BugSense - problem definition & solution

We wanted to be able to:

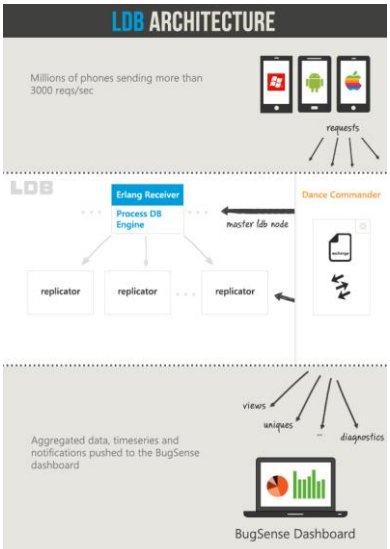
- *Abstract the application logic and feed browsers with JSON*
- *Run complex algorithms on the fly*
- *Experiment with data, without the need of a dedicated Hadoop cluster*
- *Pre-process data and then store them (cutting down storage)*
- *Be able to handle more than 1000 concurrent requests on every node*
- *Make "joins" in more than 125M rows per app*
- *Do this without spending a fortune in servers*

The solution uses:

- *Less than 20 large instances running on Azure*
- *An in-memory database*
- *A full blown custom LISP language written in C to implement queries, which is many times faster than having a VM (with a garbage collector) online all the time*
- *Erlang for communication between nodes*
- *Modified TCP_TIMEWAIT_LEN for an astonishing drop of 40K connections, saving on CPU, memory and TCP buffers*

bugsense

The architecture



When a request comes from a mobile device, the main LDB node, accepts the connection (using a pool of Erlang threads) and forwards the data to a specific DB. This request handling mechanism is implemented with fewer than 20 lines of Erlang code. Another reason we chose Erlang for communication between nodes.

When the request is "streamed" to LDB, a file called "process.lql" is responsible for analyzing, tokenizing the data and creating any counters. All this is done on the fly and for every request.

We are able to do this, because starting our LISP-VM and doing all these processes on every request, is still many times faster than having a VM (with a garbage collector) online all the time.

With LDB we can create time series and aggregated data with no more than 3 lines of code

John Vlachoyiannis (former CTO of Bugsense); www.linkedin.com/in/johnvlachoyiannis

bugsense

Healthcare

- About 80% of medical data is unstructured and clinically relevant
- Data (lots of data) resides in multiple places in multiple disparate formats
  - Individual Electronic Medical Records
  - Laboratory and imaging systems
  - Physician notes
  - Medical correspondence
  - Insurance claims
  - Research data
  - …
- Leveraging Big Data can provide numerous benefits
  - Build sustainable healthcare systems
  - Collaborate to improve care and outcomes
  - Increase access to healthcare

25

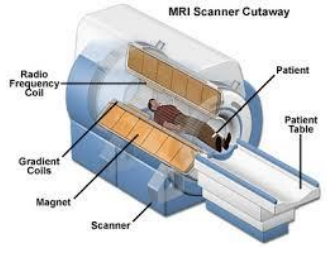
Using Analytics on Medical Imaging data

The Problem

There are a couple issues being faced by the medical industry when it comes to imaging, and they are:

1. There is no national standardisation in the UK when it comes to imaging techniques and processes. Individual trusts and imaging specialists are left to their own devices when defining standards.
2. There is no coordinated process in the UK to share imaging data or to analyse that data (in conjunction with other data sets) to the benefit of the UK population.

This is despite the fact that every modern imaging device (MRI Scanner, X-Ray Scanner, CT Scanner, …) is required to produce significant data per scan for auditing purposes.



The Solution

Working with an established imaging data auditing company, OCF (http://www.ocf.o.uk) are looking to create a Big Data environment, using Hadoop, to store data received from individual scanners. Initially this will be deployed at a local level, followed by a regionalisation, and then finally a nationwide roll-out.. In addition to the imaging data OCF are proposing to capture environmental data, weather data, and other medical data (such as outbreaks of disease or hot spots for a particular condition).

Once the data has been translated into a Hadoop environment the plan is to then run a series of algorithms to find insights and determine trends.

Result

It is anticipated that there will be a double benefit outcome for this solution:

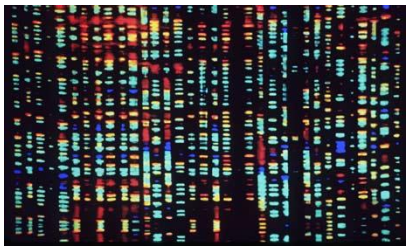
1. The ability to standardise imaging procedures across the UK using best-case scenarios and treatment results (this will also have a cost saving element).
2. The ability to be able to determine causality for a number of medical conditions based on medical and environmental data.

26

Big Data and the genome

The Problem

A major UK research university is in the process of creating a Centre for Computational Research for its Bioscience department. The department has links to other research facilities in the world and they are looking to create an environment where data can be shared across institutions. In addition, one of the researchers wants to catalogue every known genome in the environment. A significant amount of data will be produced (1 genome sequencer outputs 2.4 terabytes of raw data a week), and it needs to be managed and made available for analysis by researchers all over the world.



The Solution

An environment is going to be created using a Hadoop cluster to store the imaging data from the research centre and also medical data from the associated medical centre. In addition, IBM Data Explorer software will be supplied to allow researchers to have a unified view of all the enterprise data, including the Hadoop cluster and all other appropriate data sources within the university .

In a break from tradition the Hadoop data will be centralised rather than distributed as the university IT team want to ensure that data management standards are being adhered too especially in the realm of backup and archiving.

Result

Researchers, both inside and outside the university, will have centralised access to both the critical data from their research and just as importantly to other university (and outside) data that was previously too difficult or even impossible to obtain.

27

Predictive maintenance in the manufacturing industry

The Problem

A major UK discrete manufacturing company making capital items spends a significant amount of money (in the £billions) in maintenance costs on its product. Every major item shipped from this company is full of telemetric sensors; however this data is only analysed in real-time and isn't used to build historical models. The manufacturer is looking for ways to use this information to reduce its costs.

The Solution

The solution here is to deploy a Predictive Maintenance system, collecting all the telemetric data plus other sources of structured and unstructured data and store then in a Big Data environment. The data can then be used to build models, which can then be analysed using Predictive Analytics to develop insights and spot trends.

Result

The results obtained from the above solution can then be deployed to reduce materials and resource costs by having the correct resources and materials needed when they are required (JIT). In addition, customer relationships can be enhanced as the supplier can now inform the customer of potential issues well before they occur, allowing for appropriate planning and resources to be put in place.

28

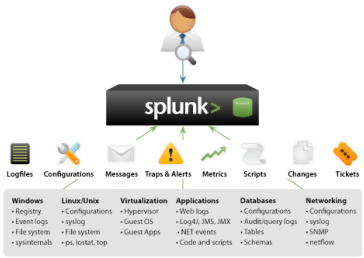
Predictive maintenance in the manufacturing industry

According to a recent (July 2014) consumer poll by Harris (Nielsen), only 14% of US car owners are familiar with connected cars, while 42% have heard of it but don't really know what they do. Only 15% of car owners say they are very/extremely interested in owning a Connected Car, while another 31% say they are not at all interested.

- Connected Cars is a Big Data problem
  - Car sensors typically produce around 2GB of data every 90 minutes
  - Roughly 60M cars are manufactured each year
  - If all those cars are connected and are driven for 2 hours a day, that equates around 50 Exabyte's of data being generated per day
  - Will this become reality? I would not be surprised
    - The Internet of Things is upon us

Log file analysis

- Splunk (http://www.splunk.com)
  - Used by some OpenVMS customers to monitor logs
  - Acquired BugSense, September 2013



- Loggly (https://www.loggly.com/)

- LogStash and ElasticSearch (http://logstash.net/, http://www.elasticsearch.org/)
  - Java-based
  - Will run on OpenVMS (I've run it)

AGENDA

- Introduction
- What is Big Data all about?
- Some case studies
- **Technologies**
- Where can OpenVMS play?
- Summary/conclusions
- Questions

Technologies

Many of the key technologies have come out of places like Google, Amazon. LinkedIn, Facebook, Yahoo, and Twitter, who by their own making have had to deal with problems of unprecedented scale. Their solutions to these problems are now finding more wide-spread applicability. Many of these solutions have been placed into the Open Source domain.

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform for the generation and processing of data used by our service as well as research and development efforts that require large data sets. The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed by hundreds of clients.

In this paper, we present file system interface extensions designed to support distributed applications, discuss many aspects of our design, and report measurements from both micro-benchmarks and real world use.

Categories and Subject Descriptors

D [4]: 3—*Distributed file systems*

General Terms

Design, reliability, performance, measurement

Keywords

Fault tolerance, scalability, data storage, clustered storage

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive commodity parts and is accessed by a comparable number of client machines. The quantity and quality of the components virtually guarantee that some are not functional at any given time and some will not recover from their current failures. We have seen problems caused by application bugs, operating system bugs, human errors, and the failures of disks, memory, connectors, networking, and power supplies. Therefore, constant monitoring, error detection, fault tolerance, and automatic recovery must be integral to the system.

Second, files are huge by traditional standards. Multi-GB files are common. Each file typically contains many application objects such as web documents. When we are regularly working with fast growing data sets of many TBs comprising billions of objects, it is unwieldy to manage billions of approximately KB-sized files even when the file system can support it. As a result, design assumptions and parameters such as I/O operation and block sizes have to be revisited.

Third, most files are mutated by appending new data rather than overwriting existing data. Random writes within a file are practically non-existent. Once written, the files are only read, and often only sequentially. A variety of data share these characteristics. In this environment, appending becomes the focus of performance optimization and atomicity guarantees, while caching data blocks in the client loses its appeal.

Fourth, co-designing the applications and the file system API benefits the overall system by increasing our flexibility.

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

1 Introduction

Over the past five years, the authors and many others at Google have implemented hundreds of special-purpose computations that process large amounts of raw data, such as crawled documents, web request logs, etc., to compute various kinds of derived data, such as inverted indices, various representations of the graph structure of web documents, summaries of the number of pages crawled per host, the set of most frequent queries in a given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a *map* operation to each logical "record" in our input in order to compute a set of intermediate key/value pairs, and then applying a *reduce* operation to all the values that shared the same key, in order to combine the derived data appropriately. Our use of a functional model with user-specified map and reduce operations allows us to parallelize large computations easily and to use re-execution as the primary mechanism for fault tolerance.

The major contributions of this work are a simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs.

Section 2 describes the basic programming model and gives several examples. Section 3 describes an implementation of the MapReduce interface tailored towards our cluster-based computing environment. Section 4 describes several refinements of the programming model that we have found useful. Section 5 has performance measurements of our implementation for a variety of tasks. Section 6 explores the use of MapReduce within Google including our experiences in using it as the basis

To appear in OSDI 2004

Note that these seminal papers are now over 10 years old.

Technologies

- MapReduce
- Hadoop
  - An ecosystem of software packages, including MapReduce, HDFS, and a whole bunch of other stuff
  - More on MapReduce and Hadoop later
- NoSQL databases and caching
  - MongoDB
  - HBase (also part of the Hadoop ecosystem)
  - Cassandra
  - Riak (http://www.basho.com)
  - CouchDB
  - Redis
  - …
- NewSQL databases
  - NuoDB (Jim Starkey founder, strategic advisor)
  - VoltDB (Michael Stonebraker co-founder, advisor)
- Array databases
  - KDB
    - Uses the K array processing language developed by Arthur Whitney and commercialized by Kx Systems
      - See http://kx.com/ and http://kx.com/kdb-plus.php
    - "*KDB was a deal breaker for the new FX guys at UBS (business guys that is), they basically said they would not be able to make money without it…*"

33

Technologies

- Data streaming and streaming analytics
  - Kafka (LinkedIn)
  - Storm (Twitter)
  - Apache Spark
    - Ubix.io (http://ubix.io/) and others are actively developing Big Data streaming tools based around Spark
    - More about Spark later
  - Apache Drill (http://incubator.apache.org/drill/)
  - …
- Reliable distributed coordination
  - Apache ZooKeeper (http://zookeeper.apache.org/)
- Analysis/query tools and techniques
  - R (http://www.r-project.org/)
  - Drill (based on Google Dremmel)
  - Impala (Cloudera)
  - Big Data, Bayesian Inference, and Monte Carlo methods
    - See for example http://www.bayesian-inference.com/softwarearticlesbigdata and http://research.google.com/pubs/pub41849.html
      - Watch this space
  - Numerous others (this is a big growth area)
  - Interestingly most of the relevant algorithms have existed for a long time, but their application was resource constrained… until now

34

Technologies

Some key programming languages

- There are a number of programming languages emerging as the best options for the development of Big Data solutions
  - Erlang
  - Scala
  - Go (Google)
    - See for example InfluxDB (http://influxdb.com/)
  - Rust (relatively new on the scene)
  - Java (well, the JVM at least)
  - Lisp-based languages such as Clojure
  - JavaScript and Python
    - As query languages (in place of traditional SQL)

- Interestingly these are mostly functional languages
  - Languages geared towards concurrency

MapReduce and Hadoop

- MapReduce
  - A framework for processing parallelizable problems across huge datasets using a large number of compute nodes (a cluster)
  - Divide and conquer algorithm
  - Inspired by the map and reduce functions commonly used in functional programming
  - Map step:
    - The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes
    - Worker nodes may in turn further divide the sub-problems
    - Worker nodes processes the smaller problems, and pass the answers back to their master node
  - Reduce step
    - The master node collects the answers to all the sub-problems and combines them as required in order to output the solution

- MapReduce frameworks such as Apache Hadoop facilitate the required orchestration
  - Marshalling the distributed servers
  - Running the various tasks in parallel
  - Coordinating communication and data transfers between the various parts of the system
  - Redundancy and fault tolerance

MapReduce and Hadoop

MapReduce allows for distributed processing of the map and reduction operations. Provided that each mapping operation is independent of the others, all maps can be performed in parallel – though in practice this is limited by the number of independent data sources and/or the number of CPUs near each source. Similarly, a set of 'reducers' can perform the reduction phase, provided that all outputs of the map operation that share the same key are presented to the same reducer at the same time, or that the reduction function is associative. While this process can often appear inefficient compared to algorithms that are more sequential, MapReduce can be applied to significantly larger datasets than "commodity" servers can handle – a large server farm can use MapReduce to sort a petabyte of data in only a few hours. The parallelism also offers some possibility of recovering from partial failure of servers or storage during the operation: if one mapper or reducer fails, the work can be rescheduled – assuming the input data is still available.

http://en.wikipedia.org/wiki/MapReduce

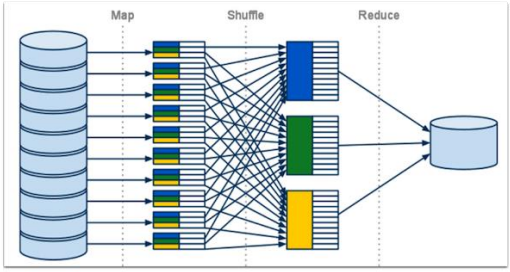
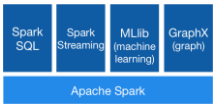


Image adapted from http://aimotion.blogspot.co.nz/2012/08/introduction-to-recommendations-with.html

Apache Spark

- http://spark.apache.org/
- A fast and general-purpose engine for large-scale data processing
- Seen as a replacement for MapReduce by some people
  - Gaining in popularity and adoption
- Up to 100 times faster than Hadoop MapReduce in memory; 10 times faster on disk
- Developers can write applications quickly in Java, Scala or Python
- Can combine SQL, streaming, and complex analytics
- Integrated with Hadoop
  - Spark can run on Hadoop 2's cluster manager, and can read any existing Hadoop data
- IBM are including Spark in the next beta release of BigInsights (IBM's Hadoop distribution)
- Cloudera (http://www.cloudera.com)
  - Enterprise support for Apache Hadoop
  - Spark is supported by Cloudera via Cloudera Enterprise 5
- HortonWorks (http://hortonworks.com/) supports Spark via its Hortonworks Data Platform
- Ubix.io are developing real-time analytics solutions and Big Data streaming tools based around Spark
- DataBricks (https://databricks.com/) recently secured $33M funding to further develop their next-generation Big Data platform
  - Databricks Cloud is a cloud platform built around Apache Spark



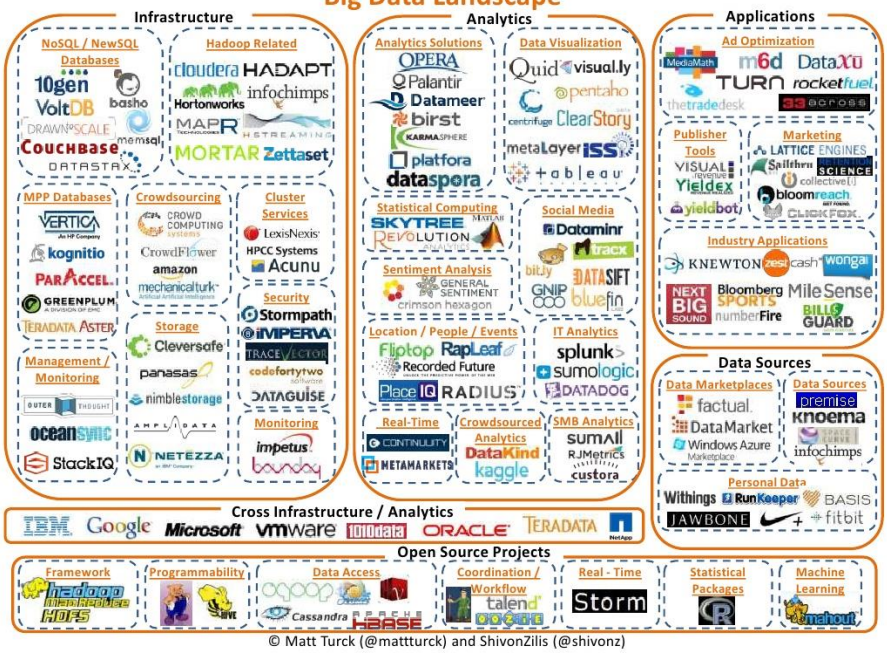
Big Data and NoSQL

- Traditional databases generally don't cut it in the Big Data world
- Big data requires:
  - Horizontal scalability (preferably near-linear)
  - High levels of concurrency
  - Fast read/write times
  - ...
- Enter NoSQL data stores
  - Different approaches to data storage, relationships, transactions, ...
    - Key/value stores
    - Column/tabular stores
    - Document stores (often with a preference for JSON or BSON documents)
    - Graph databases
    - Array databases

- Transactions in the traditional sense are generally quite slow
  - Many NoSQL databases implement what is referred to as "*eventual consistency*"
  - There are obviously tradeoffs...
- Many NoSQL data stores have a flat data structure
- Possibly no DB schema (in the traditional sense at least)

riak redis cassandra APACHE HBASE NUODB mongoDB CouchDB VoltDB

Big Data Landscape



AGENDA

- Introduction
- What is Big Data all about?
- Some case studies
- Technologies
- **Where can OpenVMS play?**
- Summary/conclusions
- Questions

Opportunities for OpenVMS

- Port Open Source solutions to OpenVMS
  - Most JVM-based Big Data technologies can in theory work on OpenVMS
    - Hbase, Kafka, Storm, Zookeeper, Logstash, …
    - In some cases this will require Java 7
    - Need to keep Java up to date on OpenVMS
  - Porting other solutions typically requires a bit more work, but is doable
- Adapt existing technologies or develop comparable solutions that take advantage of OpenVMS capabilities
  - Clustering and ICC
  - Distributed lock management, distributed file system
  - Security
  - Batch processing
  - …
- Client API's to facilitate interaction with Big Data solutions running on other platforms
- What about The Internet of Things (IoT)?
  - Not exactly Big Data, but related
  - OpenVMS on IP-enabled connected devises?
  - Provision of secure, reliable, and scalable services along the lines of https://xively.com/ and http://dweet.io/
  - Make support for IoT protocols such as MQTT an integral part of OpenVMS?

"Reading about Kafka makes me realize just how much many OpenVMS customers, or those managing their systems, are simply missing in their environments", John Apps, August 2014

Opportunities for OpenVMS

"Experts hack smart LED light bulbs "
http://www.bbc.co.uk/news/technology-28208905"
... wouldn't happen if that thing were running OpenVMS!

"Why we decided to move over eleven billion records from the legacy system"
http://jaihirsch.github.io/straw-in-a-haystack//mongodb/2014/01/06/mongodb-legacy-migration/
Cool, but wouldn't it be neat if we had stuff like MongoDB on OpenVMS?!

"NHS tears out its Oracle Spine in favour of open source; Health authority parachutes in Riak, Redis to save on dosh & screwups"
http://www.theregister.co.uk/2013/10/10/nhs_drops_oracle_for_riak/
Well, we already have most of Riak and Redis ported to OpenVMS...

AGENDA

- Introduction
- What is Big Data all about?
- Some case studies
- Technologies
- Where can OpenVMS play?
- **Summary/conclusions**
- Questions

Summary

"Data you can't process by traditional tools."

- Volume, variety, velocity
- Rapidly growing space filled with fascinating and impressive pieces of software technology
- Plenty of opportunities for OpenVMS to play
- "There Is Nothing New Under The Sun"

"A phenomenon defined by the rapid acceleration in the expanding volume of high velocity, complex and diverse types of data."

"... a collection of tools, techniques and technologies for working with data productively, at any scale."

AGENDA

- Introduction
- What is Big Data all about?
- Some case studies
- Technologies
- Where can OpenVMS play?
- Summary/conclusions
- **Questions**


Thank you