# Lawrence Spiwak Deep Azure Final Project HD Insight

## Problem Statement

Attempt faster data loads of millions of records into Mongo DB utilizing the Hadoop MapReduce infrastructure built into the HD Insight cluster. The RXBC MRDD and PBI extract at RelayHealth produces daily files ranging from 7 million to 23 million or more data documents per file. A single-threaded load would take too long, but Mongo can accommodate multiple threads connected to it, loading data in parallel to speed up loads by orders of magnitude. Compare this performance to RelayHealth Data Service’s grid-computing environment’s nominal performance.

## Technology Overview

Microsoft Azure HD Insight / Hadoop MapReduce

Cluster Type, HDI Version

Hadoop on Linux (HDI 3.6)

Head Nodes (D12 v2 x2) Worker Nodes (D4 v2 x4) – 40 cores

Cosmos DB / Mongo DB API

prod\_mrdd / MRDD collection @ 1000 RU/second

## Overview of steps

1. Configured environment (installed HD Insight, Cosmos DB)
2. Configured running environment (Java code base, Java libraries, Java compile script, Hadoop run script) by cloning from git repository <https://github.com/lumkichi/DeepAzure_Project>
3. Obtained data from RelayHealth DataServices (available in GitHub)
4. Copy out connection string from Azure Portal and ReCompile code
5. Load the data into HDFS
6. Run the MongoLoader.jar
7. View data load success in RoboMongo 3T

## Data Set

The Data set originates daily and internally at RelayHealth DataServices’ RXBC daily batch process. Known as the “Most Recently Dispensed Drug for the Most Frequently Dispensed Quantity” or “MRDD” for short, it is a collection of some 240 million records of every drug (by therapeutic class) dispensed at every pharmacy that has come through RelayHealth’s switching network. It is a flat-file of 15 fields containing the most recent pricing information for nearly all drugs across all pharmacies

## Software & Libraries

Java JDK 1.8 (installed), Hadoop HDI 3.6 (installed)

MongoDB Java Driver <https://oss.sonatype.org/content/repositories/releases/org/mongodb/mongodb-driver/3.6.1>

SLF4J No Op library <http://central.maven.org/maven2/org/slf4j/slf4j-nop/1.7.25/>

BitVise SSH Client <https://www.bitvise.com/ssh-client-download>

RoboMongo 3T <https://robomongo.org/download>

## Lessons Learned, Pros / Cons

## RelayHealth Data Services currently has a grid infrastructure with 2 control nodes and over 40 remote servers with a total of 1250 cores.

## While a parallel computing infrastructure has been developed in-house and put to good use through, it was well-worth investigating if an off-the-shelf technology such as Hadoop in the Cloud could be leveraged to handle the work we currently do in-house.

## Surprisingly easy (for a seasoned Java developer like me) to utilize.

## However because of the way MapReduce breaks up the file into chunks determined by size rather than row count, the data fed to the MongoMap was sub-optimal:

## Could not take advantage of more parallel threads

## Could not take advantage of bulk insert operations which are much faster than single inserts

## MongoDB was also restricted at 1000 RU/s, which also led to suboptimal operation

## However programs must be written in Java with the Hadoop libraries compiled in, requires dedicated rewrite instead of simply “porting over” Data Services code.

## Demonstrated that we can successfully load Mongo Data in Parallel.

## YouTube URL’s & Git Repository

* Two minute (short): <https://www.youtube.com/watch?v=4XZVUQicuIM>
* 15 minutes (long): <https://www.youtube.com/watch?v=YHRTkNVww3k&t=617s>
* GitHub Repository with all artifacts: <https://github.com/lumkichi/DeepAzure_Project>