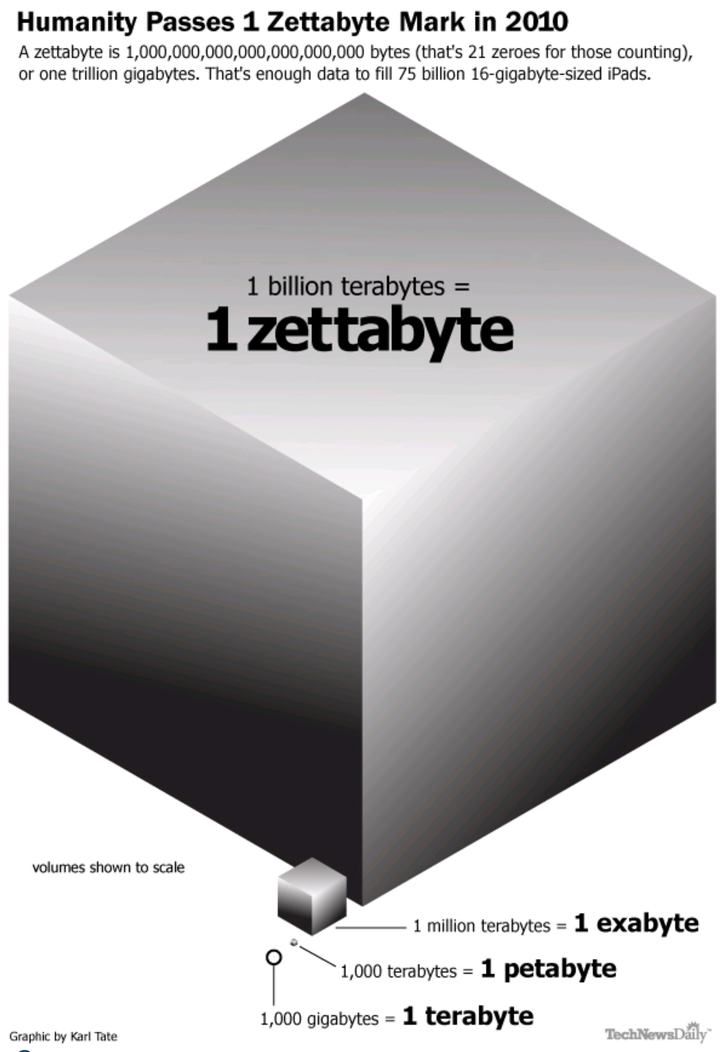
Entering the Zettabyte Age
Jeffrey Krone

1 Kilobyte	1,000 bits/byte
1 megabyte	1,000,000
1 gigabyte	1,000,000,000
1 terabyte	1,000,000,000,000
1 petabyte	1,000,000,000,000,000
1 exabyte	1,000,000,000,000,000,000
1 zettabyte	1,000,000,000,000,000,000,000

Presentation Outline

- The Evolution of Data
- Previous Data Solutions (Shard’ing, SANS)
- The Hadoop Ecosystem
- Hadoop Case Study – Financial Institution
- Founded Zettaset to address some of the deficiencies inherent in a Big Data Infrastructure such as Hadoop
- Utilizing Flash Drives with Hadoop
- Implications of utilizing Hadoop in a Cloud Computing Environment
- Open Problems to be addressed

Data volume is growing exponentially


Humanity Passes 1 Zettabyte Mark in 2010
A zettabyte is 1,000,000,000,000,000,000,000 bytes (that's 21 zeroes for those counting), or one trillion gigabytes. That's enough data to fill 75 billion 16-gigabyte-sized iPads.
1 billion terabytes =
1 zettabyte
volumes shown to scale
1 million terabytes = 1 exabyte
1,000 terabytes = 1 petabyte
1,000 gigabytes = 1 terabyte
Graphic by Karl Tate
TechNewsDaily

- Estimated Global Data Volume:
  - 2011: 1.8 ZB
  - 2015: 7.9 ZB
- The world's information doubles every two years
- Over the next 10 years:
  - The number of servers worldwide will grow by 10x
  - Amount of information managed by enterprise data centers will grow by 50x
  - Number of "files" enterprise data center handle will grow by 75x

Source: http://www.emc.com/leadership/programs/digital-universe.htm, which was based on the 2011 IDC Digital Universe Study

The Evolution of Data

- In the past, the most difficult problem for businesses was how to store all the data.
- The challenge now is no longer to store large amounts of information, but to understand and analyze this data.
- By harnessing this data through sophisticated analytics, and by presenting the key metrics in an efficient, easily discernable fashion, we are afforded unprecedented understanding and insight into our data.

The Evolution of Data

– Unlocking the true value of this massive amount of information will require new systems for centralizing, aggregating, analyzing, and visualizing these enormous data sets. In particular analyzing and understanding petabytes of structured and unstructured data poses the following unique challenges:
  - **Scalability**
  - **Robustness**
  - **Diversity**
  - **Analytics**
  - **Visualization of the Data**

Past Big Data Solutions

- **Data Shard'ing**
  - Is a "**shared nothing**" partitioning scheme for large databases across a number of servers increasing scalability of performance of traditional relational database systems. Essentially, you are breaking your database down into smaller chunks called "**shards**" and spreading them across a number of distributed servers. The advantages of Sharding is as follows:
    - **Easier to manage**
    - **Faster**
    - **Reduce Costs**

Past Big Data Solutions

- **Data Shard'ing Shortcomings:**
  - **Reliability**
  - **Distributed Queries**
  - **Writing Sharding Code is difficult**
  - **No automated way to to perform load balancing**
  - **Shards are not synchronously replicated**

Past Big Data Solutions

- **<u>SANS</u>**
  - SANS are essentially dedicated, high performance storage networks that transfer data between servers and storage devices, separate from the Local Area Network (usually through fiber channels).
  - **<u>ADVANTAGES</u>**
    - Ability to move large blocks of data
    - High level of performance and availability
    - Dynamically balances loads across the network.
  - **<u>DISADVANTAGES</u>**
    - Complex to manage a wide scope of devices
    - Lack of Standardization
    - SANs are very expensive

BIG Data / Apache Hadoop

- Apache Hadoop was developed to overcome the deficiencies mentioned previously of prior storage and analytics architectures (e.g. SANS, Sharding, Parallel Databases, BI Tools).

- The Apache Hadoop software library framework allows for distributed processing of large datasets across clusters of computers on commodity hardware. This solution is designed for flexibility and scalability, with an architecture that scales to thousands of servers and petabytes of data. The library detects and handles failures at the application layer, delivering a high-availability service on commodity hardware.

Hadoop

- Hadoop is a Platform which enables you to store and analyze large volumes of data.
- Hadoop is batch oriented (high throughput and low latency) and strongly consistent (data is always available).
- Hadoop is best utilized for:
  - Large scale batch analytics
  - Unstructured or semi-structured data
  - Flat files
- Hadoop is comprised of two major subsystems
  - HDFS (File System)
  - Map Reduce

Hadoop

- **HDFS**
  - Is a file system that supports large files
  - Files are broken into 64MB+ Blocks that are normally triple replicated.
  - **NameNode**
    - Is essentially the master meta data server.
    - The NameNode only persists metadata. It does not persist the location of each data node that hosts a block. The metadata is stored persistently on a local disk in the form of two files:
      - **Name Space Image File (FS Image)**
      - **Edit Log**

Hadoop

- **<u>HDFS</u>**
  - **<u>Secondary NameNode</u>**
    - Fetches the **FS Image** and the **Edit Log** and merges them together into a single file preventing the **Edit Log** from becoming too large.
    - Runs on a separate machine then the Primary NameNode
    - Maintains an out of date image of the merged Name Node image, which could be utilized if the Primary Name Node fails.
  - **<u>DateNode</u>**
    - The purpose of the DataNode is to retrieve blocks of data when it is told to do so by either the Clients and/or NameNode.
    - Stores all the raw blocks that represent the files being stored. Periodically reports back to the NameNode with lists of blocks it is storing.
  - **<u>NameNode – Determining Replica Placement</u>**
    - **How does the NameNode choose which DataNodes to store replica blocks on?**

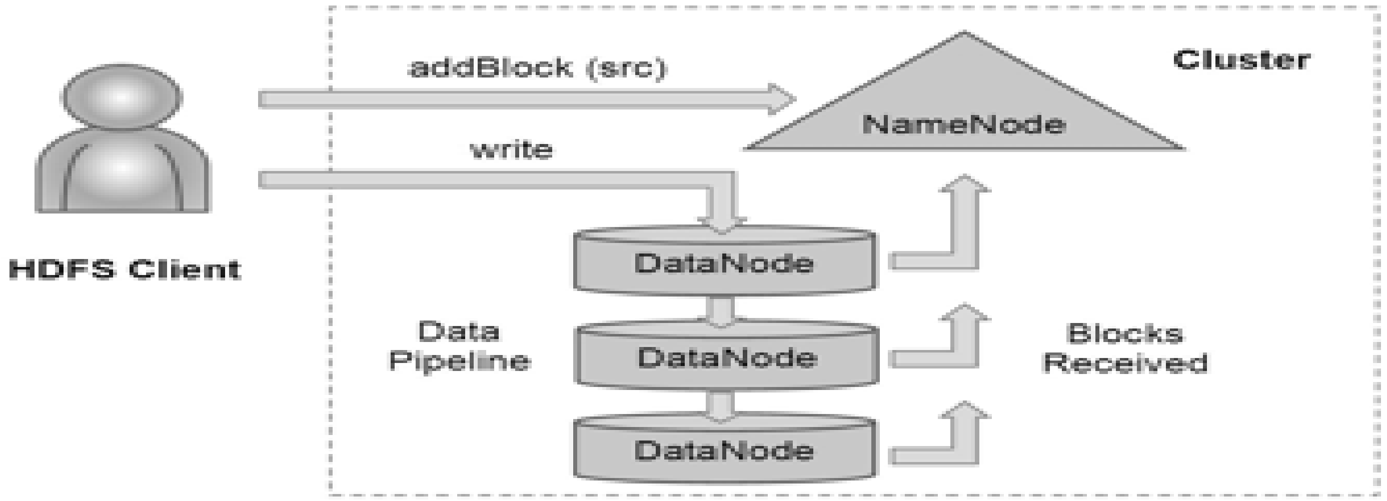
Zettaset

Hadoop

- **HDFS**
  - **File Reads (Process)**
    - Data Nodes are sorted by proximity to the Client (Application making the READ request)
    - Clients contact DataNodes directly to retrieve data and the NameNode simply guides the Client to the best datanode for each block of data being read.
    - If an error occurs while reading a block from a DataNode, then the NameNode will try the next closest DataNode to the Client in order to retrieve the block of data. DataNodes that fail are reported back to the NameNode.

Hadoop

- **HDFS**
  - **File Writes (Creating a New File)**



Hadoop / MapReduce

- Is a software framework for writing applications which process very large datasets (multi-terabyte data sets) in parallel on large clusters of machines.   Essentially enabling the user to run analytics across large blocks of data.

- The MapReduce Framework takes care of scheduling tasks, monitoring them, and re-executing failed tasks.

Hadoop / MapReduce

- **The Map Reduce Framework consists of a single master *JobTracker* and one slave *Task Tracker* per cluster node.**
  - **Job Tracker**
    - Coordinates all the jobs run on the system scheduling tasks to run on ***Task Trackers.*** If a Job fails then the ***Job Tracker*** can re-schedule it to another ***Task Tracker.***
    - Stores in-memory information about every running MapReduce Job
    - Assigns Tasks to machines in the cluster.
    - When a Job Tracker assigns a task to a machine, It will prioritize the task to machines with Data Locality.
  - **Task Tracker**
    - Runs Tasks and sends progress reports to the Job Tracker
    - Has a local directory to create a localized cache and localized job
    - Code is essentially moved to the data (Map Reduce Jars) instead of visa versa.   It is more efficient to move around small jar files then moving around data. Map Reduce Jars are sent to Task Trackers to run locally (i.e. machines where the data is local to the task).

Hadoop / MapReduce

- **MapReduce Example:**
  - Input a raw weather data file that is comma delimited and determine the maximum temperature in the dataset.
  - **MAPPER**
    - Assume the '**Key**s' of the input file are line offsets between each row of data
    - Our user defined Mapper Function simply extracts the '**Year**' and **'Temperature**' from each row of input data.
    - The Output of our Map Function is sorted before sending it to the Reduce function. Therefore, each key / value in the intermediate output (**year, temperature**) is **grouped by 'Year' and sorted by 'Temperature' within that year**.
  - **REDUCER**
    - The Reducer function takes the sorted Map(s) inputs and simply iterates through the list of temperatures per year and selects a maximum temperature for each year.
  - **Fault Tolerance**
    - What happens if Map or Reducer Tasks fail?

HBase

- HBase is a distributed Key / Value store built on top of Hadoop and is tightly integrated with the Hadoop MapReduce framework. HBase is an open source, distributed, column oriented database modeled after Google's BigTable.
- HBase shines with large amounts of data, and read/write concurrency.
- **Automatic Partitioning** – as your table grows, they will automatically be split into regions and distributed across all available nodes.
- Does not have indexes. Rows are stored sequentially, as are the columns written to each row.
- HBase makes Hadoop useable for real time streaming workloads which the Hadoop File System cannot handle itself.

OOZIE

- Is a workflow scheduler. It manages data processing jobs (e.g. load data, storing data, analyze data, cleaning data, running map reduce jobs, etc.) for Hadoop.
- Users create Directed Acyclical Graphs to model their workflow. Oozie at runtime manages the dependencies and execute the actions when the dependencies identified in DAG are satisfied.
- Supports all types of Hadoop jobs and is integrated with the Hadoop stack.
- Supports data and time triggers, users can specify execution frequency and can wait for data arrival to trigger an action in the workflow.

ZooKeeper

- Zookeeper is a stripped down filesystem that exposes a few simple operations and abstractions that enable you to build distributed queues, configuration service, distributed locks, and leader election among a group of peers.
  - Configuration Service – store and allows applications to retrieve or update configuration files
  - Distributed Lock – is a mechanism for providing mutual exclusion between a collection of processes. At any one time, only a single process may hold the lock. They can be utilized for leader election, where the leader is the process the holds the lock at any point of time.
- Zookeeper is highly available running across a collection of machines.

HIVE

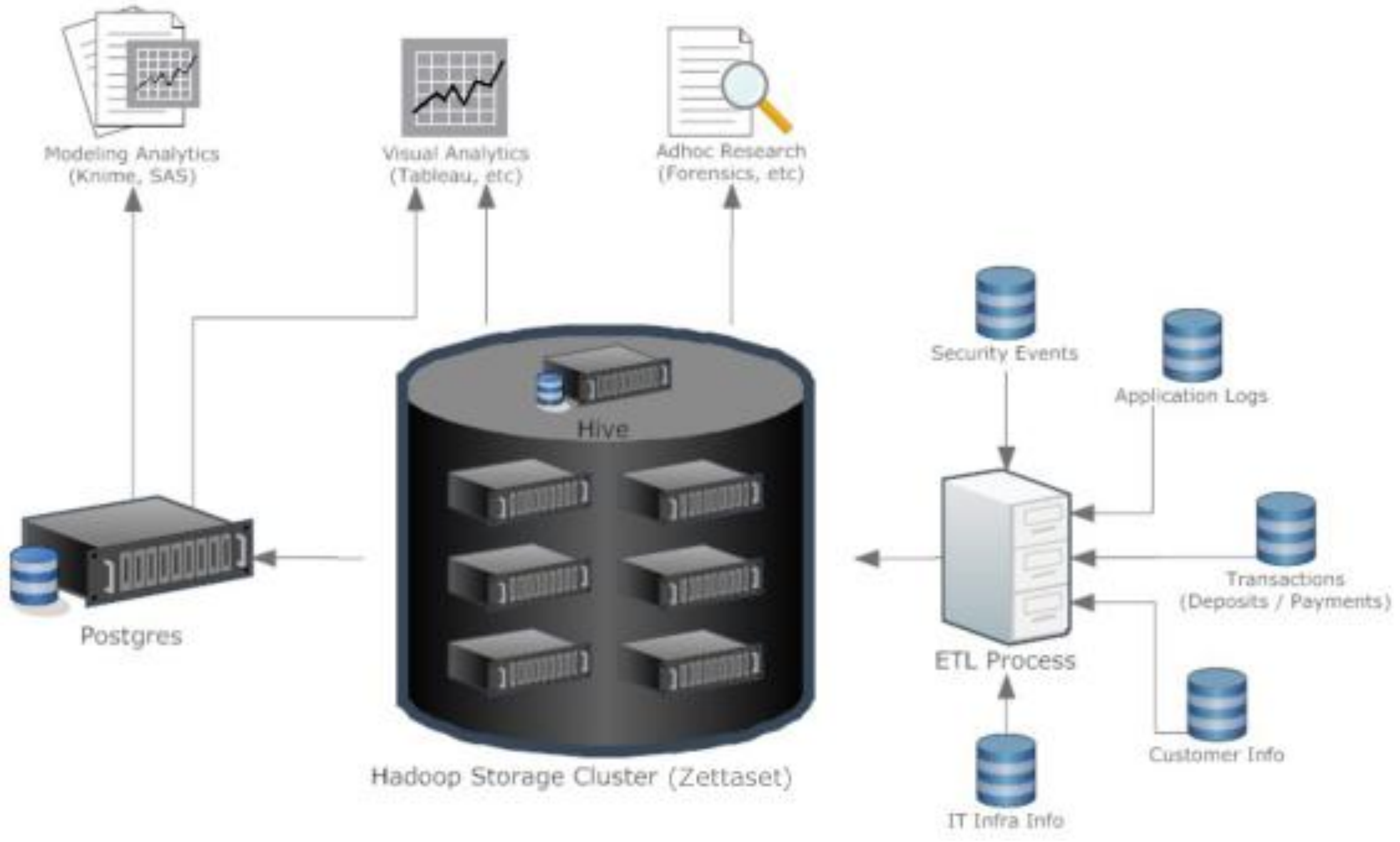
- Utilized by individuals with strong SQL Skills and limited programming ability.
- Compatible with existing Business Intelligence tools that utilize SQL and ODBC.
- Metastore – central depository of Hive Metadata.
  - It is comprised of a service and a backup store for the data.
  - Usually a standalone Database such as MySQL is utilized for the standalone Metastore.
- Partial support of SQL-92 specification

PIG

- Pig and Hive were written to insulate users from the complexities of writing MapReduce Programs. MapReduce requires users to write mappers and reducers, compilation of the code, submitting jobs and retrieving the results of the jobs. This is a very complex and time consuming process.
- A Pig program is made up up of a series of operations, or transformations that are applied to the input data to produce a desired output. The operations describe a data flow, which is converted into a series of MapReduce Programs.
- PIG is designed for batch processing of data. Pig is not designed to handle a small amount of data, since it has to scan the entire dataset.
- **PIG is comprised of two components:**
  - **The Language (Pig Latin) utilized to express data flows**
  - **The Execution Environment to run Pig Latin Programs.**

Security Intelligence Ecosystem

Putting it all together



Why I Decided to Start Zettaset?

- **<u>The Drivers behind starting Zettaset</u>**
  - – Realization of the need for a Big Data Platform
  - – Address some of the shortfalls of the Hadoop Ecosystem.
  - – Desire to build a turnkey, easily deployable, highly resilient, manageable, Enterprise data analytics platform which enables the aggregation and analysis of massive amounts of data.

How Were the Shortfalls of Hadoop Addressed?

– Building your own Hadoop infrastructure and integrating Hive, Pig, Hbase, Oozie, Zookeeper successfully would be a difficult endeavor. In this section, I will outline the complexities and potential pitfalls an organization would encounter if they attempted to successfully integrate Hadoop, Hive, Pig, Hbase, Oozie, and Zookeeper together into a single deployable platform in their organization.

– **The Topics that will be covered in this section include**:

  - Provisioning / Monitoring
  - Availability
  - Backup
  - HDFS
  - Updates to HBase, Pig, and Hive
  - Security
  - Import / Export
  - Scheduling

How Were the Shortfalls of Hadoop Addressed?

– **Provisioning / Monitoring**

- **ISSUES / SHORTFALLS THAT WERE ADDRESSED**
  - Provision all of your storage, server, and network services
  - Distributed logging
  - Optimize Configuration for your particular environment to maximize performance of the Cluster.
  - Configuration Management
  - Alerts (both global and per machine)
  - Service Migration
  - Performance Monitoring in the cluster
  - Automatic installation and provisioning of servers in order to scale to 100s to 1000s of servers.

How Were the Shortfalls of Hadoop Addressed?

– **Availability**
  - Name Node Failure
  - Job Tracker Failure
  - Hive Metastore Failure
  - Time Synchronization Issues
  - Substantial operational staff required due to the complexities of maintaining the environment and utilizing Hadoop.
  - Key Process failures
  - Failure Recovery / Self Healing ability

– **Backup**
  - No Backup tools exist for backing up a Hadoop Cluster of 1000s of TB of Data. Performing incremental or full backups at this scale is beyond the capability of existing backup tools.

How Were the Shortfalls of Hadoop Addressed?

- **File System**
  - While Hadoop can store petabytes of data, Hadoop only supports tens of millions of files, as all file locations and associated blocks are kept in RAM by the HDFS Name Node Master (single point of failure)
- **Writing Map Reduce Jobs**
  - Writing Map Reduce Jobs, no matter how trivial, typically requires programming in Java (or another language).
  - In Hadoop, no system exists for periodically scheduling MapReduce jobs -- you can only submit jobs for immediate execution.

How Were the Shortfalls of Hadoop Addressed?

– **HBase**

- HBase does not have a schema system. A User must determine all of their data encodings and map them to HBase column families and qualifiers.
- Helper Libraries in HBase for encoding your data types (e.g. change integer to byte array) have drawbacks. A default encoding for many data types does not exist (e.g. dates).
- There is no out of the box support for multi-column row keys. The row key is a single byte array. Your schema system must serialize a multi-column row key of whatever given data type into a single byte array to multi-column keys.

How Were the Shortfalls of Hadoop Addressed?

- **PIG**
  - No Multithreading capability
  - PIG only supports flat files
  - Pig has no user interface
- **Hive**
  - Hive has no user interface
- **Security**
  - The Hadoop ecosystem has only partially adopted Kerberos but many services remain unprotected and use trivial authentication systems.
- **Import / Export**
  - Hadoop has minimal support for reading, and writing different file types.

How Were the Shortfalls of Hadoop Addressed?

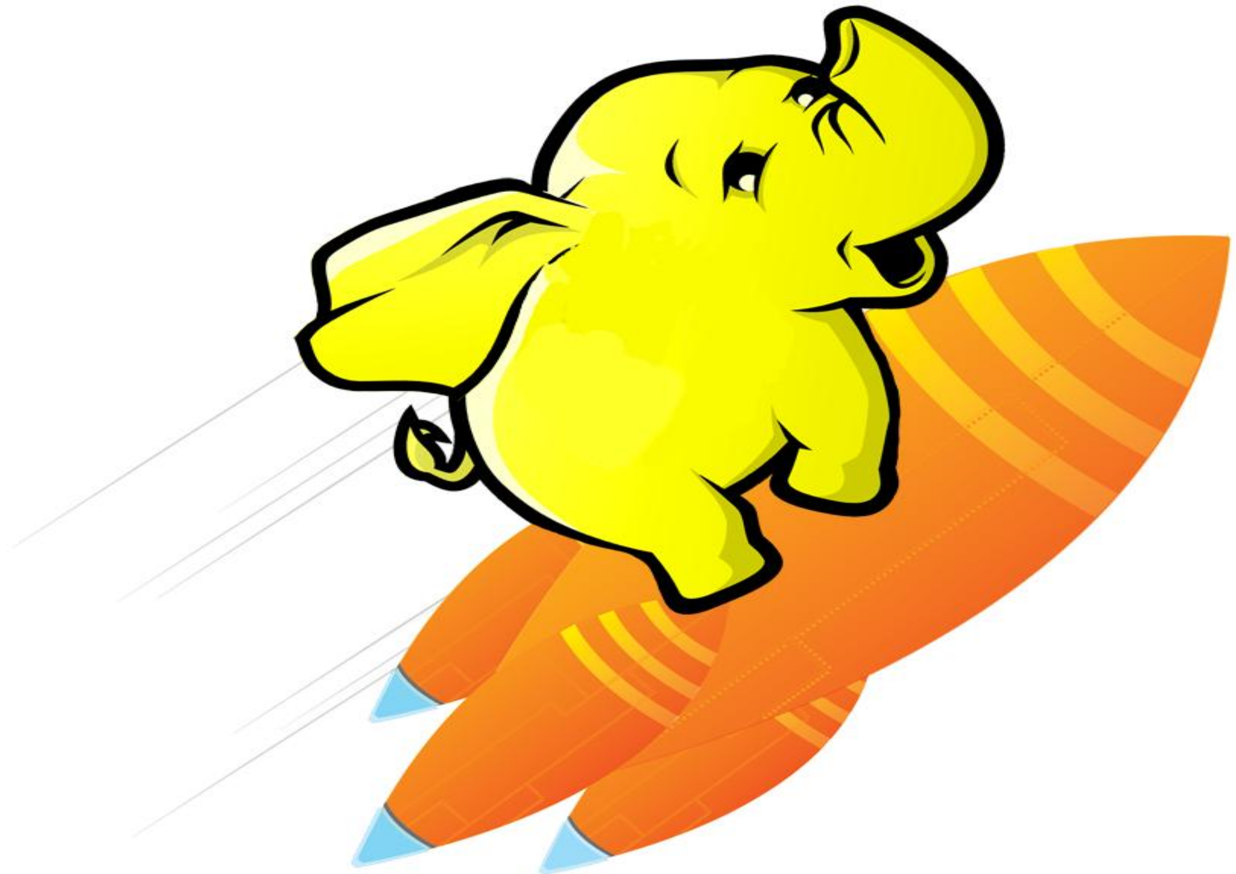
– **Scheduling**
  - Mapreduce comes with a primitive scheduling facility which enables a user to submit MapReduce Jobs and in some cases prioritize particular jobs over other Jobs. However, the Scheduler lacks the ability to assign work flow to a job, kill or pause a Running Job. The Scheduler has a primitive UI.

– **Hadoop does not have a UI**
  - Utilized to import / export Data
  - Explore the data residing within Hadoop / HDFS
  - Manipulate the data directly
  - To Monitor System / Node Health
  - Enable Security

Future Issues to Address

- **Future Enhancements**
  - HBase Coprocessor integration
  - Virtual machine deployment of Hadoop
  - Hybrid Cluster (optimize cluster with a mixture of Flash and SATA Drives)
  - Failover of non-hadoop Big Data Services (e.g. MongoDB)

FLASH IMPLEMENTATION



FLASH IMPLEMENTATION

- **Hardware**
  - **Testing Environment**
    - HP DL380 Servers (4 Full Size Flash Cards, 2 half size slots)
    - Data Size – 9.6 Terabytes Raw
    - 2 x Intel 5690 Processors (6 Cores w/hyper threading), 3.46 Ghz Processor, Smart Cache – 12MB)
    - 96GB RAM
    - 1 x 7124sx (Arista Switch – Layer 2/3/4, 10 Gigabit Ethernet Switch, 24 x 1/10 Gb ports, 500 nano second forwarding speed)
    - 4x10 GB SFP+ Network Cables (Patch Cables)
    - 2 x Solarflare 5122 NICs (Dual Port – 10GB Ethernet Adapter)
    - 8 x 1.2TB PCI Flash Drives

FLASH PERFORMANCE

FlashScale BD Terasort Performance Matrix*

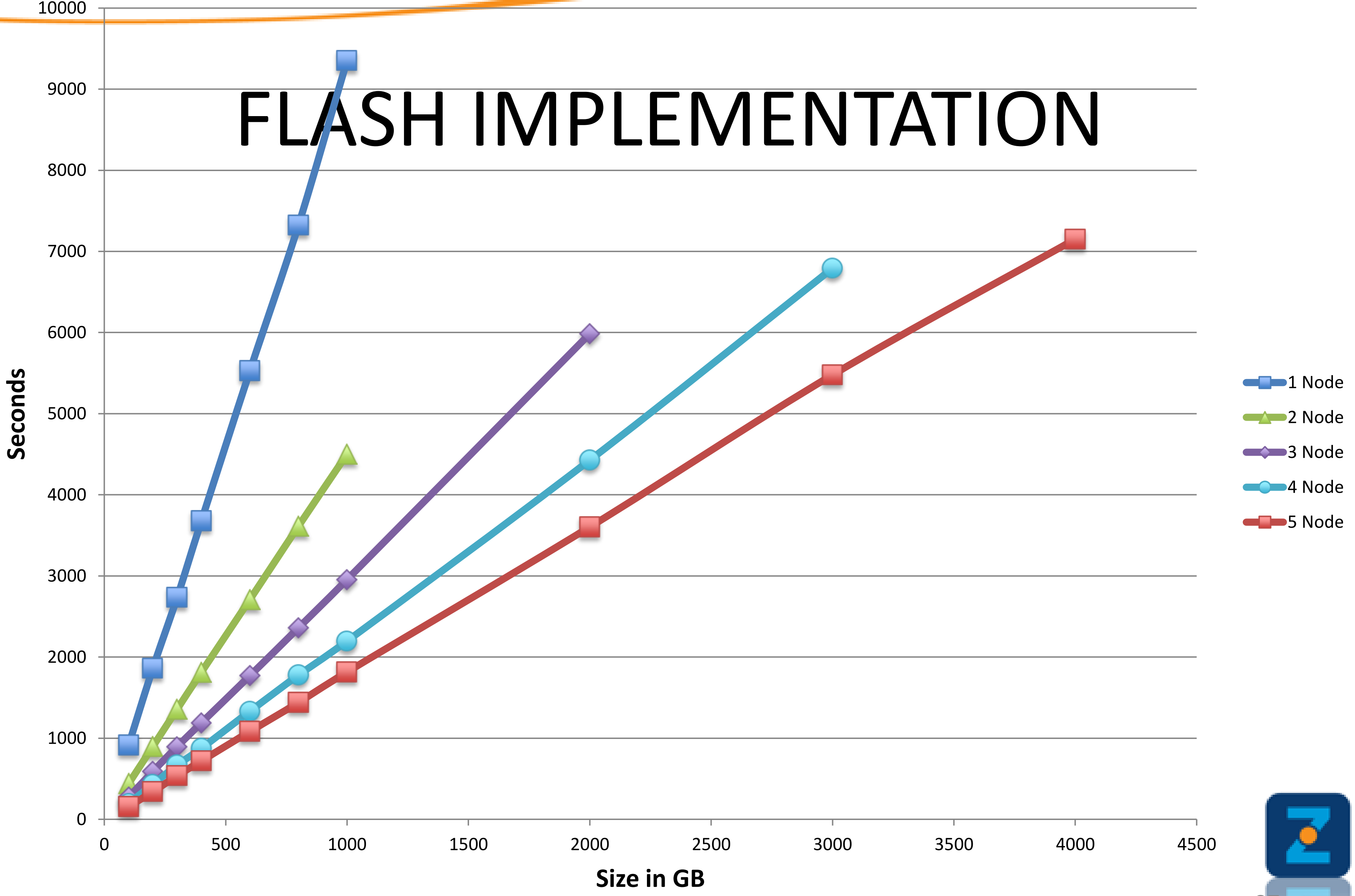
Time	100 GB	1 TB	2 TB	3 TB	4 TB
1 Node	0:15:14	2:35:53	N/A	N/A	N/A
2 Node	0:07:20	1:15:02	N/A	N/A	N/A
3 Node	0:04:32	0:49:15	1:39:45	N/A	N/A
4 Node	0:03:17	0:36:37	1:13:47	1:53:12	N/A
5 Node	0:02:37	0:30:16	1:00:04	1:31:18	1:59:11

Oracle x2270 – Single Node – 100GB ➔ 1 Hr 40 Min
Yahoo - 3452 nodes – 100TB ➔ 2 Hr 43 Min

***BD tested with 2.6Ghz x5650 CPUs, shipping with 3.46 Ghz x5690 CPUs**

Zettaset FlashScale Terasort

FLASH IMPLEMENTATION



Cloud Computing

Public Cloud

- **Amazon's Elastic Compute Cloud (EC2)**
  - Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. Users can increase or decrease the number of resources at will.
- **Amazon's S3 Storage**
  - It's a storage service that offers client's a highly scalable, reliable, and low latency infrastructure.
  - Store an large amounts of data (1 byte to 5TB per S3 object)
  - Charged by the storage Tier
- **Amazon's EBS Storage**
  - Storage Volumes are similar to raw, unformatted block devices.
  - Create storage volumes from 1GB to 1TB that can be mounted as devices by Amazon EC2 images.

Public Cloud

- **Amazon's Elastic Compute Cloud (EC2)**
  - **How to Setup A Hadoop Ecosystem in the Cloud?**
    - Amazon S3
    - Amazon EBS
  - **IP Addressing for Hadoop Service Failover**
    - Any Service that has an automatic failover would require an Elastic IP Address.
  - **Items to Consider in Deciding to utilize a Cloud Infrastructure**:
    - If you need your cluster on 24x7, Amazon's per CPU costs become very expensive, easily comparable to running your own servers in a datacenter including power / connections.

Public Cloud

- **Items to Consider in Deciding to utilize a Cloud Infrastructure or Not**
  - Hadoop is RAM and storage intensive; you will need a cluster of large or Extra Large High-CPU instances.
  - Amazon' storage is problematic. S3 is slow and has no locality, but is durable.
  - Failover of Primary Components (e.g. Name Node Server) may take several minutes due to the switching of Elastic IPs.
  - EBS is faster, but is not as reliable. In addition, EBS is more expensive if you perform large scale analytics, as it charges for both storage and each I/O operations.
  - Virtualized, shared network cloud environments such as EC2 often experience periods of high latency during peak traffic conditions, even internal communications within the cluster.

Open Problems

- Execution Frameworks that are fault tolerant, scalable, and are amenable to problems that are poorly suited for MapReduce.
- Real Time Analysis Frameworks. Hadoop is entirely batch oriented (except for HBase, which is a key-value store, not an analytics platform). For instance, HBase coprocessors, Twitter Storm, Yahoo! S4. We need to explore tradeoffs between durability, consistency, scalability, etc.
- Hot Failover, sub second for all services.

Any Questions?