Big-Data: Theory, Analytics and Engineering Perspectives

Dr. Vijay Srinivas Agneeswaran

Director, Technology

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

IMPETUS

services for the Fortune 500®

Big-data: Ming Boggling Numbers

- Digital universe – 1.8 Zettabytes (1 billion terabytes or 10^{12} GB) of data in 2011 (EMC Report)
  - expected to be 2.7 ZB in 2012 and 8 ZB in 2015.
  - 10^{15} files
  - 75% of information generated by individual users.
- 5 billion mobile phones in 2011, 30 billion content pieces on Facebook every month (Mckinsey report).
- US Library of Congress has collected 235 TB of data (Infographic)
  - Data per company in 15/17 sectors in US is > than 235 TB.
- Important areas (Mckinsey report)
  - Healthcare – personalized medicine, clinical trial design, fraud detection etc.
  - Governments (Aadhar project) – increased tax collection, transparency.
  - Retail – consumer behaviour prediction, sentiment analysis, merchandizing
  - Manufacturing – digital factory, R&D design, supply chain management etc.
  - Telecom – Personal location data (GPS and other technologies) – smart routing (navigation), automotive telematics, mobile Location Based Services (LBS).

Top Big-data analyzers/processors

- LinkedIn – petabytes of social data represented as graphs
  - People You May Know feature
- Facebook – analyses petabytes of user generated data
- NY Times – processed 4 TB of raw images in less than a day.
- Amazon – retailer
  - Recommendation system – consumer behaviour analysis
    - *30% of books/products sold*
- Akamai – analyzes 75 million events per day
  - Targeted advertising
- Twitter – 340 million tweets per day or about 4000 tweets per second on average.
  - Peak 15000 tweets/second for Spain’s fourth goal in Euro 2012.
- Google – processes around 20000 terabytes (20 petabytes) per day.
- Flickr – 6 billion images (Flickr blog)

Brewer's CAP Conjecture

Liveness

- If something good happens, system is live.
- Availability – safety property.
- Each request eventually receives a response

Safety

- Informally, nothing bad happens = Safety
- Consistency – safety property
- All client requests are given correct responses based on service specification
    - Strict consistency – all readers read correct value – write order is the same across replicas and is propogated to readers synchronously

CAP

CAP Conjecture: trade-off between consistency and
availability in case of failures – network partitions [SG12].

General conjecture: In an unreliable distributed system, both safety and liveness cannot be guaranteed [TDC96]

[TDC96] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. Journal of the ACM, 43(2):225–267, 1996.
[SG12] Seth Gilbert and Nancy A. Lynch. Perspectives on the CAP Theorem. *Computer*, 45(2):30-35, 2012. IEEE.

Consensus

- Termination
- Eventually, every process must output a value

Liveness

Safety

- Agreement
  - Every process must output same value
- Validity
  - The value output must have been input of at least one process

Fault Tolerant Agreement is impossible in a completely asynchronous distributed system [MJF85]

Conjecture

Synchronous distributed system

1. All nodes have synchronized clocks.
2. Message delivery is bounded
3. Every process takes steps at fixed and known rate

[MJF85] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. Journal of the ACM, 32(2):374–382, 1985.

Consensus in Distributed Systems

Consensus

- Initially
  - *processes begin in **undecided** state*
  - *propose an initial value from a set D*
- Then
  - *processes communicate, exchanging values*
  - *attempt to decide*
- cannot change the decision value in **decided** state
- The difficulty
  - *must reach decision even if crash has occurred*
  - *or arbitrary failure!*

Consensus....

Consensus: correctness

- Consistency
  - *All agents agree on same value and decisions are final*
- Validity
  - *The agreed value must have been some agents input*
- Termination
  - *Eventually agent reaches its decision within a finite number of steps*

Consensus....

Processors

- Synchronous/Asynchronous

Message delivery

- Ordered/unordered
- Bounded/unbounded

Communication

- Broadcast/point-to-point

Failures

- Fail-stop/Byzantine

Consensus [1]

Table 1. Conditions under which consensus is possible.

Processors	Message Order: Unordered	Message Order: Unordered	Message Order: Ordered	Message Order: Ordered	Communication
	Transmission: Point-to-point	Transmission: Broadcast	Transmission: Broadcast	Transmission: Point-to-point	
Asynchronous	No	No	Yes	No	Unbounded
	No	No	Yes	No	Bounded
Synchronous	Yes	Yes	Yes	Yes	
	No	No	Yes	Yes	Unbounded

[1] John Turek and Dennis Shasha. 1992. The Many Faces of Consensus in Distributed Systems. *Computer* 25, 6 (June 1992), 8-17.

Consensus in Distributed Shared Memory Systems [1]

Intuitively easier to achieve consensus

- Actually, normal distributed shared memory gives only equivalent of reads and writes.
  - Fault-tolerant consensus is impossible without ordered broadcast in shared memory systems

```
fetch&add(m, v)
  begin /*Atomic action*/
    oldm ← m
    m ← m + v;
    return(oldm);
  end; /*Atomic action*/
```

Figure 2. Fetch&add (consensus number = 2).

```
compare&swap(m, new, old)
  begin /*Atomic action*/
    if (m = old) then
    begin
      m ← new;
      return (true)
    end
    else return (false);
  end; /*Atomic action*/
```

Figure 3. Compare&swap (consensus number = n).

Byzantine Failures: Consensus

- One or more nodes is malicious and prevents others from reaching consensus.

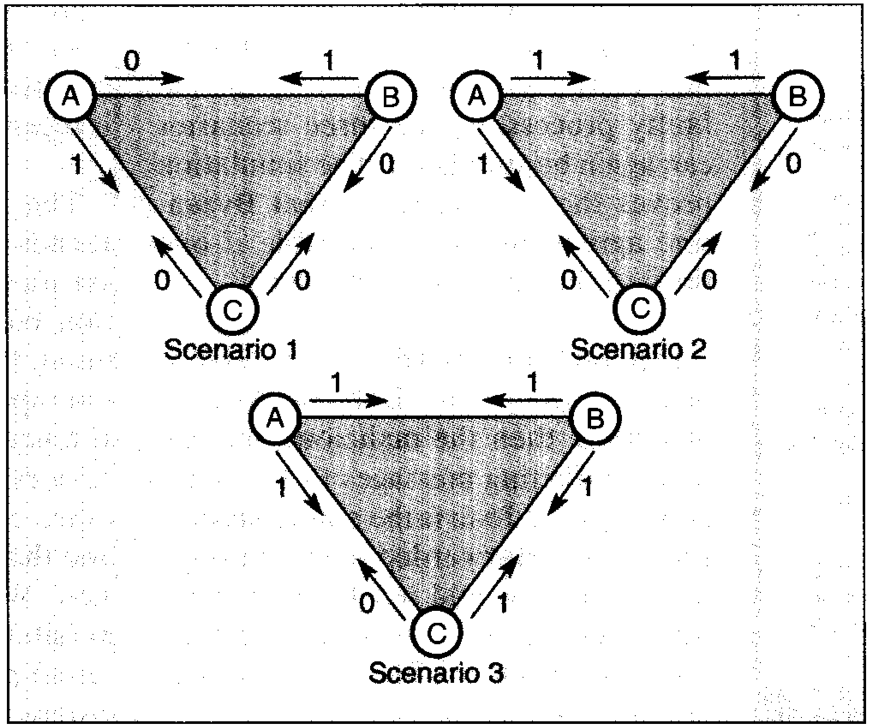


Figure 10. Scenarios leading to failure of Byzantine agreement.

[1] John Turek and Dennis Shasha. 1992. The Many Faces of Consensus in Distributed Systems. *Computer* 25, 6 (June 1992), 8-17.

Brewer's CAP Conjecture

Liveness

- If something good happens, system is live.
- Availability – safety property.
- Each request eventually receives a response

Safety

- Informally, nothing bad happens = Safety
- Consistency – safety property
- All client requests are given correct responses based on service specification
  - Strict consistency – all readers read correct value – write order is the same across replicas and is propogated to readers synchronously

CAP

CAP Conjecture: trade-off between consistency and availability in case of failures – network partitions [SG12]

General conjecture: In an unreliable distributed system, both safety and liveness cannot be guaranteed [TDC96]

[TDC96] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. Journal of the ACM, 43(2):225–267, 1996.

[2] Seth Gilbert and Nancy A. Lynch. Perspectives on the CAP Theorem. *Computer*, 45(2):30-35, 2012. IEEE.

Overcoming limitations of CAP Conjecture

Weaken availability or weaken consistency in the presence of partitions

Best effort availability

- Presence of partitions – make best effort at availability
- Have strong consistency
- Example – Chubby lock service from Google [MB06]
- Paxos [TDC07] or replicated state machine protocol to achieve consistency – assumes presence of primary/master.

Best effort consistency

- Presence of partitions – make best effort at consistency
- Have higher availability - meaning weak reads are allowed.
- Example – Amazon Dynamo – eventual consistency [GD07]
- Updates applied to local copy and then propagated to other replicas – no guarantees about ordering across replicas
- No consistency guarantees in the presence of partitions

[MB06] Michael Burrows. The chubby lock service for loosely-coupled distributed systems. In The Proceedings of the Symposium on Operating System Design and Implementation (OSDI), pages 335–350, 2006.
[TDC07] Tushar D. Chandra, Robert Griesemer, and Joshua Redstone. Paxos made live: an engineering perspective. In The Proceedings of the International Symposium on Principles of Distributed Computing (PODC), pages 398–407, New York, NY, USA, 2007.
[GD07] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon’s highly available key-value store. In ACM Symposium on Oper-ating Systems Principles, 2007.

PACELC Formulation [DJA12]

Consistency-Latency Trade-offs

- Normal operation of a distributed system
  - CAP theorem does not apply
- Network partition is rare occurrence compared to other kinds of failures [MS10]
- Strong consistency can only be achieved under high latency
- Online/cloud data serving systems
  - Amazon Dynamo – created to ensure data is served to core services for e-commerce platforms.
  - PNUTS system from Yahoo – created to serve data to more than 100 Yahoo applications from Weather to Mail to Answers
  - Voldemart from LinkedIn – online updates from write intensive features of social platform
  - Cassandra – Inbox search of Facebook.

[DJA12] Daniel J. Abadi, "Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story," Computer, vol. 45, no. 2, pp. 37-42, Feb. 2012.

[MS10] Michael Stonebraker “Errors in Database Systems, Eventual Consistency and the CAP Theorem”, CACM 2010, available from: http://m.cacm.acm.org/blogs/blog-cacm/83396-errors-in-database-systems-eventual-consistency-and-the-cap-theorem/comments

Why consistency-latency trade-off

Consistency-Latency-Availability Trade-offs [DJA12]

- Availability – can be viewed as a latency issue only
  - data item is unavailable – implies unacceptable latency
  - Latency is below a threshold implies availability.
- Why the trade-off between consistency and latency?
  - Latency forces programmers to prefer local copies even in absence of partitions [RR12]
- Replication is essential to achieve availability – only 3 choices
  1. Data updates sent to all replicas at the same time
     - Replica divergence (order of updates different) – if there is no agreement protocol or a centralized node – preprocessing node.
  2. Data updates sent to a data-item specific node – master for this data-item.
     A. Synchronous – involves latency
     B. Asynchronous – could be inconsistent if reads are from all nodes & consistent if reads are only from master.
     C. Quorum protocols – updates sent to W nodes, reads from any of R nodes, R+W>N for N nodes in the system for consistency reads.
  3. Data updates sent to arbitrary location first – master not always the same node.

•Cassandra, Riak and Dynamo – combination of 2.C and 3 above – quorum protocols, but with different nodes acting as masters.
•PNUTS – chooses 2B – inconsistent reads for reduced latency. In case of partitions, disable data item updates – availability is compromised under CAP (this is to avoid conflicting updates from different partitions).

[DJA12] Daniel J. Abadi, "Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story," Computer, vol. 45, no. 2, pp. 37-42, Feb. 2012.
[RR12] Ramakrishnan, R.; , "CAP and Cloud Data Management," *Computer* , vol.45, no.2, pp.43-49, Feb. 2012
doi: 10.1109/MC.2011.388,.

Yahoo PNUTS Data Serving Platform [BFC08]

Data/Query Model

- Data model
  - Simplified relational model – flexible schemas, blob data types
- Queries
  - Selection, projection over single table
  - Scan (range queries)
- No integrity constraints or complex queries.

Consistency Model

- Record level timeline consistency
  - Stricter than eventual consistency.
  - Master – to order updates
  - Replicas move forward in timeline, never backward.
  - Varying consistency guarantees
    - Read-any
    - Read-critical or read-your-writes
    - Read-latest – synchronous

Others

- Notification model
  - Table level publish-subscribe
    - Multiple topics per table
  - Cache invalidation.
  - Slow clients
    messages above threshold are discarded

[BFC08] Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, and Ramana Yerneni. 2008. PNUTS: Yahoo!'s hosted data serving platform. *Proceedings of the VLDB Endowment* 1, 2 (August 2008), 1277-1288.

NLC – PAC Variation

[AF97] Armando Fox, Steven D. Gribble, Yatin Chawathe, Eric A. Brewer, and Paul Gauthier. 1997. Cluster-based scalable network services. In *Proceedings of the sixteenth ACM symposium on Operating systems principles* (SOSP '97), William M. Waite (Ed.). ACM, New York, NY, USA, 78-91.

NL-PA Systems

- Relaxed consistency systems (Basically Available Soft State Eventually Consistent or BASE [AF97])
  - Amazon Dynamo, Cassandra and Riak
  - Quorum protocols to implement consistency – R readers, W writers with R+W<N for N nodes.
  - In case of partitions, allow weak reads – quorum reads not possible.

NC-PC Systems

- Fully Atomicity Consistency Isolation Durability (ACID) systems
- VoltDB, Megastore and BigTable
  - Pay availability or latency price to achieve consistency.

Others

- NL-PC systems
  - PNUTS from Yahoo.
    - Normal operation – weak reads
    - Under partitions – master unavailable for updates
- NC-PA systems
  - MongoDB
  - Strict consistency under normal operations
  - Under partitions sacrifice consistency.

NoSQL Databases: Another Perspective

Document Stores

- CouchDB, MongoDB, Terrastore
- Document-oriented databases
  - JSON objects or BSON
  - Documents can be organized into collections
  - No transactions

Key value Stores

- Amazon Dynamo, Voldemart [RS12], Riak
- Distributed Hash Tables
  - Some implement consistent hashing

Column Stores

- Cassandra, HBase, BigTable
- Optimized to retrieve mulitple rows within a single column

Graph Databases

- Neo4j, VertexDB, Allegro (Resource Description Framework (RDF from W3C) Store)
- Store vertices, edges and relations.

[RS12] Roshan Sumbaly, Jay Kreps, Lei Gao, Alex Feinberg, Chinmay Soman, and Sam Shah, “Serving Large-scale Batch Computed Data with Project Voldemort” to appear in USENIX Conference on File and Storage Technologies (FAST) 2012.

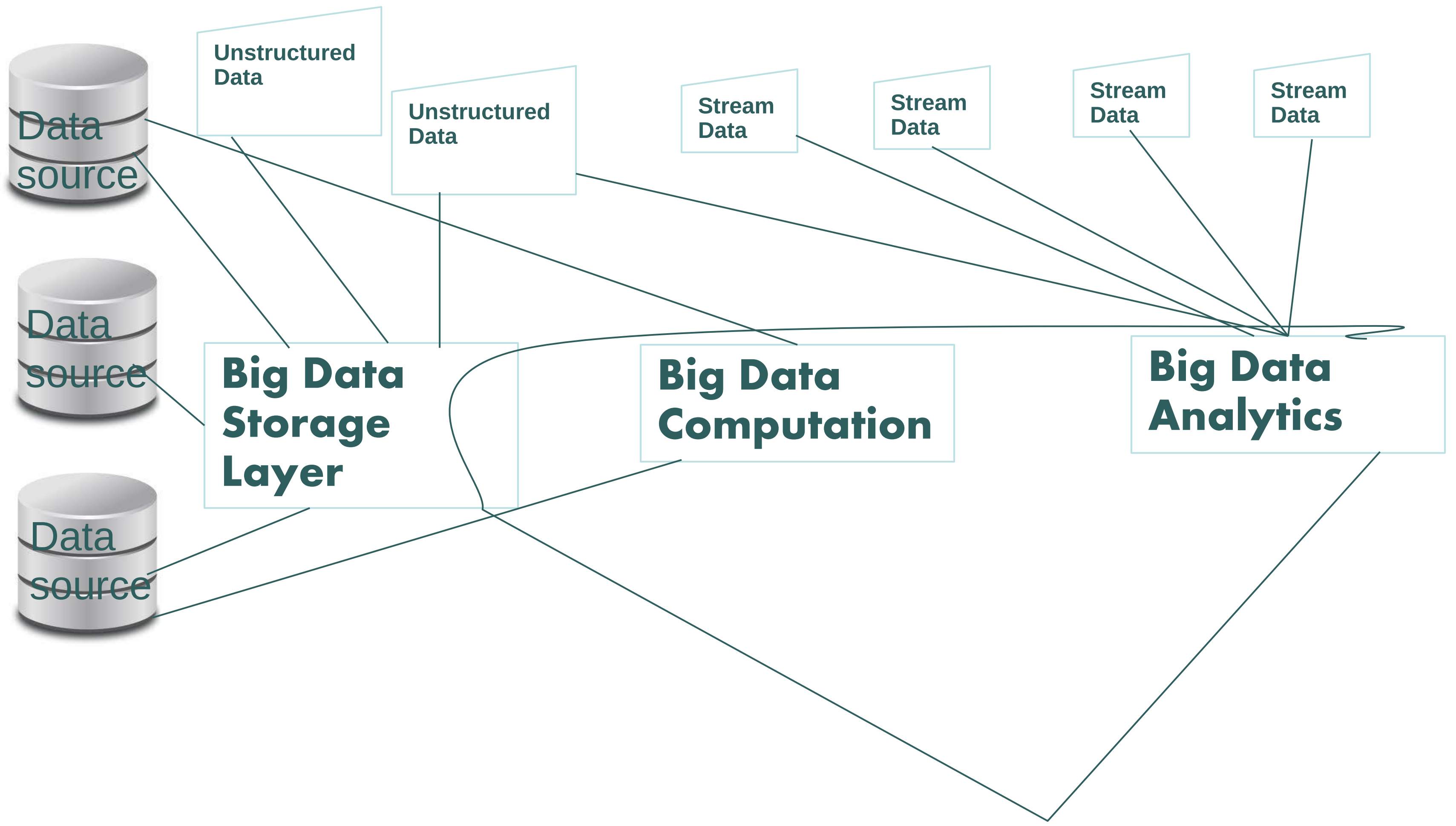
Voldemart System from LinkedIn

Latest NoSQL system from Industry

- Voldemart [RS12] Inspired by Amazon Dynamo
- NL-PA system like Dynamo – sacrifices consistency during normal operations as well as during partitions.
- PNUTS problem
  - Bulk insertion into ordered table (range partitioned – remember that hash partitioning is not order preserving)
    - affects throughput of serving systems – bulk insert operation is compute intensive
    - Uneven distribution of inserts across the range – some systems may be heavily loaded by the bulk insertion
      - Normal workload processing on those machines will be severely hit.
  - One solution – planning phase to gather statistics on ranges affected by bulk load [AS08]
    - Preparation phase may *split* ranges – so that resulting in small partitions post bulk insert
    - May also balance the ranges – involves data copying.
    - Optimization problem
  - Another solution [AS11] lies in using Hadoop for bulk loading – combine batch and serving systems
    - Map job to scan ranges in PNUTS
    - Checkpointing Hadoop – if a task fails, Hadoop will restart it from scratch.

[RS12] Roshan Sumbaly, Jay Kreps, Lei Gao, Alex Feinberg, Chinmay Soman, and Sam Shah, "Serving Large-scale Batch Computed Data with Project Voldemort" to appear in USENIX Conference on File and Storage Technologies (FAST) 2012.
[AS08] Adam Silberstein, Brian Cooper, Utkarsh Srivastava, Erik Vee, Ramana Yerneni, and Raghu Ramakrishnan. Efficient Bulk Insertion into a Distributed Ordered Table. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08), pages 765–778, New York, NY, USA, 2008.
[AS11] Adam Silberstein, Russell Sears, Wenchao Zhou, and Brian Cooper. A batch of PNUTS: experiences connecting cloud batch and serving systems. In Proceedings of the 2011 International Conference on Management of Data (SIGMOD '11), pages 1101–1112, New York, NY, USA, 2011.

Broad Focus



How to maximize efficiency, scalability of performing operations on Big-data – including storage, search, computation and 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.

Erasure Coding VS Replication [HW02]

	Fixed MTTF & Repair Epoch	Fixed Storage Overhead & Repair Epoch	Fixed Storage and MTTF (10 million machines, 10% down).
Erasure Coding	Much lower storage	MTTF ~ 10^{20} years	8 nines availability (with 32 fragments)
Replication	Much higher bandwidth	MTTF < 100 years	2 nines availability (with 2 replicas)

MTTF – mean time to failures
Repair epoch – protocol for repairing failed disks

[HW02] Hakim Weatherspoon and John Kubiatowicz. 2002. Erasure Coding Vs. Replication: A Quantitative Comparison. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems* (IPTPS '01), Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron (Eds.). Springer-Verlag, London, UK, 328-338.

Erasure Coding in Big-data Storage

- Microsoft Windows Azure File System (WAS) [CH12]
  - Users can store infinite data forever.
  - Uses EC – local reconstruction codes
    - *Lowers no. of EC fragments required for reconstruction.*
    - *Append only distributed file system*
    - *Active extents are replicated 3 times – once > 1 GB, ECed. Replicas deleted subsequently.*
    - *Performance trade-off between replication VS EC – fragments can be offline, network/node failures, reconstruction involves network bandwidth, computation time.*
- HDFS RAID – uses 4/5 EC special case of general EC.
  - Hadoop 503 – incorporated into code, not a general EC mechanism.
- Rethinking EC for cloud [OK12] – proposes rotated Reed-Solomon codes.

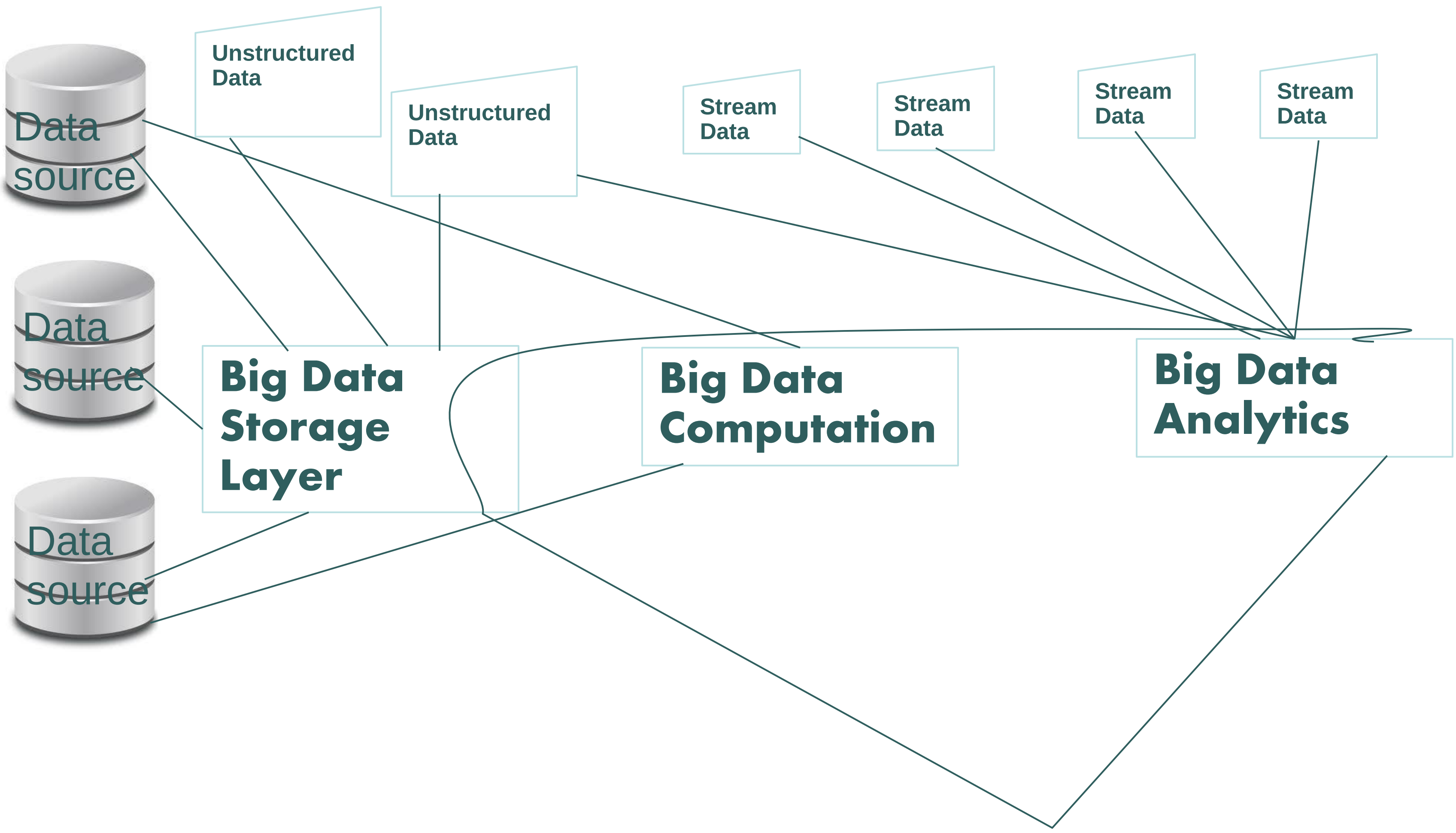
[CH12] Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, and Sergey Yekhanin. 2012. Erasure coding in windows azure storage. In *Proceedings of the 2012 USENIX conference on Annual Technical Conference* (USENIX ATC'12). USENIX Association, Berkeley, CA, USA, 2-2.
[OK12] Osama Khan, Randal Burns, James Plank, William Pierce, and Cheng Huang. 2012. Rethinking erasure codes for cloud file systems: minimizing I/O for recovery and degraded reads. In *Proceedings of the 10th USENIX conference on File and Storage Technologies* (FAST'12). USENIX Association, Berkeley, CA, USA, 20-20.

Big-data Storage Trends

- Hadapt: Distributed SQL queries – founder – Daniel Abadi – database star.
- Acunu – Modifying Linux kernel for custom storage – replace HDFS
- Massively Parallel Databases – Aster, Teradata
  - Big-data appliances.
  - Interesting startup – Paraccel.
    - *Analytics on top of MPPs.*
- Analysis of how MR workloads interact with storage layer [CLA12]
  - Log-normal distribution – huge no. of small files, very small no. of large files.
  - Mostly short-lived access to files (80% of access is within 5 days of creation.
  - High rate of change in file population – calls for tiered storage.

[CLA12] ABAD, C., ROBERTS, N., LU, Y., AND CAMPBELL, R. A storage centric analysis of Map-Reduce workloads: File popularity, temporal locality and arrival patterns. In Proc. IEEE IISWC (2012).

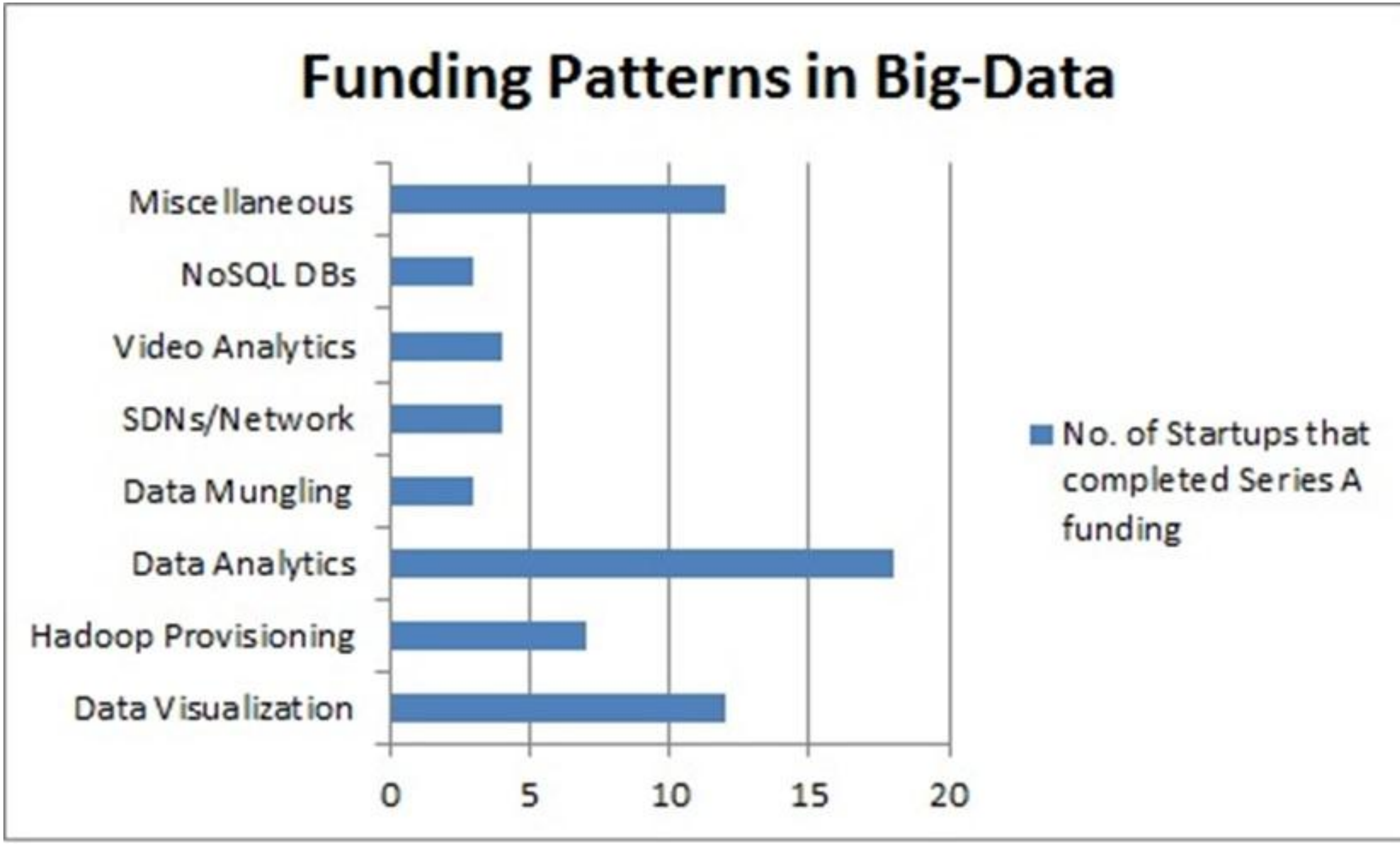
Broad Focus



How to maximize efficiency, scalability of performing operations on Big-data – including storage, search, computation and analytics.

Big-data Funding Pattern

Big-data = volume + velocity + variety + (value)



Thanks to Vishal Malik, a former colleague for this slide.

Big-data Funding Pattern

- **Data Analytics**
  - *Parstream, Bloomreach, Skytree, Platfora, Datameer, Revolution Analytics, Zementis, Versium, Cascading, Quibole, Palantir*
- **Data Visualization**
  - *Tableau, Jaspersoft, Microstrategy*
- **Hadoop Provisioning**
  - *Cloudera, MapR, Hortonworks,*
- **Video Analytics**
  - *Ooyala, TubeMogul, Video Breakouts, 3VR*
- **Software Defined Networks (SDNs)**
  - *Arista, Pronto Networks (Pica8), Nicira (acquired by Vmware), Contrail system (acquired by Juniper networks).*
- **NoSQL Databases**
  - *DataStax (Cassandra), 10gen (MongoDB)*
- **Data Munging –** converting raw data into a form that can be consumed.
  - *Trifacta (Joseph Hellerstein), Dataspora*

Hadoop Adoption Status: Sep 2012

- Enterprise level – not yet mainstream
  - Experimental – lot of big companies have their own Hadoop clusters including Sears, Walmart, Disney, AT&T etc.
  - Departmental production – not quite enterprise production yet?
- 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.

Future of Hadoop Adoption

- Enterprises
  - ETL for production
  - Hindrance – single cluster – Hadoop YARN is the way forward.
- Analytics
  - May not replace data warehouses
  - Real-time analytics is certainly the way forward.
  - Hadoop can be alternative to scale-out analytical RDBMSs (Vertica/VoltDB/SAP-HANA)
- Appliance market for Hadoop?
- Map-Reduce for iterative computations
  - Hadoop not currently well suited
  - Alternatives include Twister, Spark, HaLoop.
- Beyond Map-Reduce
  - Pregel from Google, built on top of Bulk Synchronous Parallel (BSP).

Suitability of Map-Reduce for Machine Learning

- Origin in functional programming languages (Lisp and ML)
- Built for embarrassingly parallel computations
  - The map function outputs key value pairs
    - *Map: (k1, v1) -> list(k2, v2)*
  - Reducer functions perform aggregate operations over the key
    - *Reduce: list(k2, list(v2)) - > list(v2).*
- Suitable for matrix multiplications, n-body problem and sorting problems – linear regression, batch gradient descent will work well – Mahout has these implementations.
  - Algorithms which can be expressed in Statistical Query Model in summation form – highly suitable for MR [CC06].
  - Linear regression, linear SVM, Naïve bayes etc. fall in this category.
  - Mahout has implemented only sequential version of logistic regression.
    - *Very hard to do in MR – inherently iterative*
    - *Training is very fast and in parallel, but basic algorithm is sequential.*

[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.
  - Communication cost just overwhelms computation time – it takes unreasonable time to run CG on MR.
- 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.

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) is not like that – power law graphs – small no. of highly connected people/vertices (popular) and large no. of low degree vertices.
  - 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)*.

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 - ivide 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)

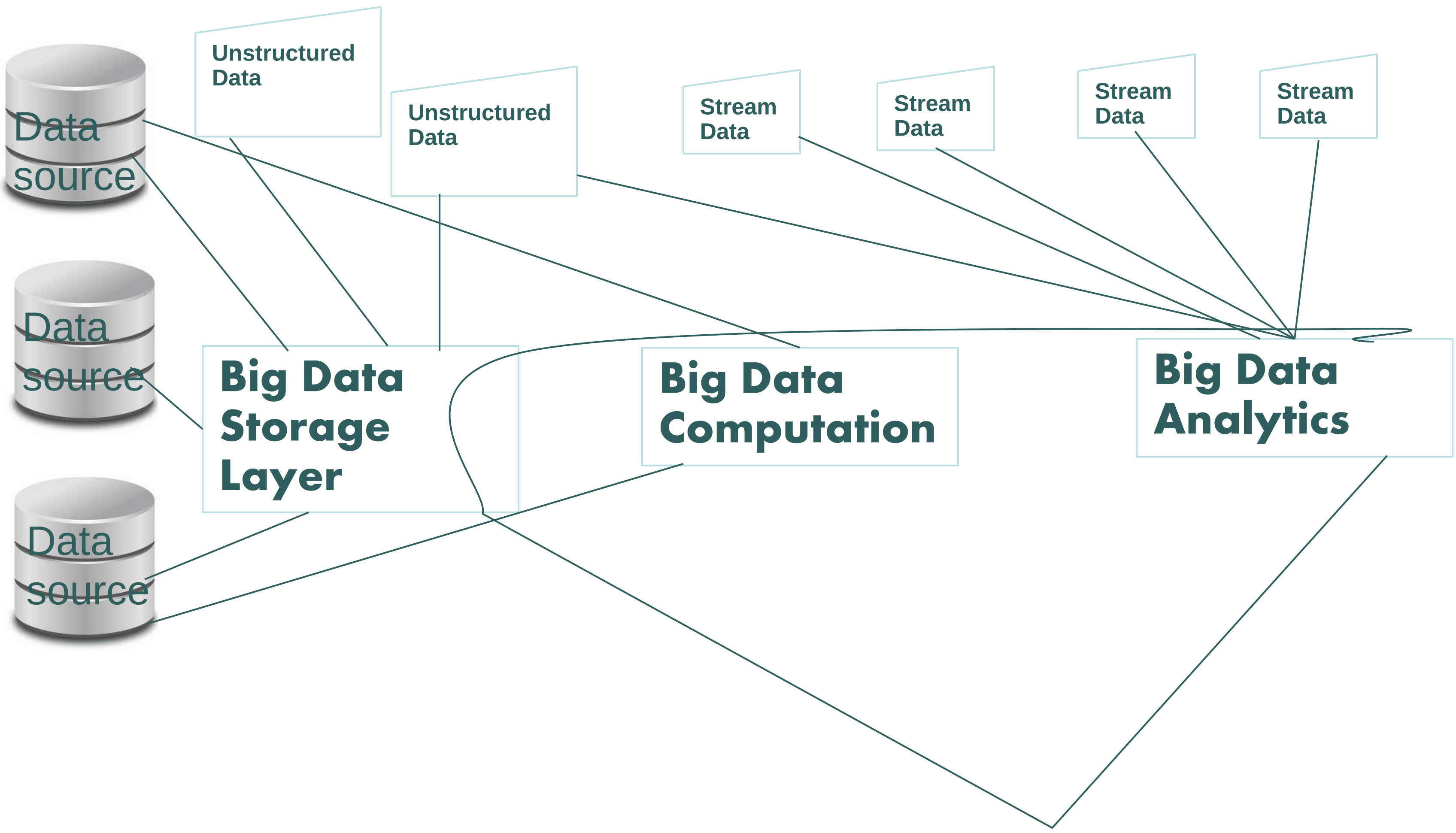
```
val spark = new SparkContext(<Mesos master>)
var count = spark.accumulator(0)
for (i <- spark.parallelize(1 to 100000, 12))
 { val x = Math.random * 2 - 1 val
y = Math.random * 2 - 1
if (x*x + y*y < 1) count += 1 }
println("Pi is roughly " + 4 * count / 100000.0)
```

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

Broad Focus



How to maximize efficiency, scalability of performing operations on Big-data – including storage, search, computation and analytics.

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.

Big-data Governance

Framework for Big-data governance

- Stakeholders, use-cases for Big-data. What are we trying to do with data?
- Data Provenance – keeping track of data
  - *What are the expected volume, velocity and variety of data – ensure Data Quality.*
  - *How was the data ingested? What was ingestion rate? Formats over time.*
  - *How is data to be stored? Retrieval SLAs. Information Lifecycle Management.*
  - *Create annotations on data – metadata – **data cataloguing***
- Analysis provenance – keeping track of analysis
  - *What questions were asked of the data? How was the analysis performed/validated? What was the accuracy of the analysis? How can it be improved? Where does human element fit in? Interpretation of big-data stats.*
- Security – how to ensure big-data is stored securely, safely (never lost)
  - *Privacy issues – especially with social data.*
- Data Architecture – how does NoSQL fit in with Hadoop? Which data gets where.
- Data risk management – things like disaster recovery
- Policy – specification, enforcement – operational aspects.

Cataloguing Big-data

Data Markets – Data as a Service (DaaS) – SoA based platforms.

- Characterizing data markets
  - *Domain, source, community, operating/pricing, query languages, data tools (visualizations).*
- Examples
  - *DataMarket (blog.datamarket.com) – search engine for statistical data*
  - *Timetric (http://timetric.com)*
    - *Governmental economic data – analyze stock portfolios.*
  - *Google Public Data http://www.google.com/publicdata/home)*
    - *Data Set Publishing Language (DSPL) – visualization of data.*
    - *Has governmental data sets – economic, social including World Bank and UN data sets.*
  - *Infochimps – well funded start-up*
  - *Freebase (http://freebase.com, Factual (www.factual.com), Kasabi.*

Thank You!

vijay.sa@impetus.co.in or

on LinkedIn at http://in.linkedin.com/in/vijaysrinivasagneeswaran

Blogs at blogs.impetus.com

Or on Twitter @a_vijaysrinivas.

Backup Slides

Big-data Governance

- Framework for Big-data governance – apply CMM for Big-data?
- Stakeholders, use-cases for Big-data. What are we trying to do with data?
- Data Provenance – keeping track of data
  - *What are the expected volume, velocity and variety of data – ensure Data Quality.*
  - *How was the data ingested? What was ingestion rate? Formats over time.*
  - *How is data to be stored? Retrieval SLAs. Information Lifecycle Management.*
  - *Create annotations on data – metadata.*
- Analysis provenance – keeping track of analysis
  - *What questions were asked of the data? How was the analysis performed/validated? What was the accuracy of the analysis? How can it be improved? Where does human element fit in? Interpretation of big-data stats.*
- Security – how to ensure big-data is stored securely, safely (never lost)
  - *Privacy issues – especially with social data.*
- Data Architecture – how does NoSQL fit in with Hadoop? Which data gets where.
- Data risk management – things like disaster recovery
- Policy – specification, enforcement – operational aspects.

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)
```