GREAT 2011 Summer School

C1: How to store a petabyte

By Matthew J. Graham (Caltech, VAO)

Overview

- Understanding the problem
- Data structures
- Transport protocols
- Large file systems
- Large databases

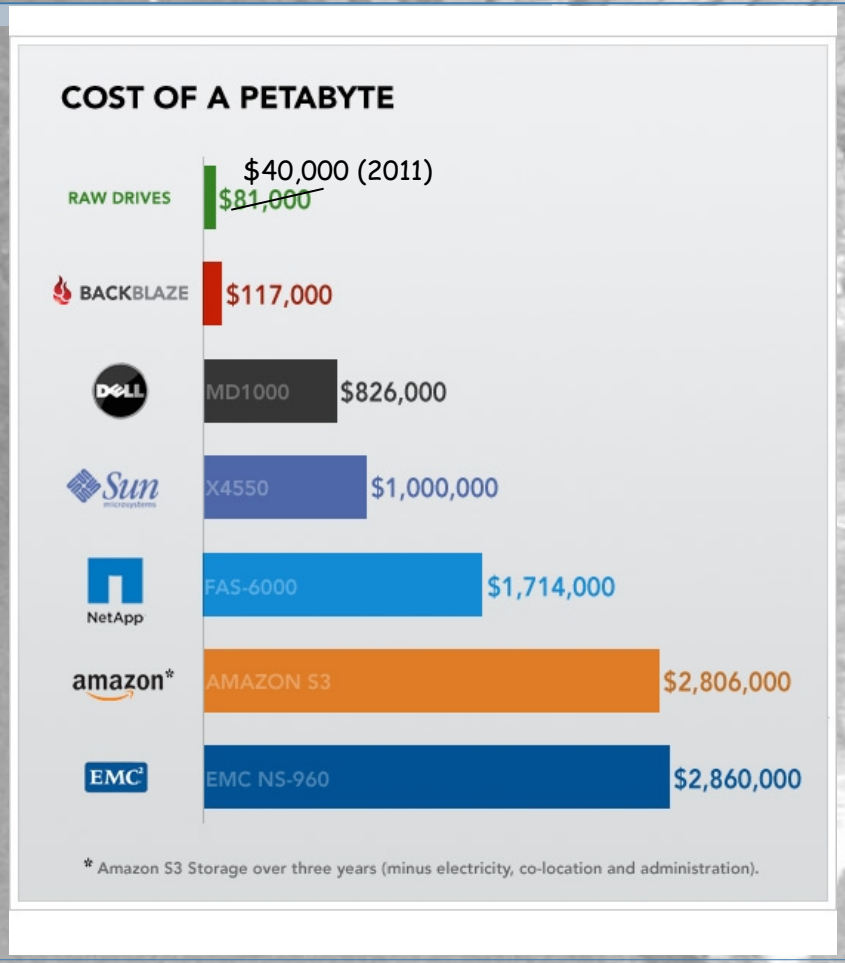
The lie of the land

- How much data and how frequently?
  - Continuously vs. burst mode
- What sort of data?
  - Images / binary data
  - Catalogs / textual data
    - Raw vs. structured
- What sort of storage model?
  - Write once, read many
  - Frequent writes/updates/appends
- What sort of access?
  - High throughput vs. availability
  - Sequential (processing) vs. random (querying)
  - Immediate access to new data?
- How much power do you have?

What the experts say…

- "Bring the computation to the data" – anon.
- "Just store the original data; processing, etc. just adds bloat" – David Hogg
- "Databases own the sweet spot between 1GB and 100 TB" – Jim Gray
- "Current problems not on Google scale yet: 300TB is really hard" – Alex Szalay
- "Extreme computing is about tradeoffs" – Stu Feldman (Google)

The cost of a petabyte



backblaze.com
Aug 2009

Data structures

- Binary
  - with separable description (header):
    - FITS (tile compression)
    - HDF5
  - with common data model and API:
    - CDF / NetCDF
- Text:
  - XML (VOTable) / JSON
  - Structure description (IDL) + binary data representation:
    - Google Protocol Buffers
    - Apache Avro
- Archive format:
  - Sequence files : collection + index

Textual comparison

XML:

```
<Object>
    <ID>Sirius</ID>
    <Type>Star</Type>
    <RA>101.28</RA>
    <Dec>-16.72</Dec>
    <Mag>-1.46</Mag>
</Object>
```

JSON:

```
{
    "ID": "Sirius",
    "Type": "Star",
    "RA": 101.28,
    "Dec": -16.72,
    "Mag": -1.46
}
```

Protocol Buffer:

```
message Object {
    required string id = 1,
    required string type = 2,
    required float ra = 3,
    required float dec = 4,
    optional float mag = 5
}
```

Avro:

```
{   "type": "record", "name": "Object",
    "fields": [{"name":"ID", "type":"string"},
        {"name":"Type", "type":"string"},
        {"name":"RA", "type":"float"},
        {"name":"Dec", "type":"float"},
        {"name":"Mag", "type":"float"} ]}
```

VOEvent

```
<voe:VOEvent  ivorn="ivo://raptor.lanl/VOEvent#235649409”  role="observation"
   version="2.0” xmlns:voe=http://www.ivoa.net/xml/VOEvent/v2.0 >
  <Who>
    <AuthorIVORN>ivo://raptor.lanl/organization</AuthorIVORN>
    <Date>2005-04-15T14:34:16</Date>
  </Who>
  <What>
    <Description>An imaginary event report about SN 2009lw.</Description>
    <Reference uri="http://raptor.lanl.gov/data/lightcurves/235649409"
       mimetype="application/x-votable+xml"
       meaning="http://www.ivoa.net/rdf/IVOAT#LightCurve">
    <Param name="seeing" value="2" unit="arcsec" ucd="instr.obsty.site.seeing"/>
    <Group name="magnitude">
      <Description>Time is days since the ref time in the WhereWhen section</
            Description>
      <Param name="time"    value="278.02" unit="day" ucd="time.epoch" />
      <Param name="mag"     value="19.5"   unit="mag" ucd="phot.mag"/>
      <Param name="magerr" value="0.14"    unit="mag" ucd="phot.mag; stat.err"/>
    </Group>
    <Table>
      <Param name="telescope" value="various" utype="whatever"/>
      <Description>Individual Moduli and Distances for NGC 0931 from NED</Description>
      <Field name="(m-M)"     unit="mag" ucd="phot.mag.distMod"/>
      <Field name="err(m-M)" unit="mag" ucd="phot.mag.distMod;stat.err"/>
      <Field name="D"         unit="Mpc" ucd="pos.distance"/>
      <Data>
        <TR><TD>33.16</TD><TD>0.38</TD><TD>42.9</TD></TR>
        <TR><TD>33.32</TD><TD>0.38</TD><TD>46.1</TD></TR>
        <TR><TD>33.51</TD><TD>0.48</TD><TD>50.4</TD></TR>
        <TR><TD>33.55</TD><TD>0.38</TD><TD>51.3</TD></TR>
        <TR><TD>33.71</TD><TD>0.43</TD><TD>55.2</TD></TR>
        <TR><TD>34.01</TD><TD>0.80</TD><TD>63.3</TD>/TR>
      </Data>
    </Table>
  </What>
```

```
  <WhereWhen id="Raptor-2455100">
    <ObsDataLocation>
      <ObservatoryLocation id="RAPTOR"/>
      <ObservationLocation>
        <AstroCoordSystem id="UTC-ICRS-TOPO"/>
        <AstroCoords coord_system_id="UTC-ICRS-TOPO">
          <Time>
            <TimeInstant>
              <ISOTime>2009-09-25T12:00:00</ISOTime>
            </TimeInstant>
            <Error>0.0</Error>
          </Time>
          <Position2D unit="deg">
            <Value2>
              <C1>37.0603169</C1> <!-- RA  -->
              <C2>31.3116578</C2> <!-- Dec -->
            </Value2>
            <Error2Radius>0.03</Error2Radius>
          </Position2D>
        </AstroCoords>
      </ObservationLocation>
    </ObsDataLocation>
  </WhereWhen>
  <Citations>
    <EventIVORN cite="followup">ivo://raptor.lanl/VOEvent#235649408</EventIVORN>
  </Citations>
  <Why>
    <Concept>process.variation.burst;em.opt</Concept>
    <Description>Looks like a SN</Description>
    <Inference relation="associated" probability="0.99">
      <Name>NGC0931</Name>
    </Inference>
  </Why>
</voe:VOEvent>
```

HDFS

- Inspired by Google FS
- Distributed, scalable, portable
- Rack (location (network switch)) aware
- Variety of backends:
  - local fs, remote cluster, cloud (S3)
- Architecture:
  - Cluster of data nodes with a master name node
  - Each data node serves blocks of data (~64 MB)
  - Data replicated across multiple hosts (default is 3 times: two same rack, one different)

HDFS interfaces

- Java API, Thrift, FUSE, WebDAV
- Command-line tool as part of Hadoop
  - Hadoop config file in /usr/local/hadoop/conf

> hadoop fs –mkdir input

> hadoop fs –put mydata input/

> hadoop fs –ls input

> hadoop fs –cat input

> hadoop fs –get input myresults

Alternates to HDFS

- OpenStack Object Storage ("Swift")
  - No single name node
  - Store any sized file
  - Write many times
- iRODs
  - Provides logical mappings for digital entities
  - Rule-based adaptive middleware allowing customization:
    - All data in a particular directory cannot be deleted
    - Additional access control checks for sensitive data

Transferring data

- Sneakernet is a fast bespoke solution
- Internet2 will allow advanced capabilities such as on-demand creation and scheduling of high-bandwidth high-performance data circuits
- Conventional transfers do not maximize bandwidth
- Parallel streams:
  - GridFTP
- TCP not great with long-distance, high bandwidth or multiple flows so:
  - Fine tune TCP with large bandwidth-delay product
  - Use a TCP variant: SACK
  - Use UDP instead: UDT
  - Use a new protocol: SCTP

The problem with RDBMS

- Too many reads:
  - add memcached to cache common queries -> reads not strictly ACID, cached data must expire
- Too many writes:
  - scale vertically with beefed up hardware -> costly
- Too many joins:
  - denormalize data to reduce joins
- Server swamped:
  - stop any server-side computations
- Some queries still slow:
  - prematerialize most complex queries, stop joining in most cases
- Writes getting ever slower:
  - drop secondary indexes + triggers

NoSQL

"select fun, profit from real_world where relational = false"

- Structured storage
- modern RDBMS show poor performance on certain data-intensive applications:
  - indexing large no. of documents
  - serving pages on high-traffic websites
  - delivering streaming media
- RDBMS are tuned for small but frequent read/write transactions or large batch transactions with rare write accesses
- real world deployments:
  - Digg
  - Facebook (50 TB)
  - eBay (2 PB)
- middleware layers can be added to provide RDBMS-type functionality (ACID guarantees)

Types of NoSQL

- Document store (XML databases)
- Graph (superset of triple store)
- Key-value store (Cassandra, Dynamo, Project Voldemort, Velocity, Keyspace?)
- Object database (Objectivity, Versant)
- Tabular (BigTable, HBase, Hypertable)
- Tuple store (Apache River)
- Multivalue databases

Column orientation

- Databases store their data as a series of 1-dimensional structures (normally rows)

- Faster seek times, aggregate operations
- Slows writing, accelerates reading
- Can aid compression – column data is all of same data type
- Note that R uses column-oriented data structures

HBase (hbase.apache.org)

- Distributed column-oriented "database" built on top of HDFS
- Sparse, distributed, persistent, multidimensional sorted map
- Suitable for real-time read/write random access
- Java API and REST interface (Stargate)

```
> hbase shell
create 'events', 'where', 'why'
put 'events', 'event1', 'where:ra', '123,45'
put 'events', 'event2', 'where:dec', '-16.25'
get 'events', 'event3', {COLUMN => 'why:concept'}
-> SNe
```

SciDB

- Column-oriented db designed specifically for scientific data including astronomy
- Use (immutable) arrays as first-class objects rather than tables
- Maintains ACID
- AQL and AFL:

```
CREATE ARRAY pixels <flux:double> [ID=0:999,1000,0,
    X=0:255,256,0, Y=0:255,256,0]
CREATE ARRAY dark <flux:double> [ID=0:999,1000,0,
    X=0:255,256,0, Y=0:255,256,0]
SELECT pixels.ID, pixels.x, pixels.y, pixels.flux – dark.flux
    FROM pixels, dark
```

VOSpace

- Lightweight layer on top of networked storage
- Highly agnostic:
  - Backend implementation
  - Transport protocol
  - Data format
- Arbitrary metadata
- Expose third-party capabilities