Three Generations of Tools/Paradigms Realizing Machine Learning Algorithms

Innovation Architected.

Dr. Vijay Srinivas Agneeswaran

Director, Technology and

Head, Big-Data R&D, I-Labs

We Implement Big Data

Agenda

- Introduction
  - Hadoop Suitability – iterative and real-time applications
- Three generations of realizations – machine learning (ML) algorithms
- Third generation tools
  - Spark
  - comparison of graph processing paradigms
  - ML algorithm in Storm-Kafka – manufacturing use case
- Benchmarking Big-data analytics

Introduction: Hadoop Adoption Status

- Enterprise level – slowly becoming mainstream
  - Experimental – lot of big companies have their own Hadoop clusters including Sears, Walmart, Disney, AT&T etc.
- Business use case
  - Extract, Transform, Load ETL/ELT/data refinement
    - Pentaho, Datameer SMEs in this space.
    - Big-players – Informatica, Splunk (log analytics company) and IBM
- Industry-wise adoption
  - Financial investment/trading – quite high, just as for any new tech.
  - Banking Financial – slower.
  - Telecom, Retail – cautious.

Introduction: Hadoop Adoption Future

- Hindrances/Hurdles towards Hadoop adoption
    - Single cluster – Hadoop YARN is the way forward.
    - Lack of ODBC connectivity
- Hadoop suitability
    - Mainly for embarrassingly parallel problems
    - Not suitable if data splits need to communicate or data splits are inter-related.
    - Map-Reduce for iterative computations
        - Hadoop not currently well suited – no long lived MR job nor in-memory data structures for persisting/caching data across MR iterations.

Suitability of Map-Reduce for Machine Learning

- Origin in functional programming languages (Lisp and ML)
- Built for embarrassingly parallel computations
  - Map: (k1, v1) -> list(k2, v2)
  - Reduce: list(k2, list(v2)) - > list(v2).
- Algorithms which can be expressed in Statistical Query Model in summation form – highly suitable for MR [CC06] – suitable for Hadoop MR.
- Different categorization of ML algorithms
  - Algorithms which need a single execution of an MR model
  - Algorithms which need sequential execution of fixed no. of MR models.
  - Algorithms where 1 iteration is a single execution of MR model.
  - Algorithms where 1 iteration itself needs multiple MR model executions.

[CC06] Chu, C.-T., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G. R., Ng, A. Y., and Olukotun, K. Map-reduce for machine learning on multicore. In *NIPS* (2006), pp. 281--288.

What about Iterative Algorithms?

- What are iterative algorithms?
  - Those that need communication among the computing entities
  - Examples – neural networks, PageRank algorithms, network traffic analysis
- Conjugate gradient descent
    - Commonly used to solve systems of linear equations
    - [CB09] tried implementing CG on dense matrices
    - DAXPY – Multiplies vector x by constant a and adds y.
    - DDOT – Dot product of 2 vectors
    - MatVec – Multiply matrix by vector, produce a vector.
  - 1 MR per primitive – 6 MRs per CG iteration, hundreds of MRs per CG computation, leading to 10 of GBs of communication even for small matrices.
- Other iterative algorithms – fast fourier transform, block tridiagonal

[CB09] C. Bunch, B. Drawert, M. Norman, Mapscale: a cloud environment for scientific computing, Technical Report, University of California, Computer Science Department, 2009.

Further exploration: Iterative Algorithms

- [SN12] explores CG kind of iterative algorithms on MR
- Compare Hadoop MR with Twister MR (http://iterativemapreduce.org)
  - It took 220 seconds on a 16 node cluster to solve system with 24 unknowns, while for 8000 unknowns – took almost 2 hours.
  - MR tasks for each iteration – computation is too little, overhead of setup of MR tasks and communication is too high.
    - Data is reloaded from HDFS for each MR iteration.
    - Surprising that Hadoop does not have support for long running MR tasks
- Other alternative MR frameworks?
  - HaLoop [YB10] – extends MR with loop aware task scheduling and loop invariant caching.
  - Spark [MZ10] – introduces resilient distributed datasets (RDD) – RDD can be cached in memory and reused across iterations.
- Beyond MR – Apache Hama (http://hama.apache.org) – BSP paradigm

[SN12] Satish Narayana Srirama, Pelle Jakovits, and Eero Vainikko. 2012. Adapting scientific computing problems to clouds using MapReduce. *Future Generation Computer Systems* 28, 1 (January 2012), 184-192, Elsevier Publications

[YB10] Yingyi Bu, Bill Howe, Magdalena Balazinska, Michael D. Ernst. HaLoop: Efficient Iterative Data Processing on Large Clusters In*VLDB'10*: The 36th International Conference on Very Large Data Bases, Singapore, 24-30 September, 2010

[MZ10] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing* (HotCloud'10). USENIX Association, Berkeley, CA, USA, 10-10

Data processing: Alternatives to Map-Reduce

- R language
  - Good for statistical algorithms
  - Does not scale well – single threaded, single node execution.
  - Inherently good for iterative computations – shared array architecture.
- Way forward
  - R-Hadoop integration – or R-Hive integration
  - R extensions to support distributed execution.
    - [SV12] is an effort to provide R runtime for scalable execution on cluster.
    - Revolution Analytics is an interesting startup in this area.
- Apache HAMA (http://hama.apache.org) is another alternative
  - Based on Bulk Synchronous Parallel (BSP) model – inherently good for iterative algorithms – can do Conjugate gradient, non-linear SVMs – hard in Hadoop MR.

[SV12] Shivaram Venkataraman, Indrajit Roy, Alvin AuYoung, and Robert S. Schreiber. 2012. Using R for iterative and incremental processing. In *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Computing* (HotCloud'12). USENIX Association, Berkeley, CA, USA, 11-11.

Three Generation of Realization – ML Algos.

- First Generation
  - SAS, SPSS, Informatica etc.
  - Deep analytics – large collection of serial ML algorithms
  - Logistic regression, kernel Support Vector Machines (SVMs), Principal Component Analysis (PCA), Conjugate Gradient (CG) Descent etc.
  - Vertical scaling – increase computation/memory power of node.

Three Generation of Realization – ML Algos.

- Second Generation
  - Mahout, RapidMiner, Pentaho
  - Works over Hadoop Map-Reduce (MR) – can potentially scale to large data sets
  - Shallow analytics may only be possible on Big-data
  - Small number of algorithms only available – including linear regression, linear SVMs, Stochastic gradient descent, collaborative filtering, k-means clustering etc.

Three Generation of Realization – ML Algos.

- Third Generation
  - Spark, HaLoop, Twister MR, Apache Hama, Apache Giraph, GraphLab, Pregel, Piccolo
  - Deep analytics on Big-data?
  - Possible – many more algorithms can be realized in parallel
  - Kernel SVMs, Logistic Regression, CG etc.
  - Motivated by iterative processing + social networks (power law graphs)
  - Realize ML algorithms in real-time – use Kafka-Storm integrated environment.

Paradigms for Processing Large Graphs in Parallel

- Pregel [GM10] – Computation engine from Google for processing graphs
  - Implementation of Bulk Synchronous Parallel (BSP) – paradigm from traditional parallel programming
  - User defined compute() for each vertex at each super-step S.
  - Edges – messages between vertices.
  - Parallelism – Vertex compute functions run in parallel
  - Compute-communicate-barrier – each iteration.
  - Similar open source alternatives – Apache Giraph, Golden orb, Stanford GPS
  - Pregel is good at graph parallel abstraction, ensures deterministic computation, easy to reason with, but
    - user must architect movement of data
    - curse of slow job (barrier synchronization can be slowed by slow jobs – sequential dependencies in the graph).
    - Cannot prioritize/target computation where it is needed most – not adaptive

[GM10] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: A System for Large-scale Graph Processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*(SIGMOD '10). ACM, New York, NY, USA, 135-146.

Piccolo: Another Graph Processing Abstraction

- Piccolo [RP10] – provides asynchronous graph processing abstraction.
  - Application programs comprise
    - control functions – executed on a single machine (master)
      - Create kernels, shared tables, perform global synchronization.
    - Kernel functions – executed on slaves in parallel.
  - Table operations include get, put, update, flush, get_iterator.
  - User defined accumulation functions for concurrent access to table entries.
  - User defined table partition.
- Does not ensure serializable program execution.
  - May be required for some ML algorithms, including dynamic Alternating Least Squares (ALS) and Gibbs sampling.

[RP10] Russell Power and Jinyang Li. 2010. Piccolo: Building Fast, Distributed Programs with Partitioned Tables. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation* (OSDI'10). USENIX Association, Berkeley, CA, USA, 1-14.

GraphLab: Ideal Engine for Processing Natural Graphs [YL12]

- Goals – targeted at machine learning.
  - Model graph dependencies, be asynchronous, iterative, dynamic.
- Data associated with edges (weights, for instance) and vertices (user profile data, current interests etc.).
- Update functions – lives on each vertex
  - Transforms data in scope of vertex.
  - Can choose to trigger neighbours (for example only if Rank changes drastically)
  - Run asynchronously till convergence – no global barrier.
- Consistency is important in ML algorithms (some do not even converge when there are inconsistent updates – collaborative filtering).
  - GraphLab – provides varying level of consistency. Parallelism VS consistency.
  - Implemented several algorithms, including ALS, K-means, SVM, Belief propagation, matrix factorization, Gibbs sampling, SVD, CoEM etc.
  - Co-EM (Expectation Maximization) algorithm 15x faster than Hadoop MR – on distributed GraphLab, only 0.3% of Hadoop execution time.

[YL12] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M. Hellerstein. 2012. Distributed GraphLab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment* 5, 8 (April 2012), 716-727.

GraphLab 2: PowerGraph – Modeling Natural Graphs [1]

- GraphLab could not scale to Altavista web graph 2002, 1.4B vertices, 6.7B edges.
  - Most graph parallel abstractions assume small neighbourhoods – low degree vertices
  - But natural graphs (LinkedIn, Facebook, Twitter) – power law graphs.
  - Hard to partition power law graphs, high degree vertices limit parallelism.
- GraphLab provides new way of partitioning power law graphs
  - Edges are tied to machines, vertices (esp. high degree ones) span machines
  - Execution split into 3 phases:
    - Gather, apply and scatter.
  - Triangle counting on Twitter graph
    - Hadoop MR took 423 minutes on 1536 machines
    - GraphLab 2 took 1.5 minutes on 1024 cores (64 machines)

[1] Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin (2012). "**PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs.**" *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI '12).*

Graph Paradigm	Computation (Synchronous or Asynchronous)	Determinism and Effects	Efficiency VS Asynchrony	Efficiency of Processing Power Law Graphs.
GraphLab1	Asynchronous	Deterministic – serializable computation.	Uses inefficient locking protocols	Inefficient – Locking protocol is unfair to high degree vertices
Pregel	Synchronous – BSP based	Deterministic – serial computation.	NA	Inefficient – curse of slow jobs.
Piccolo	Asynchronous	Non-deterministic – non-serializable computation.	Efficient, but may lead to incorrectness.	May be efficient.
GraphLab2 (PowerGraph)	Asynchronous	Deterministic – serializable computation.	Uses parallel locking for efficient, serializable and asynchronous computations.	Efficient – parallel locking and other optimizations for processing natural graphs.

Comparison of Graph Processing Paradigms

Spark: Third Generation ML Tool

- Two parallel programming abstractions [MZ10]
  - Resilient distributed data sets (RDDs)
    - Read-only collection of objects partitioned across a cluster
    - Can be rebuilt if partition is lost.
  - Parallel operation on RDDs
    - User can pass a function – first class entities in Scala.
    - Foreach, reduce, collect
  - Programmer can build RDDs from
    1. a file in HDFS
    2. Parallelizing Scala collection - divide into slices.
    3. Transform existing RDD - Specify flatmap operations such as Map, Filter
    4. Change persistence of RDD Cache or a save action – saves to HDFS.
  - Shared variables
    - Broadcast variables, accumulators

[MZ10] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing* (HotCloud'10). USENIX Association, Berkeley, CA, USA, 10-10

Some Spark(ling) examples

Scala code (serial)

```
var count = 0
for (i <- 1 to 100000)
{ val x = Math.random * 2 - 1
val y = Math.random * 2 - 1
if (x*x + y*y < 1) count += 1 }
println("Pi is roughly " + 4 * count / 100000.0)
```

Sample random point on unit circle – count how many are inside them (roughly about PI/4). Hence, u get approximate value for PI.

Based on the PS/PC = AS/AC=4/PI, so PI = 4 * (PC/PS).

Some Spark(ling) examples

Spark code (parallel)

```
val spark = new SparkContext(<Mesos master>)
var count = spark.accumulator(0)
for (i <- spark.parallelize(1 to 100000, 12))
 { val x = Math.random * 2 - 1 val
y = Math.random * 2 - 1
if (x*x + y*y < 1) count += 1 }
println("Pi is roughly " + 4 * count / 100000.0)
```

Notable points:

1. Spark context created – talks to Mesos[1] master.
2. Count becomes shared variable – accumulator.
3. For loop is an RDD – breaks scala range object (1 to 100000) into 12 slices.
4. Parallelize method invokes foreach method of RDD.

[1] Mesos is an Apache incubated clustering system – http://mesosproject.org

Logistic Regression in Spark: Serial Code

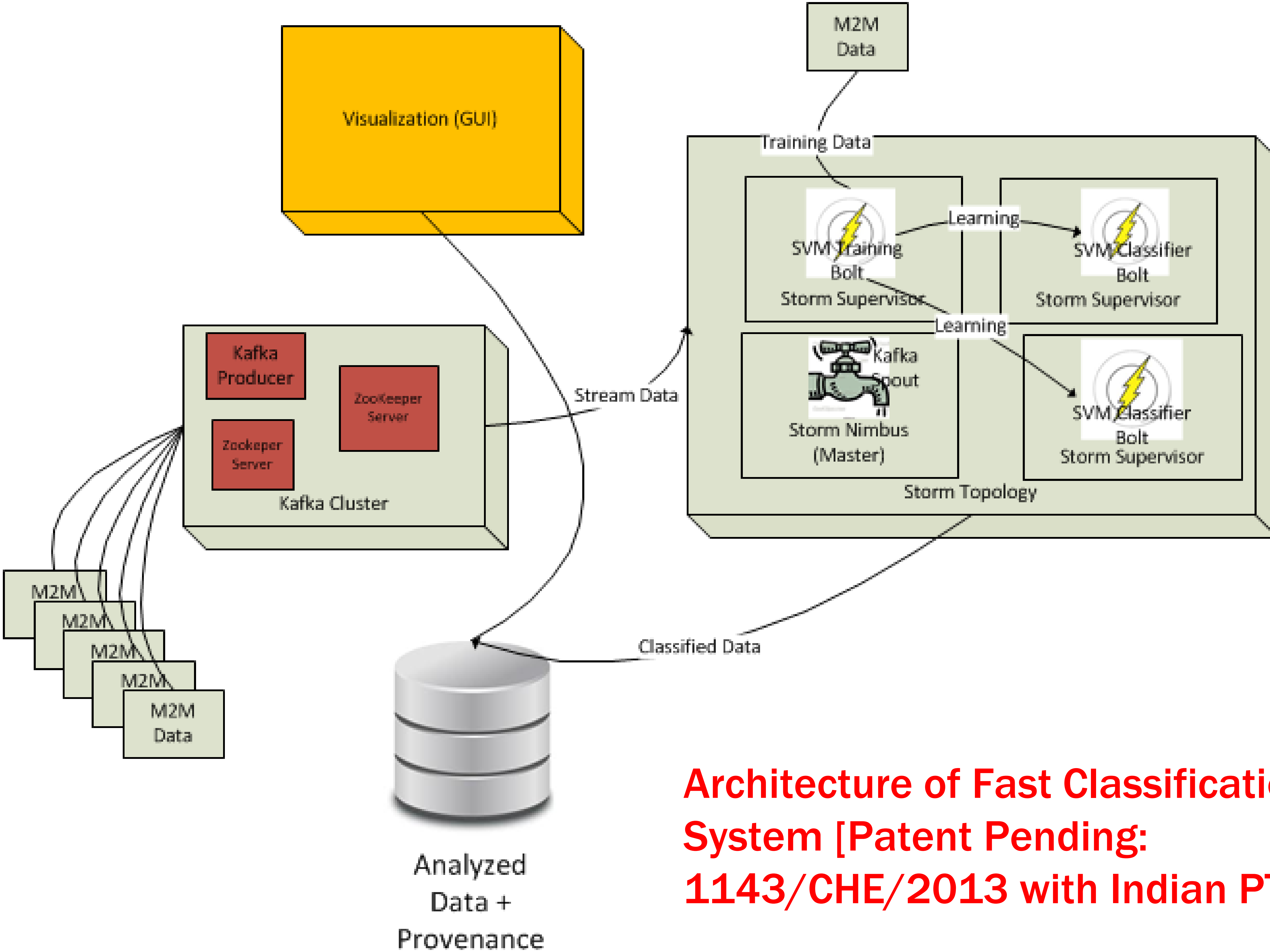
```
// Read data file and convert it into Point objects
val lines = scala.io.Source.fromFile("data.txt").getLines()
val points = lines.map(x => parsePoint(x))

// Run logistic regression
var w = Vector.random(D)
for (i <- 1 to ITERATIONS) {
  val gradient = Vector.zeros(D)
  for (p <- points) {
    val scale = (1/(1+Math.exp(-p.y*(w dot p.x)))-1)*p.y
    gradient += scale * p.x
  }
  w -= gradient
}
println("Result: " + w)
```

Logistic Regression in Spark

```
// Read data file and transform it into Point objects
val spark = new SparkContext(<Mesos master>)
val lines = spark.hdfsTextFile("hdfs://.../data.txt")
val points = lines.map(x => parsePoint(x)).cache()

// Run logistic regression
var w = Vector.random(D)
for (i <- 1 to ITERATIONS) {
  val gradient = spark.accumulator(Vector.zeros(D))
  for (p <- points) {
    val scale = (1/(1+Math.exp(-p.y*(w dot p.x)))-1)*p.y
    gradient += scale * p.x
  }
  w -= gradient.value
}
println("Result: " + w)
```



Architecture of Fast Classification System [Patent Pending: 1143/CHE/2013 with Indian PTO]

Benchmarking Big-data

- Efforts made to benchmark NoSQL databases
  - Yahoo Cloud Serving Benchmark
    - Workloads – insert, read, update, scan
  - BigBench
    - Good big-data querying systems/NoSQLs.
  - Data analytics pipeline
    - End-to-end data consumption from ingestion to analysis
    - Metrics for scoring

Benchmarking Big-data Analytics

- Batch analytics or real-time analytics
  - Performance/scalability VS accuracy
    - Increasing data volume – time taken
  - Throughput VS accuracy
  - Increasing velocity – time taken
- Data munging time
- Machine learning algorithm time
  - Training/Testing

Thank You!

- Mail

  vijay.sa@impetus.co.in

- LinkedIn

http://in.linkedin.com/in/vijaysrinivasagneeswaran

- Blogs

  blogs.impetus.com

- Twitter

  @a_vijaysrinivas.

IMPETUS
Innovation Architected.

Backup slides

Real-time Analytics for Big-Data

- *Interesting technologies in this space.*
  - Google Dremel – incremental processing
    - Open source version led by MapR – Apache Drill
  - Real time analytics Database from Metamarkets – Druid.
  - Apache S4 from Yahoo – distributed stream computing platform.
  - Storm + Kafka + Trident – can be used for highly scalable stream processing + simple aggregation/summarization.

Interesting Startups in this space.

- *Hstreaming, Truviso (acquired by Cisco), Mixpanel (mobile analytics)*
- *Space Time Insight – $14M funding for geospatial and visual analytics software in real-time Big-data* space.

Visualization + analytics at speed of thought

- *Self-service data science – no need of data scientist*
- *Integration of visualization + big-data + Artificial intelligence + social + analytics*
- *Interesting startups in this space – Tableau, Cliktech, Edgespring.*

Video Analytics

- Retail – product pilferage. Nearly 30% loss and 50% of pilferage by employees themselves.
  - Need to analyze few hundred hours of surveillance videos
  - Useful in a no. of security applications
- Approach.
  - Video meta-data extraction, storing in NoSQL DB.
  - Video object identification
    - Parallelized image comparison algorithm
  - All sequences/frames identifying occurrences of a given object in video files.
    - Parallelized algorithm over Hadoop MR.

Video Analytics: State of Art

- Video Analytics – focus mainly on
  - Object identification
  - Indexing/Annotating – creating meta-data on video.
- Tools available
  - OpenTLD a.k.a Predator (https://github.com/zk00006/OpenTLD)
    - Object identification/detection via custom made algorithms
    - Uses Matlab – can work with Octave.
  - OpenCV (Computer Vision project from Intel – http://opencv.org)
    - Open source image processing – segmentation, object identification, motion tracking etc.
    - Uses Machine Learning algorithms including decision trees, random forests, expectation maximization, SVMs etc.
    - Can be re-written to work over Hadoop – works on CUDA as of now.
  - EMC – presented Hadoop MR based algorithms to speed up video analytics.
  - H-Streaming – start-up claims to have MR based video analytics.

Spanner: State of the Art Distributed Database

- Spanner from Google [CJC12] – focus on maintaining cross data centre replicated data.
  - 2 research contributions.
    - Externally consistent reads & writes (Linearizable)
      - Transaction T_1's timestamp < T_2's if T_1 commits earlier than T_2
    - Globally consistent reads across the database at any timestamp
    - Key idea is the TrueTime API – exposes clock uncertainty
      - Guarantees on Spanner's timestamp depends on bounds on uncertainty provided by the implementation.
      - Implementation – uses GPS and atomic clocks based elaborate clock synchronization protocols to minimize uncertainty.
      - Uses Paxos algorithm [LL98] within each Data centre at Tablet level.
      - Directory/bucket – set of contiguous keys

[CJC12] Corbett, James C; Dean, Jeffrey; Epstein, Michael; Fikes, Andrew; Frost, Christopher; Furman, JJ; Ghemawat, Sanjay; Gubarev, Andrey et al., "Spanner: Google's Globally-Distributed Database", *Proceedings of Usenix Conference on Operating System Design and Implementation (OSDI) 2012* (Google).

[LL98] Leslie Lamport. 1998. The part-time parliament. *ACM Transactions on Computer System,* 16, 2 (May 1998), 133-169.