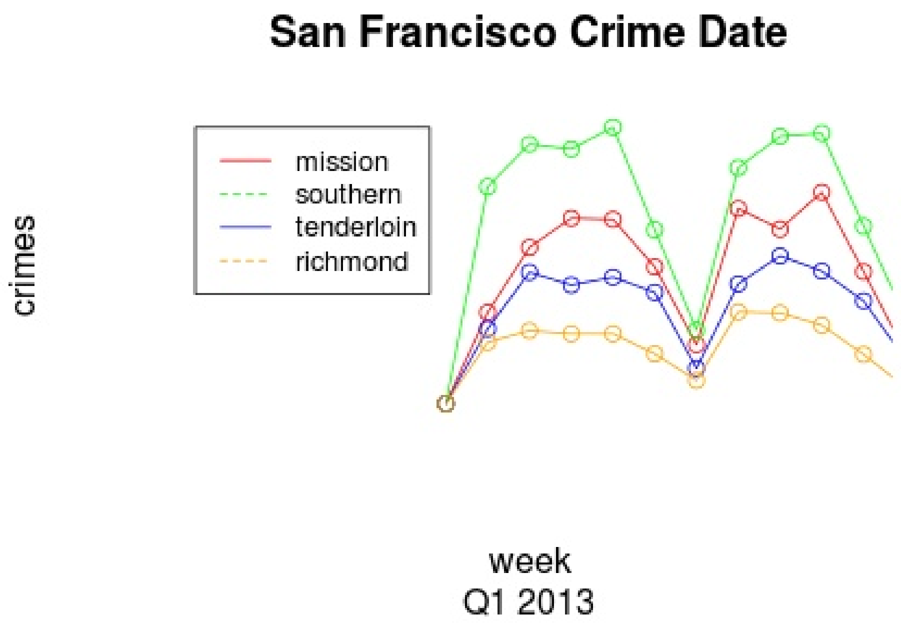
Hadoop fully embraces distributed computing.

I hope that you are as excited about this as I am. This is totally cool but it is hard to visualize trends or gain analytical insight just by looking at a comma delimited list of numbers. Most folks use spreadsheet software to visualize data like this. You can open up this by district by week output CSV file in your favorite spreadsheet software and use the chart wizard to generate a pretty graph. You get to select from a wide variety of chart types. You might have to fiddle with the data range and the data series in order to represent the information appropriately. You can also customize the chart elements too.

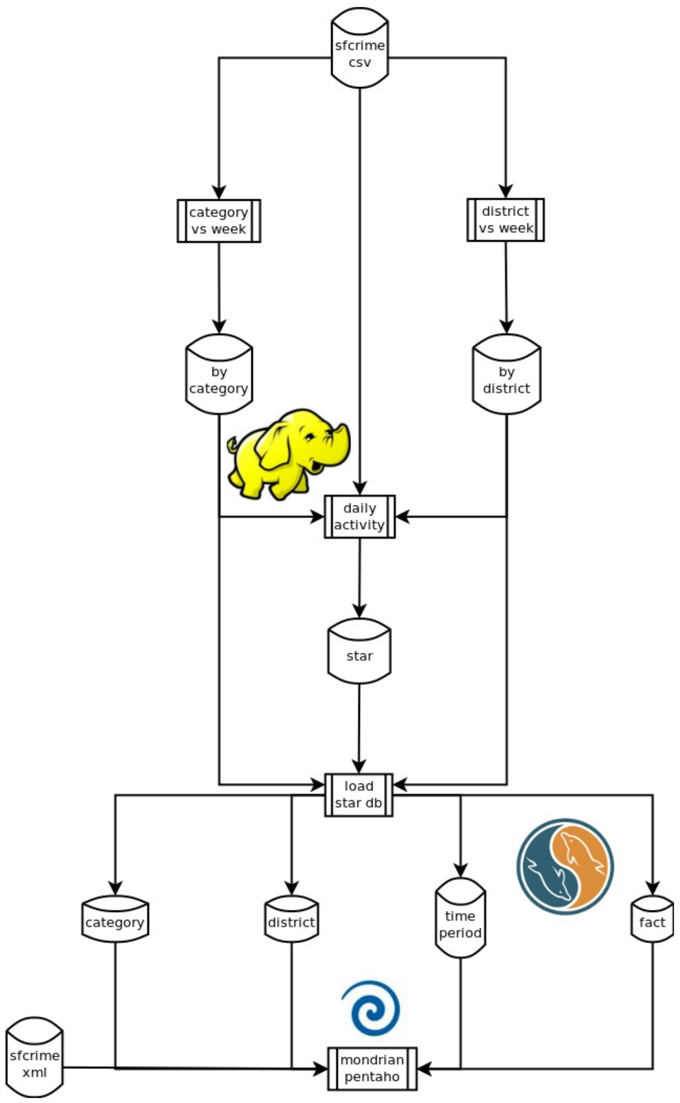
Data scientists who like to code will most probably use the R programming language to graph data like this. You can read the CSV file into a data frame which can then be plotted. The "t" function transforms or pivots the data for plotting. You need to get the range of values to set the vertical limits of the graph. The lines command can plot multiple lines on the same graph. Other commands such as title and legend can complete the graphic. Here is a code excerpt for plotting weekly crime counts for Richmond, Southern, Tenderloin, and Mission districts.

```
district <- read.csv('/home/glenn/oss/hadoop/hadoop-
1.0.4/bydistrict/part-00000', header = FALSE)
pr <- range(0, district$V2, district$V3, district$V4, district$V5,
district$V6)
mission <- t(district[district$V1 == "MISSION",])
southern <- t(district[district$V1 == "SOUTHERN",])
tenderloin <- t(district[district$V1 == "TENDERLOIN",])
richmond <- t(district[district$V1 == "RICHMOND",])
plot(mission, type="o", col="red", ylim=pr, xlim=c(-4, 12),
axes=FALSE, ann=FALSE)
lines(southern, type="o", col="green")
lines(tenderloin, type="o", col="blue")
lines(richmond, type="o", col="orange")
legend(-4, max(pr), c("mission", "southern", "tenderloin",
"richmond"), col=c("red", "green", "blue", "orange"), cex=0.8,
lty=1:2)
title(main="San Francisco Crime Date", sub="Q1 2013", xlab="week",
ylab="crimes")
```

And here is the corresponding chart. Notice the dip in all four plot lines? That is not to be interpreted as a drop in crime for that week. Rather, it is an artifact of how Java handles week of the month calculations. Months and weeks don't align perfectly. A week can be divided such that the first half is in one month while the rest is in the next month. That dip is because the first week of February only had two days in it.



Charts like this are static. You can't navigate through the chart interactively and explore the data. That is why most analysts use OLAP technology. Poorly named On Line Analytical Processing allows you to surface and navigate through hierarchical, multi-dimensional data using common spreadsheet software. Most companies who use OLAP to navigate through their big data typically use the OLAP technology offered by either Microsoft or Oracle. In this article, I will show you how to use Hadoop map/reduce to load San Francisco crime data into an open source OLAP project called Pentaho Mondrian.



You will find three map/reduce jobs in the map-reduce-sf-crime project. Category by week and district by week generate report data from the raw data. Daily activity takes the output from those two other jobs and the raw crime data to create a CSV file that the ETL load star db program uses to load the relational database tables that get mapped to the OLAP cube.

Most OLAP technologies these days use a relational database to actually house the data in what is called a star schema. The relational database technology of choice for Mondrian is MySql.

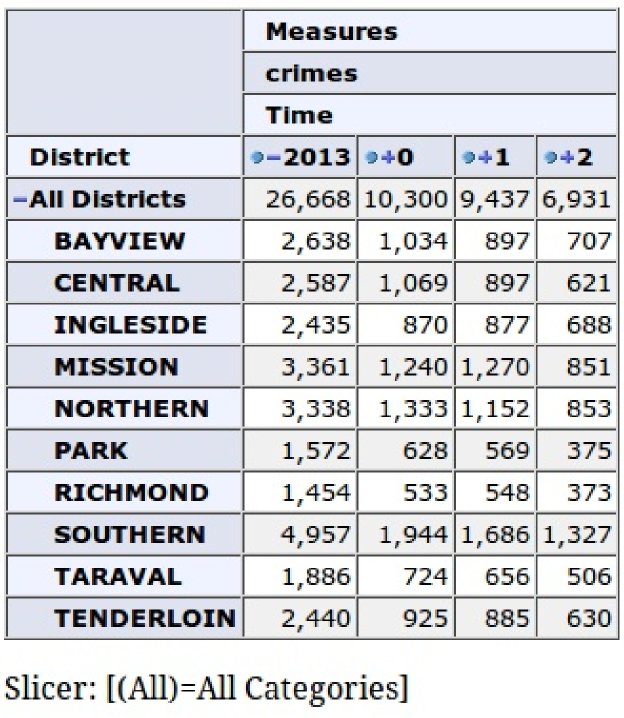
The daily activity map/reduce job takes the raw crime data and generates output where every row is a comma delimited 4-tuple of date, district, category, and number of crimes committed on that date, in that district, and fit that category. The load star db program takes the category by week, district by week, and daily activity files and uses that data to populate the MySql tables used by Mondrian. This load star db program is an example of an ETL or Extract, Transform, and Load job. Most ETL suites, such as Pentaho Kettle, don't require much programming.

This is a pretty simple and strait forward three dimensional cube. The only measure in this cube is a count of the number of crimes committed. The dimensions are category, district, and time. There is an XML file in the project that maps this three dimensional cube to tables in the mysql database. The district dimension is mapped to the district table. The category dimension is mapped to the category table. The time dimension is organized into a hierarchy of year, month, week, and day which is captured in the timeperiod table. In the sfcrime.xml file, hierarchy is modeled by the level tag. Notice the levelType attributes TimeYears, TimeMonths, TimeWeeks, TimeDays. When you include this attribute with one of these values in the level tag, Mondrian understands that this is a time based hierarchy. The crime measure is mapped to a table called fact.

The user interface to Mondrian is a web app that looks like a spreadsheet. You navigate through the cube by slicing and dicing, drilling down, rolling up, and pivoting. The corresponding programming interface is written in an extension to SQL called MDX. The parts of SQL that are included in MDX are the select statement with both the “from” and the “where” clauses. The extensions include rows, columns, children, crossjoin, hierarchize, and union. Here is an example that shows monthly totals of crime per district.

```
select Hierarchize(Union(Crossjoin({[Measures].[crimes]},
{[Time].[2013]}), Crossjoin({[Measures].[crimes]},
[Time].[2013].Children))) ON COLUMNS,
Hierarchize(Union({[District].[All Districts]}, [District].[All
Districts].Children)) ON ROWS
from [sfcrime] where [Category].[All Categories]
```

This is what that query looks like.



	Measures			
	crimes			
	Time			
District	-2013	+0	+1	+2
-All Districts	26,668	10,300	9,437	6,931
BAYVIEW	2,638	1,034	897	707
CENTRAL	2,587	1,069	897	621
INGLESIDE	2,435	870	877	688
MISSION	3,361	1,240	1,270	851
NORTHERN	3,338	1,333	1,152	853
PARK	1,572	628	569	375
RICHMOND	1,454	533	548	373
SOUTHERN	4,957	1,944	1,686	1,327
TARAVAL	1,886	724	656	506
TENDERLOIN	2,440	925	885	630

Slicer: [(All)=All Categories]

Notice how the rows and columns designations describes how to lay out the data. How does Mondrian know what “All Districts” means? The answer to that lies in the sfcrime.xml file of which this is the relevant excerpt.

```
<Dimension name="District"
 foreignKey="district_id">
  <Hierarchy hasAll="true"
    primaryKey="district_id"
    allMemberName="All Districts"
    defaultMember="All Districts">
    <Table name="district"/>
    <Level name="name"
      column="name"
      uniqueMembers="true"/>
  </Hierarchy>
</Dimension>
```

It is the allMemberName attribute that identifies what phrase to use when identifying the root of a dimension. This excerpt shows how the district dimension is mapped to the district table in the mysql database.

This is actually a fairly common approach to big data. Use some combination of Hadoop pig, hive, and map/reduce to crunch the big data numbers into something small enough to be imported by an ETL suite into some OLAP cubes then use OLAP's spreadsheet style user interface to surface this summary information for your managers and analysts to explore.

I hope that you have benefited from this short little developer focused introduction to big data. What remains is references to some additional resources if you wish to explore this topic further.

https://data.sfgov.org/

The Open Data Portal is provided by the City and County of San Francisco to enhance open government, transparency, and accountability by improving access to data. The Open Data Portal is a one-stop destination for all approved City data that will help constituents make better use of information. This new ease of access will lead to innovation in how residents interact with government, resulting in social and economic benefits for the City.

http://hadoop.apache.org/

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-avaiability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-availabile service on top of a cluster of computers, each of which may be prone to failures.

http://hive.apache.org/

Hive is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems. Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language called HiveQL

http://www.r-project.org/

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

http://mondrian.pentaho.com/

Welcome to the community home for Pentaho Analysis Services Community Edition also known as Mondrian. Mondrian is an Online Analytical Processing (OLAP) server that enables business users to analyze large quantities of data in real-time. Users explore business data by drilling into and cross-tabulating information with speed-of-thought response times to complex analytical queries.

http://www.mysql.com/

MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout it's history. With its superior speed, reliability, and ease of use, MySQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications.

http://www.gnu.org/software/gawk/manual/gawk.html

This file documents awk, a program that you can use to select particular records in a file and perform operations upon them.