# I/Opener to the Big I/O

*Oleg Zhurakousky*

*Principal Architect @ Hortonworks*

*Twitter: z_oleg*

# Agenda

- **Domain Problem**
  - What is Big Data?
  - Structured vs. Unstructured data and what does it mean?
  - How do we measure it?
  - How do we store it?
  - How do we access it?
- **Architecture**
  - Compute Mechanics
  - Why things are slow – understand the problem
  - Mechanics of Data Capture
  - Mechanics of Data Access
  - Mechanical Sympathy
  - optimize Storage, I/O, Network and CPU

# What is Big Data?

- **New Buzzword**
- **Means many things to many different people**

***Data sets that are too large and complex to manipulate or interrogate with standard methods or tools.***

*Wikipedia*

- Is this complete?
- Is something missing?

# What is Big Data?

- **How do we measure it?**
  - Size?
  - If so what Is Big?
  - If not what else?
- **How do we store it?**
  - Dump and go?
- **How do we access it?**
  - Blind Scan relying only on parallelism provided by Hadoop?
  - Bring third-party technology to make sense of it?

*One thing is for sure the data will live somewhere for us to use.*

# Big Data problem

- **Historically, Big Data problem was defined by its size and to speed up its processing Hadoop gave us:**
  - Distributed file system - HDFS
  - Distributed Computation framework – Map Reduce

*Distribute the data and bring your code to data*

*.*

# Big Data compute vs. I/O mismatch?

- **Distributed I/O is coupled with Distributed compute**
  - Blocks to Splits to Input Formats(Readers) to “units of data” to MR Tasks
- **Two orthogonal problems**
  - Speed of serving the “unit of data” from its store to the task
  - Speed of processing the “unit of data” before the next one can be served
- **Mismatch**
  - I/O may be slower then compute or vice versa resulting in imbalanced system.

# Solving mismatch?

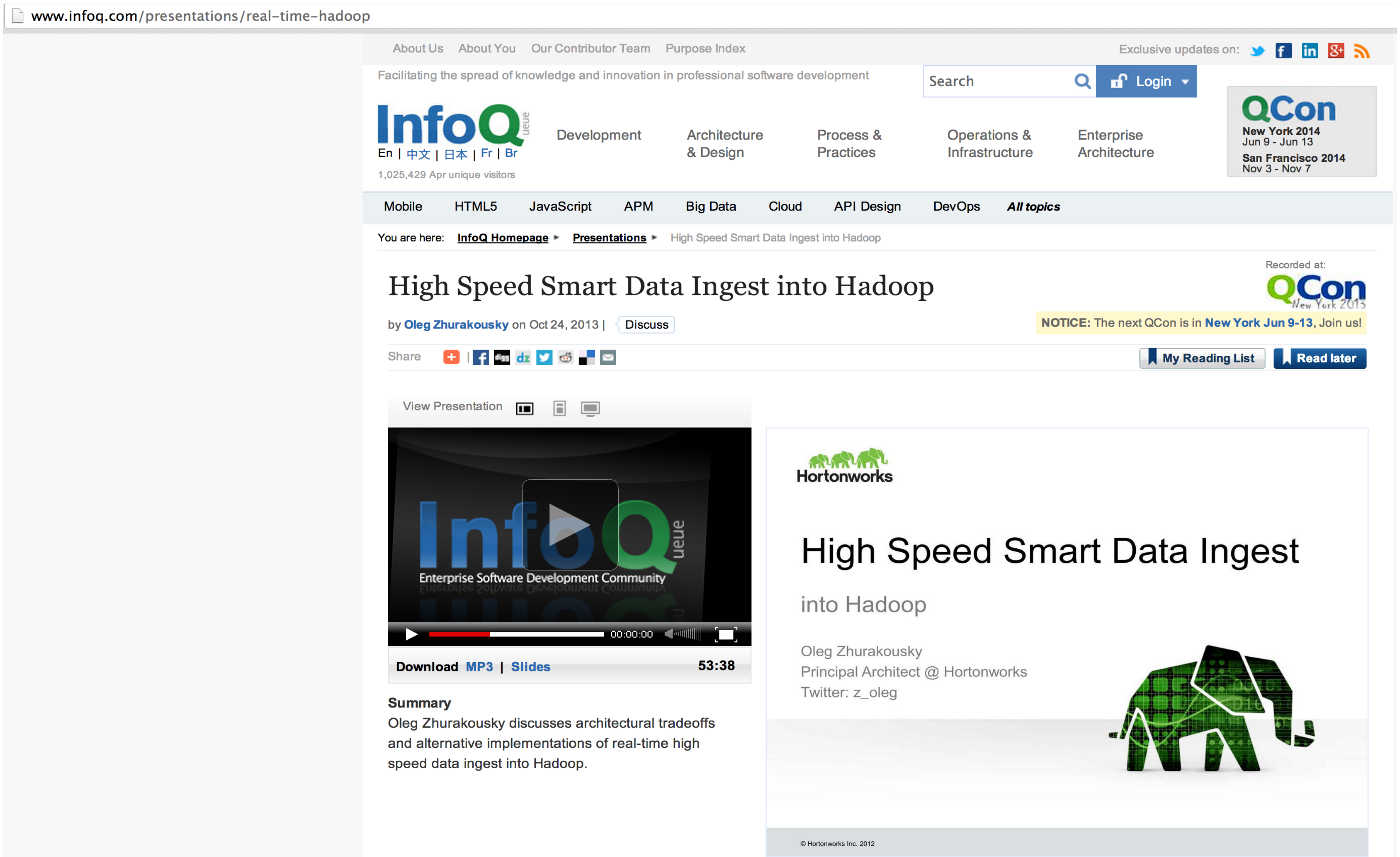
- **Custom Data Buffering**
- **Data Encoding and Compression**
  - Generalization vs. Specialization
- **Data organization**
  - Pages and efficient page creation
  - Meta-information about your data
  - Headers and Footers
  - File naming
  - Directory structure
  - Page caching
  - Data Sampling
- **Efficient Input Formats (data readers)**
  - “Word Count”  - is what NOT to do!

# Strive for Mechanical Sympathy

- **Mechanical Sympathy – *ability to write software with deep understanding of its impact or lack of impact on the hardware its running on***
- **http://mechanical-sympathy.blogspot.com/**
- **Ensure that all available resources (hardware and software) work in *balance* to help achieve the end goal.**

*And it all starts from Data Capture*

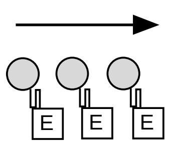
# Smart and efficient Data Capture



# Streaming Sources

- **When dealing with Event data (log messages, call detail records, other event / message type data) there are some important observations:**
  - Source data is streamed (not a static copy), thus always in motion
  - More then one streaming source
  - Streaming (by definition) never stops
  - Events are rather small in size (up to several hundred bytes), but there is a lot of them

  ***Hadoop was not design with that in mind***




Geos

**Telco Use Case**

- Tailing the syslog of a network device
- 300K events per second per streaming source
- Bursts that can double or triple event production

# Mechanics of Data Capture & Why and which things are slow and/or inefficient?

*Big Data begins with its capture. How you do it will have an affect on everything downstream (e.g., access, search, etc.).*

# Mechanics of a compute


# Why things are slow?

- **I/O | Network bound**
- **Inefficient use of storage**
- **CPU is not helping**
- **Memory is not helping**

***Imbalanced resource usage leads to limited growth potentials and misleading assumptions about the capability of your hardware***

# Mechanics of the capture

- **Multiple sources of data (readers)**
- **Write (Capture) >= Read (Source) | Else**
  - Slows everything down
  - Fills accumulation buffer to the capacity
  - Renders parts or whole system unavailable which may brings everything down
  - Etc…

***"Else" is just not good!***

# Strive for mechanical sympathy

- **Demo**

# Data Organization

**<24> 2013-04-16T13:46:36 CREATE: proto 2 (UDP), ge-11/0/0.0:100.120.188.5:32071**
**<24> 2013-04-16T13:46:36 DELETE: proto 3 (UDP), ge-11/0/0.0:13.10.138.5:31989**
**<24> 2013-04-16T13:46:37 APPEND: proto 2 (HTTP), ge-11/0/0.0:98.10.145.5:2030**
**<24> 2013-04-16T13:46:37 CREATE: proto 2 (TCP), ge-11/0/0.0:120.120.88.5:6475**
**. . .**

**Dictionary:**

**[0]<24>, [1]2013-04-16T13:46:36, [2]2013-04-16T13:46:37, [3]CREATE, [4]DELETE, [5]APPEND, [6]proto, [7]2, [8]3, [9]UDP, [10]TCP, [11]HTTP, [12]ge-11/0/0.0, [13]100.120.188.5, [14]13.10.138.5, [15]120.120.88.5, [16]98.10.145.5, [17]32071, [18]31989, [19]6475, [20]2030**

**Index:**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 3 | 6 | 7 | 9 | 12 | 13 | 17 |
| 0 | 1 | 4 | 6 | 8 | 9 | 12 | 14 | 18 |
| 0 | 2 | 5 | 6 | 7 | 11 | 12 | 16 | 20 |
| 0 | 2 | 3 | 6 | 7 | 10 | 12 | 15 | 19 |

**SAME**
**RANGE**
**RANDOM**

# The Suitcase Pattern

- **Storing event data in unorganized form is not feasible**
  - Event Data just keeps on coming, and storage is finite
  - The more history we can keep in Hadoop the better our trend analytics will be
  - Not just a Storage problem, but also Processing
  - I/O is #1 impact on both Ingest & MapReduce performance

- **Suitcase Pattern**
  - Before we travel, we take our clothes off the rack, organize and pack them (easier to store)
  - We may unpack them when we arrive to put them back on the rack (easier to process), but we may choose to use it as is.


  - Consider event data “traveling” over the network to Hadoop
  - we want to organize it before it makes the trip, but in a way that facilitates how we intend to process it once it arrives

# What’s next?

- **Is our data really unstructured?**
- **Have we considered sorting of data elements**
- **How much data is available or could be available to us during the ingest is lost after the ingest?**
- **How valuable is the data available to us during the ingest after the ingest completed?**
- **How expensive would it be to retain (not lose) the data available during the ingest?**

# Data available during the ingest?

- **Record count**
- **Highest/Lowest record length**
- **Average record length**
- **Compression ratio**

***But with a little more work. . .***

- **Column parsing**
  - Unique values
  - Unique values per column
  - Access to values of each column independently from the record
  - Relatively fast column-based searches, without indexing
  - Value encoding
  - Etc…

***Imagine the implications on later data access. . .***

# Anything else?

- **Value packing**
  - Printable ASCII characters can be easily packed in 5-6 bits – 20-45% space increase.
  - Constant values could be represented as fixes-length Integers which depending on the range could be packed in even less bits

# Conclusion - 1

- **Hadoop data partitioning and distributed processing can only take you so far**
  - Mappers are event-driven consumers: they do not have a running thread until a callback thread delivers a message, so think of what's in that message in relation to what needs to be done with it.
- **"Don't wake me up unless it is important"**
  - In Hadoop, generally speaking, several thousand bytes to several hundred thousand bytes is deemed important

  - Buffering records during collection allows us to collect more data before waking up the elephant
  - Buffering records during collection also allows us to organize the whole block of records as a single record to be sent over the network to Hadoop – resulting in lower network and file I/O

# Conclusion - 2

- **Understand your SLAs, but think ahead**
  - While you may have accomplished your SLA for the ingest your overall system may still be in the unbalanced state
  - Don’t settle on “throw more hardware” until you can validate that you have approached the limits of your existing hardware
  - Strive for mechanical sympathy. Profile your software to understand its impact on the hardware its running on.
  - Don’t think about the data capture in isolation. Think how the captured data is going to be used.
  - Analyze, the information that is available to you during the capture and devise a convenient mechanism for storing it. Your data analysts will thank you for that.

# Thank You!

Questions & Answers

Follow: @z_oleg, @hortonworks