Very Large Data Management Lecture 1

Jim Leach

15 October 2015

A definition of big data?

Due to the hype of 'big data' there is a lot of uncertainty as to what it actually *is*. A good working definition is:

> Big data is any data that has become hard to really handle/process (with traditional methods).

There are many challenges associated with working with big data, including:

- Captutre;
- Curation;
- Storage;
- Search;
- Sharing;
- Transfer;
- Analysis; and
- Visualisation

The 3 V's:

1. Volume - data are continually increasing over time. Often more limited by disk I/O bandwidth as opposed to raw storage space.
2. Velocity - Any online process (both web-based and, for example, sensor data) generates data almost continually. The speed with which it arrives means that there is the potential for missed opportunities if decisions are not made in (near) real time.
3. Variety - data come in many forms: relational, text (unstructured), seme-structured (e.g. `XML`), graphs, streaming etc. The challenge is not processing just any one of these types, but in linking them all together to derive meaning.
4. An additional V is Veracity - there can be uncertainty in the data due to inconsistency, incompleteness, latency, ambiguities etc.

Data creation model

Previously, there were few producers of data, and many consumers. Now there are as many producers as there are consumers.

Cloud computing

Has many benefits:

- Cost & management - i.e. economies of scale, out-sourcing;
- Reduced time to deployment;
- Scalability;
- Reliability; and
- Sustainability - the hardware is not owned as does not need to be maintained.

Has several types: public cloud, private cloud, hybid (critical apps/data on private, non-critical public), cloud bursting (switching to cloud providers during peak load times).

Comprised of:

- Infrastructure as a Service (IaaS) - hardware related services e.g. storage or computational power (e.g. Amazon EC2, S3);
- Platform as a Service (PaaS) - cloud based development platforms, e.g. Google application engine, Microsoft Azure; and
- Software as a Service (SaaS) - complete software offering in the cloud, e.g. Salesforce CRM, Google Docs.

Key ingredients are: * Service-Oriented Architechture (SOA); * Utility computing (on demand computing); * Virtualisation; * SaaS; * PaaS; and * IaaS.

Depending on the service, different functionality and restrictions are provided or applied. E.g. Google App Engine has high level functionality (e.g. it scales automatically) but is more restrictive for developers. Amazon EC2 is incredibly simple with low level functionality ownly, but as a result can be more flexible.

Virtualisation

Often uses Hypervisor. The concept ensures that apps are isolated from each other and therefore do not crash the overall/underlying phyrical system if they fail.

Virtualisation is good for flexible resource management, esp. on CPU, reasonably well with RAM. Resource management of hard disk storage is still a challenge (esp. if apps want to read/write large amounts of data!)

Business Intelligence OLAP and OLTP

Relational databases

Formally given sets $D_1, D_2, ...D_n$, a **relation** r is defined as a subset such that $D_1 x D_2 x ... x D_n$, i.e. a relation is a set of n tuples. e.g.

```
customer_name = {Jones, Smith, Curry}
```

```
customer_street = {Main, North, Park}
```

Then:

```
r = { (Jones, Main), (Smith, North), (Curry, Park)}
```

i.e. `r` is a relation over customer_name and customer_street.

Relation Schema

If $A_1, A_2, ..., A_n$ are defined as *attributes* then $R = (A_1, A_2, ..., A_n)$ is a *relation schema.* e.g. `Customer_schema = (customer_name, customer_steet)`.

$r(R)$ is a relation on the relation schema R, e.g. `customer (Customer_schema)`.

To put this more plainly:

- Attributes = columns;
- Tuples = rows; and
- Relation = table.

It is therefore possible to say that a *database* is something that consists of multiple relations (tables). Information is broken up accross each relation (table) such that each relation stores one part of the information, e.g. the account relation, the depositor relation and the customer relation. Storing all information in one relation (table) is a poor choice as it would introduce repitition/duplication and/or the need for `NULL` values.

SQL sits on top of the database and operates using relational algebra. We use a *database management system (DBMS)* to handle the database (e.g. DB2, Oracle, SQL-Server etc).

OLTP vs. OLAP

OLTP (Online Transaction Processing)

- User facing;
- Real time;
- Low latency;
- Highly-concurrent;
- Many, small transactions, often of a "standard" nature;
- Read and write the database;
- E.g. e-commerce, online banking, transaction databases.

OLAP (Online Analytics Processing)

- Back end;
- Batch workloads;
- Low concurrency;
- Large, complex analytical queries, often ad-hoc;
- Often read data only;
- e.g. business intelligence, data mining etc.

OLTP with OLAP

To solve the issue of a single database using both OLTP and OLAP, it is normal to have separate databases for each process.

One OLTP database for user-facing, real time, high-volume applications and a second, separate OLAP database for data warehousing and large scale analytics.

The two are connected via an *ETL* process (Extract, Transform, Load).

Star Schema

Within the data warehouse, data are often organised by a **star schema**. In this set up there is:

1. A *fact* table: a large accumlation of facts, e.g. sales. Often insert-only; and
2. A set of *dimension* tables: smaller, static information that relates to the entities involved in the facts.