
Hadoop at ContextWeb

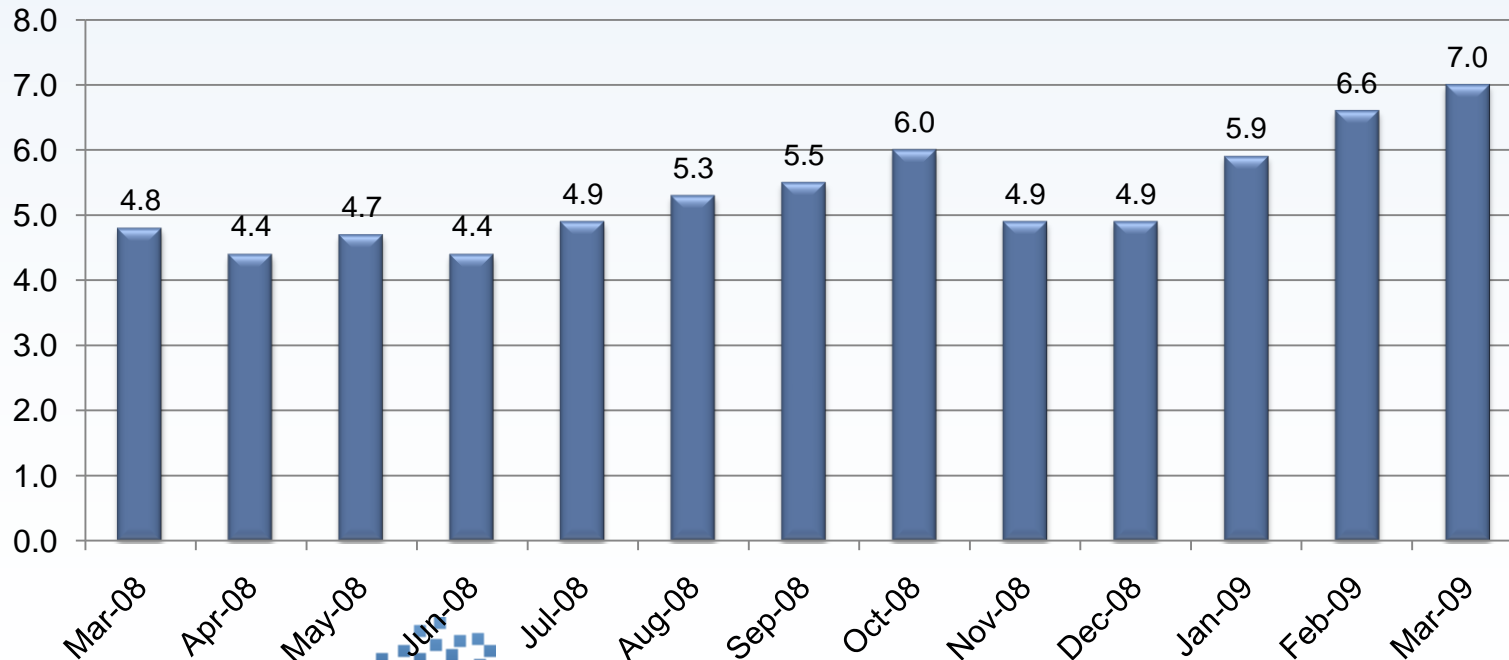
Alex Dorman, VP Engineering
Paul George, Sr. Systems Architect



June 2009

ContextWeb: Traffic

- ▶ ADSDAQ – Advertisement Exchange
- ▶ Traffic – up to 6 thousand Ad requests per second.
- ▶ Comscore Trend Data:
ContextWeb Total Impressions (in billions)



ContextWeb Architecture highlights

▶ Pre – Hadoop aggregation framework

- Logs are generated on each server and aggregated in memory to 15 minute chunks
- Aggregation of logs from different servers into one log
- Load to DB
- Multi-stage aggregation in DB
- About 20 different jobs end-to-end
- Could take 2hr to process through all stages

Hadoop Data Set

- ▶ Up to 100GB of raw log files per day. 40GB compressed
- ▶ 60 different aggregated data sets 15TB total to cover 1 year (compressed)
- ▶ 40 different reports for Business and End Users

Architectural Challenges

- ▶ How to organize data set to keep aggregated data sets fresh.
 - Logs constantly appended to the main Data Set. Reports and aggregated datasets should be refreshed every 15 minutes
- ▶ Mix of .NET and Java applications. (80%+ .Net, 20%- Java)
 - How to make .Net application write logs to Hadoop?
- ▶ Some 3rd party applications to consume results of MapReduce Jobs (e.g. reporting application)
 - How make 3rd party or internal Legacy applications to read data from Hadoop ?

Hadoop Cluster

▶ Today:

- 30 nodes/240 Cores (DELL 2950, 1.8TB per node)
- 50TB total capacity
- NameNode high availability using DRBD Replication.
- Hadoop 0.18.3 + patches
- In-house developed Java framework on top of hadoop.mapred.*
- PIG and Perl Streaming for ad-hoc reports
- ~1,200 MapReduce jobs per day
- OpsWise scheduler
- Exposing data to Windows: WebDav Server with WebDrive clients
- Reporting Application: Qlikview
- Cloudera support for Hadoop
- Archival/Backup: Amazon S3

▶ By end of 2009

- ~50 nodes/400 Cores
- ~85TB total capacity

High Availability for NameNode

- ▶ Two nodes to satisfy availability requirements.
- ▶ High availability for internal components of each node.
 - Disk redundancy
 - Network redundancy
- ▶ Redundant network architecture.
- ▶ Heartbeat mechanism between the two nodes.
- ▶ Replication of namenode metadata.
- ▶ Automatic fail over with no human action required.

NameNode: Internal Components

- Disks
 - 2x 300 GB 15k RPM SAS.
 - Hardware RAID 1 mirroring.
- Network
 - Dual 1Gbps on-board NICs.
 - Linux bonding with LACP.

Redundant Network Architecture

- Linux bonding
 - See bonding.txt from Linux kernel docs.
 - LACP, aka 802.3ad, aka mode=4.
 - (http://en.wikipedia.org/wiki/Link_Aggregation_Control_Protocol)
 - Must be supported by your switches.
 - Throughput advantage
 - Observed at 1.76Gb/s
 - Allows for failure of either NIC instead of a single heartbeat connection via crossover.

Heartbeat Between Nodes and DRBD replications

- Provided by "heartbeat" package.
 - (<http://www.linux-ha.org/>)
- Manage multiple resources:
 - Virtual IP address
 - DRBD Disk
 - Hadoop processes
- 3 second timeout for "deadtime".
- Created LSB compliant hadoop init script.
- DRBD replication of metadata (<http://www.drbd.org/>)
- More details:
<http://files.meetup.com/1228907/Hadoop%20Namenode%20High%20Availability.pptx>

Partitioned Data Set: Date/Time

- ▶ Date/Time as main dimension for Partitioning
- ▶ Segregate results of MapReduce jobs into Monthly, Daily or Hourly Directories
- ▶ Use `MultipleOutputFormat` to segregate output to different files
- ▶ Reprocess only what changed – check `DateTime` in filename to determine what is affected. Data Set is regenerated if input into MR job contains data for this Month/Day/Hour.
- ▶ Use `PathFilter` to specify what files to process

Partitioned Data Set: Revisions

- ▶ Need overlapping jobs:

12:00 -12:10 Job 1.1 A->B

12:10-12:20 Job 1.2 B->C

12:20-12:30 Job 1.3 C->D

12:15-12:25 Job 2.1 A->B !!! Job 1.2 is still reading set B !!!

12:25-12:35 Job 2.2 B->C

12:35-12:45 Job 2.3 C->D

- ▶ Use revisions:

12:00 -12:10 Job 1.1 A.r1->B.r1

12:10-12:20 Job 1.2 B.r1->C.r1

12:20-12:30 Job 1.3 C.r1->D.r1

12:15-12:25 Job 2.1 A.r2->B.r2

12:25-12:35 Job 2.2 B.r2->C.r2

12:35-12:45 Job 2.3 C.r2->D.r2

- ▶ Assign revision (timestamp) when generate output

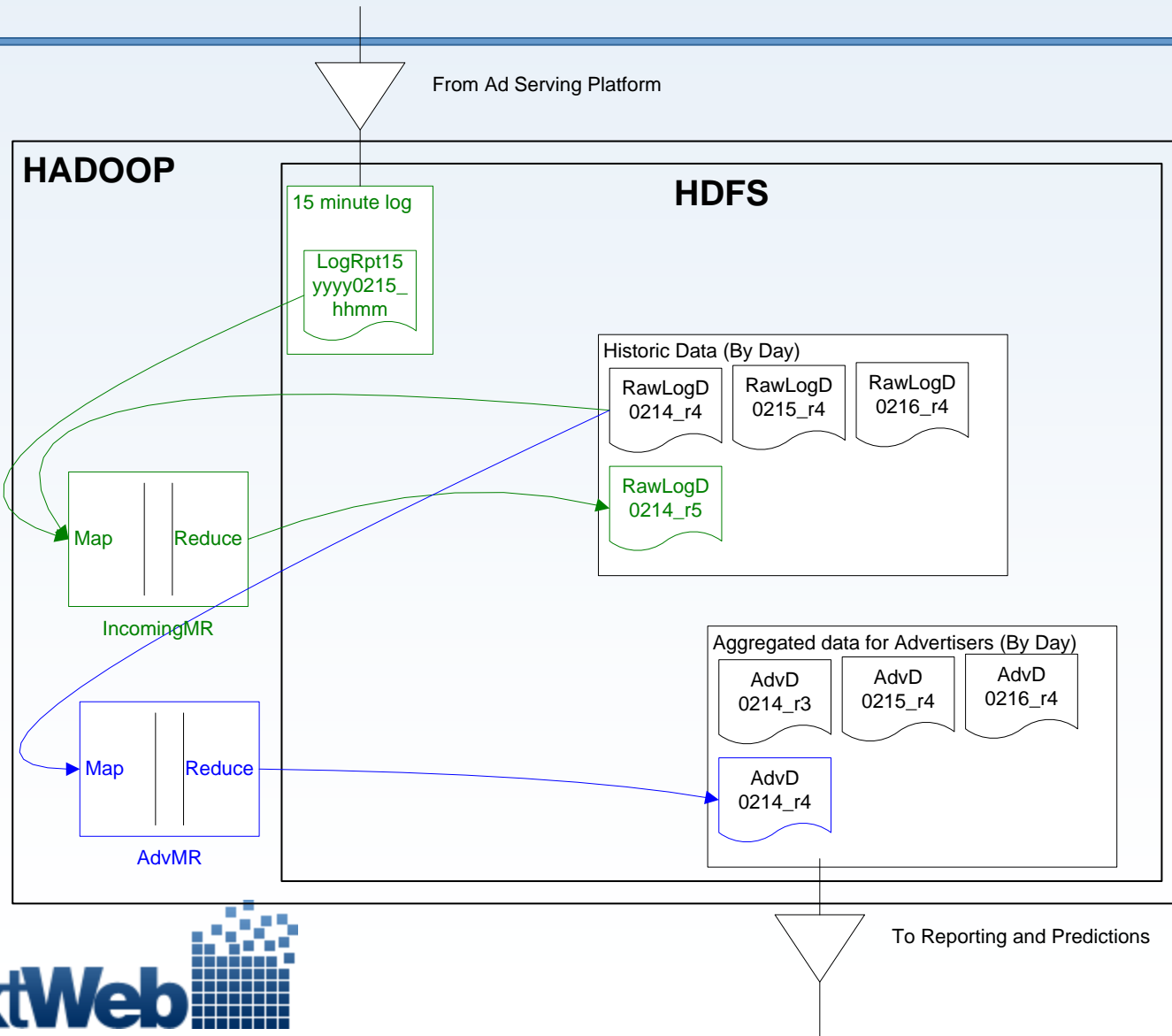
- ▶ Use `MultipleOutputFormat` to segregate output to different files

- ▶ Use highest available revision number when selecting input

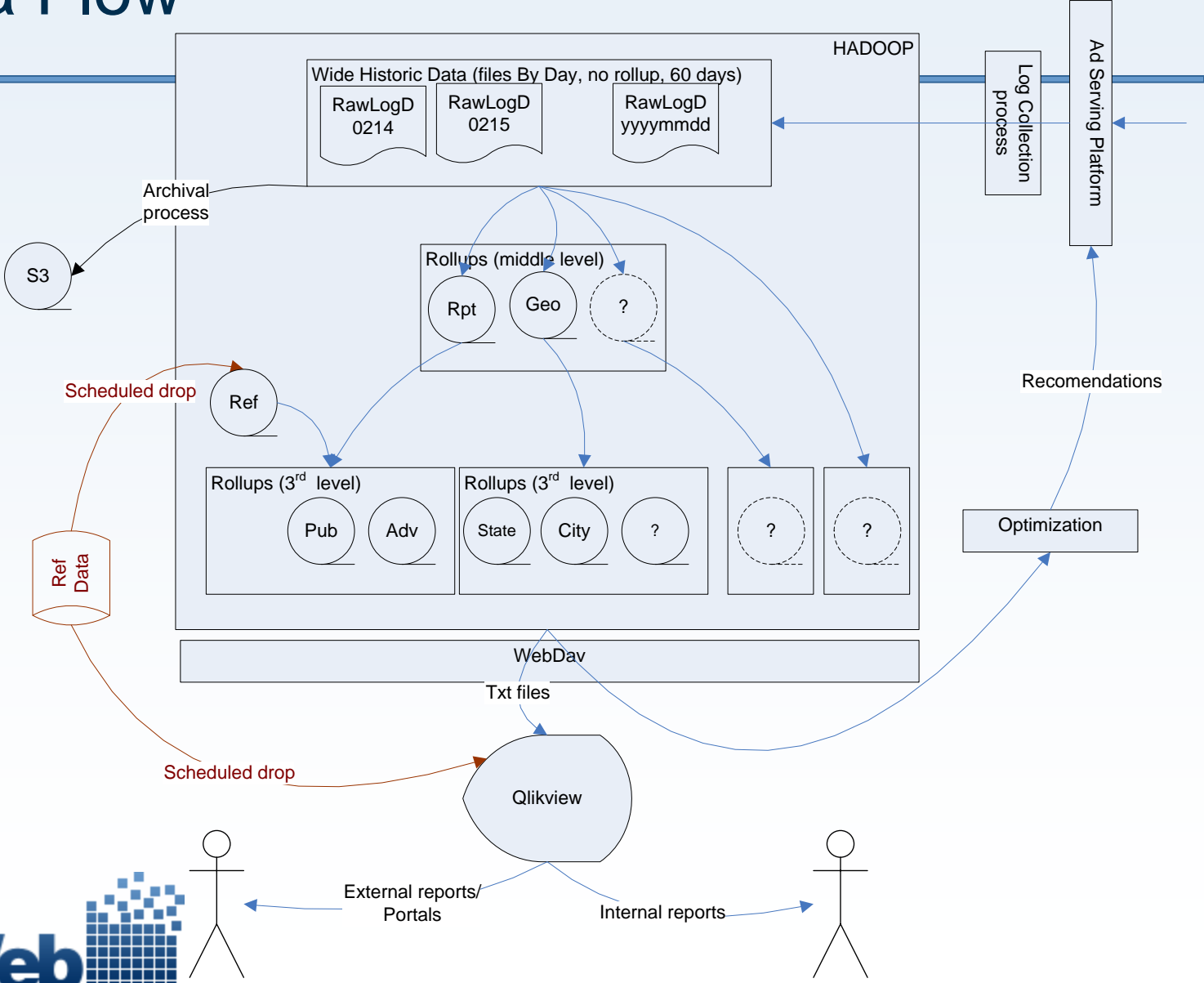
- ▶ Use `PathFilter` to specify revisions to process

- ▶ Clean up “old” revisions after some grace period

Partitioned Data Set: processing flow

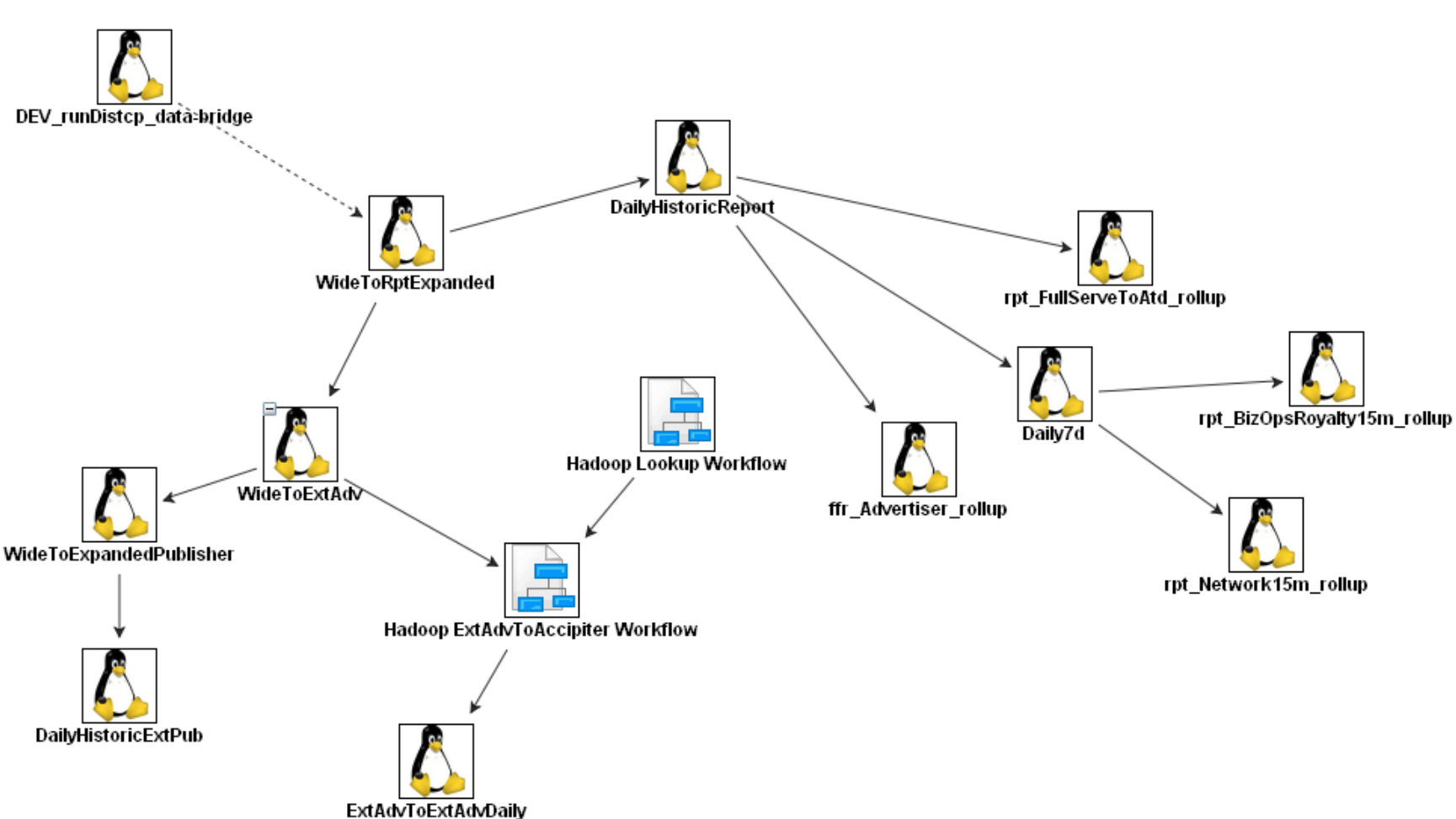


The Data Flow



Workflow

▶ Opswise scheduler



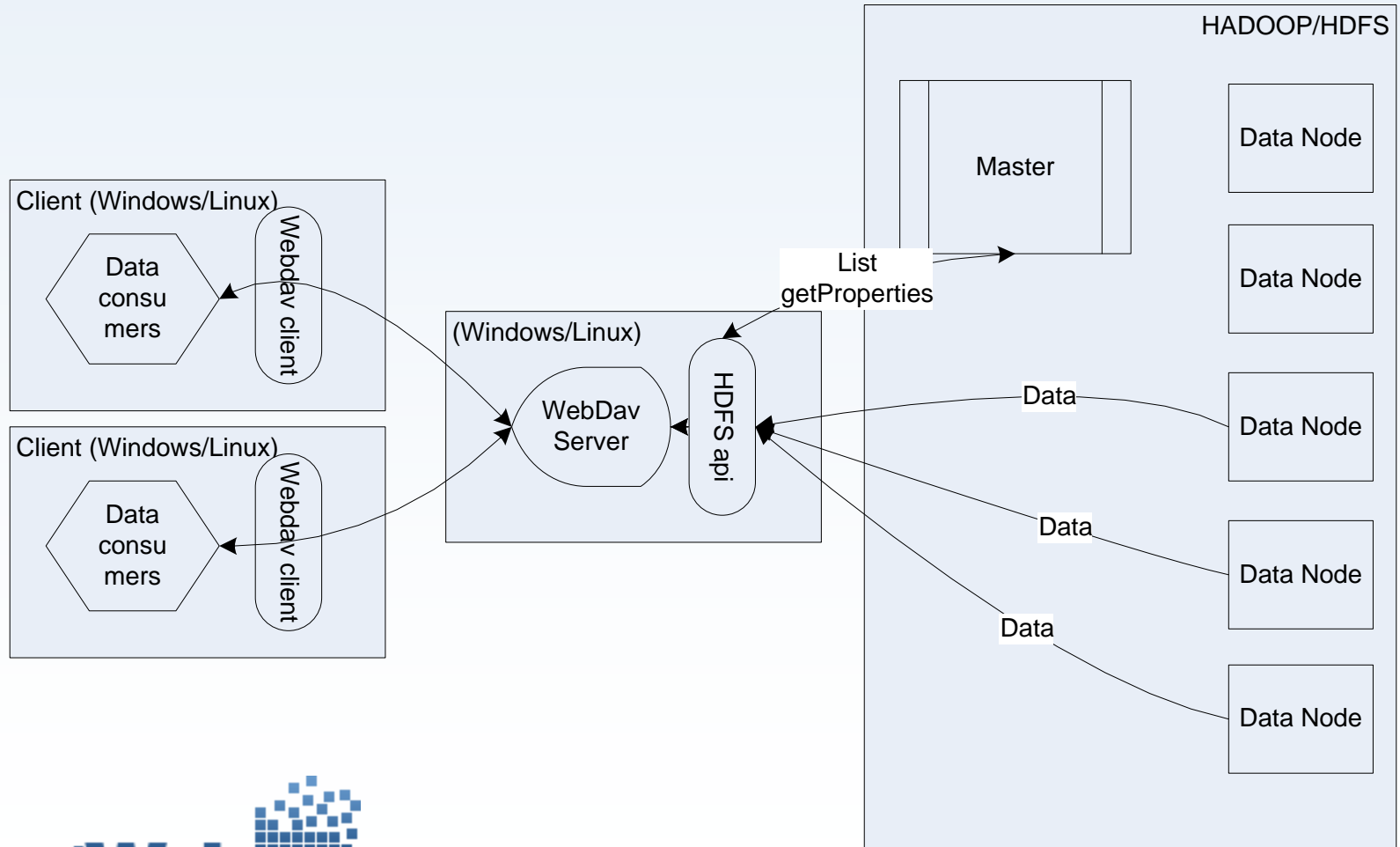
Getting Data in and out

- ▶ Mix of .NET and Java applications. (70%+ .Net, 30%- Java)
 - How to make .Net application write logs to Hadoop?
- ▶ Some 3rd party applications to consume results of MapReduce Jobs (e.g. reporting application)
 - How make 3rd party or internal Legacy applications to read data from Hadoop ?

Getting Data in and out: WebDAV driver

- ▶ WebDAV server is part of Hadoop source code tree
 - Needed some minor clean up. Was co-developed with IponWeb.
Available <http://www.hadoop.iponweb.net/Home/hdfs-over-webdav>
- ▶ There are multiple commercial Windows WebDav clients you can use (we use WebDrive) <http://www.webdrive.com/>
- ▶ Linux
 - ▶ Mount Modules available from <http://dav.sourceforge.net/>

Getting Data in and out: WebDav



WebDAV and compression

- ▶ But your results could be stored on HDFS as compressed files...
- ▶ Options:
 - Decompress files on HDFS – an extra step again
 - Refactor your application to read compressed files...
 - Java – Ok
 - .Net – much more difficult. Cannot decompress SequenceFiles
 - 3rd party- not possible
- ▶ Solution –
 - extend WebDAV to support compressed SequenceFiles

QlikView Reporting Application

- ▶ In-memory DB
- ▶ AJAX support for integration into WEB portals
- ▶ TXT files are supported
- ▶ Understands headers
- ▶ WebDav allows to load data directly from Hadoop
- ▶ Coming soon: generation of Qlikview files as output of Hadoop MR jobs