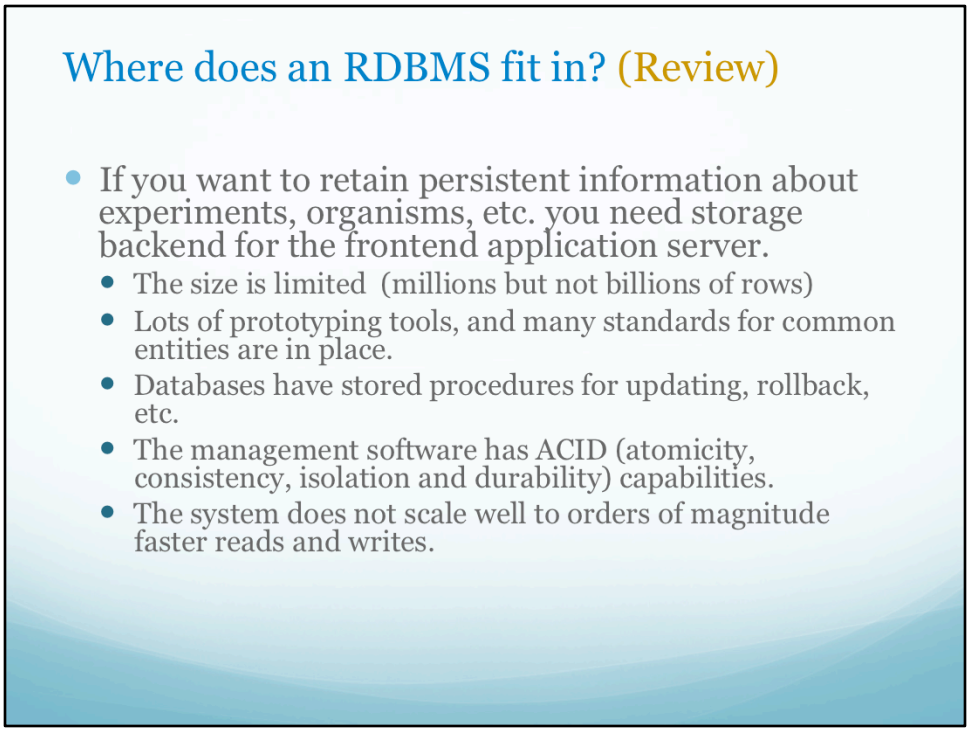
BINF 8211/6211
Design and Implementation of Bioinformatics Databases
Lecture #22

Dr. D. Andrew Carr
Dept. Bioinformatics and Genomics UNCC
Spring 2016

Today Topics

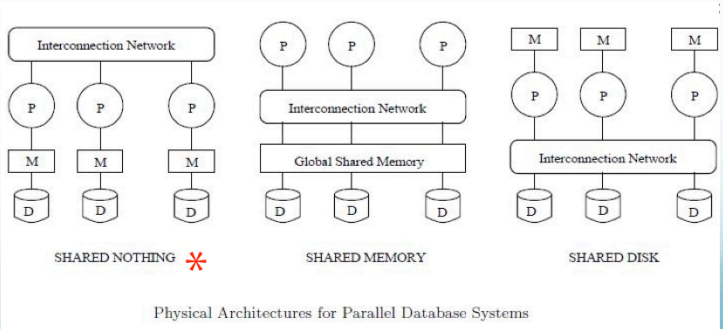
- NoSQL – Map Reduce
- LIMS
- Schedule of Dates for Presentations



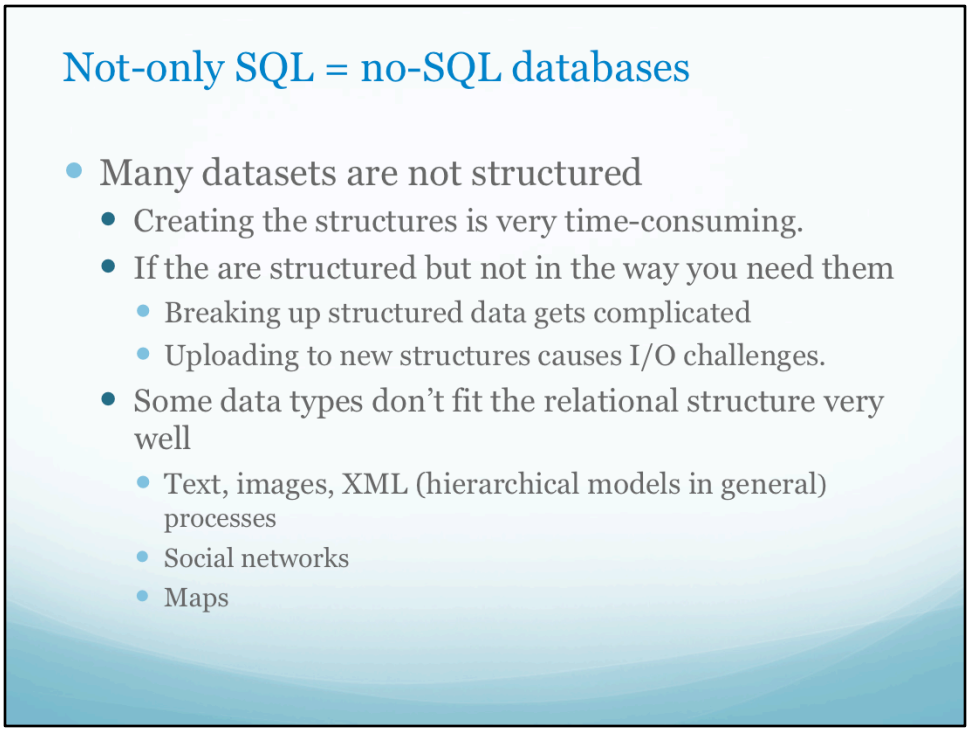
ACID is a term used to judge RDBMS applications.
Atomicity states that database modifications must follow an all or nothing rule. Each transaction is said to be atomic. If one part of the transaction fails, the entire transaction fails. It is critical that the database management system maintain the atomic nature of transactions in spite of any DBMS, operating system or hardware failure.
Consistency states that only valid data will be written to the database. If, for some reason, a transaction is executed that violates the database s consistency rules, the entire transaction will be rolled back and the database will be restored to a state consistent with those rules. On the other hand, if a transaction successfully executes, it will take the database from one state that is consistent with the rules to another state that is also consistent with the rules.
Isolation requires that multiple transactions occurring at the same time not impact each other' s execution. For example, if Joe issues a transaction against a database at the same time that Mary issues a different transaction, both transactions should operate on the database in an isolated manner. The database should either perform Joe s entire transaction before executing Mary' s or vice-versa. This prevents Joe 's transaction from reading intermediate data produced as a side effect of part of Mary s transaction that will not eventually be committed to the database. Note that the isolation property does not ensure which transaction will execute first, merely that they will not interfere with each other.

Parallel DBMS essentials

- Relational tables and SQL are supported.
  - Tables are partitioned over nodes in a cluster
  - An optimizer translates SQL into a query plan for execution across the multiple nodes.
- Reorganize data at load time
  - Each attribute might be stored separately so retrieval does not require parsing from unused attributes
  - Data abstraction allows reuse of queries and slight modifications.
  - Multiple indices per table can be created.
- The computational code is sent to the data – the optimizer also balances computational workloads



Physical Architectures for Parallel Database Systems



We have talked about ways to tag and represent relatively unstructured data, or perhaps over-structured data would be another way to frame it – we have data that is related in multiple interesting ways, and picking just one perspective may be tough or too limiting.

Transactional procedures depend on timely access to data – posing a well-formed query can be time-consuming.

Hadoop runs on many types of architecture and has been built to withstand failures within clusters, is meant to scale easily and to be simple to work with.

> You can query data sets, just not with a consistent DML, but the consistency is the main shortcoming, but this is true of most distributed systems. You can have concurrency or consistency but not both.

Basic Availability. The NoSQL database approach focuses on availability of data even in the presence of multiple failures. It achieves this by using a highly distributed approach to database management. Instead of maintaining a single large data store and focusing on the fault tolerance of that store, NoSQL databases spread data across many storage systems with a high degree of replication. In the unlikely event that a failure disrupts access to a segment of data, this does not necessarily result in a complete database outage.

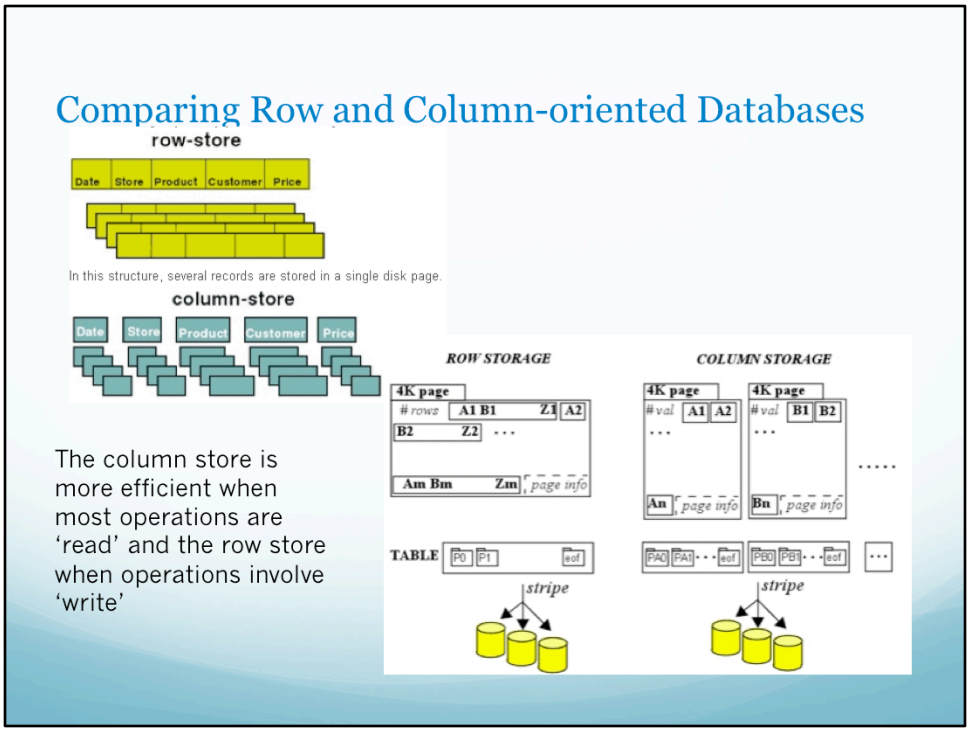
Soft State. BASE databases abandon the consistency requirements of the ACID model

Column Oriented Databases

- Data is stored by type in columns instead of multi-attribute rows.
  - Used for data warehouses and clinical data
  - Fast computation of aggregates of similar data items
  - Large data of singular type works well
  - Used for when functions need to be applied across entire data sets
    - Analytical data processing
  - Row based are good at singular items not entire sets.
    - Single transaction processing

Column Oriented DBMS

- Row will look like
  - 001:10, Protein, Gub-bacillus, 134
  - NOTE: RDBMS uses indexing help with column orientation and improve speed.
- Column
  - 001:134
- Column Databases are heavily indexed
  - Greater overhead when writing new data into the DB.
    - Faster when entire column is replaced with new. Single point entyry or change is slow.
- Smaller chunks of data that mean fewer disk reads.
- Data redundancy altered by pointer storage.
  - 134: 001, 013



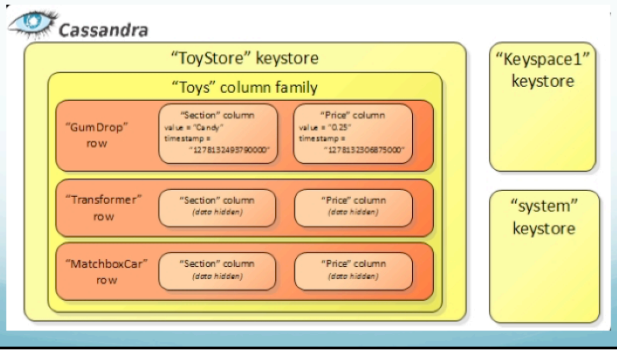
But in a column store several entries will need to be modified to change a record (multiple I/O)

Compression is better for column-store.

Column is better for range queries and rows for random access because indices are configured at the record-level.

Apache Cassandra – a column oriented no-SQL BigTable-like database

- Column: a tuple that is a triple containing a name, a value and a timestamp
  - The name and value are both binary (byte[]) and can be of any length.
  - SuperColumn is a tuple with a binary name and value which is a map containing an unbounded number of columns that are keyed by the Columns name
  - See http://arin.me/blog/wtf-is-a-supercolumn-cassandra-data-model



Map Reduce

- Map Reduce
  - Associated with first generation Google
  - Used for processing Big data sets
  - Advantage if and only if multi-threaded.

- Process:
  - **"Map" step:** Each worker node applies the "map()" function to the local data, and writes the output to a temporary storage.
  - **"Shuffle" step:** Worker nodes redistribute data based on the output keys (produced by the "map()" function), such that all data belonging to one key is located on the same worker node.
  - **"Reduce" step:** Worker nodes now process each group of output data, per key, in parallel.

From Wikipedia

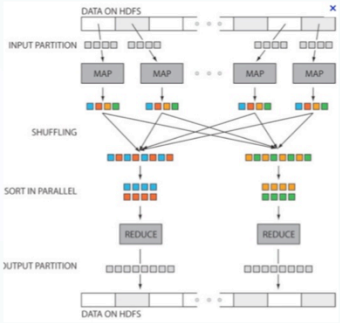
Map Reduce Continued

- Application
  - distributed pattern-based searching
  - distributed sorting
  - web link-graph reversalS
  - Singular Value Decomposition
  - web access log stats document clustering
  - Machine learning
- Environment
  - Anything multinode or multithreaded
    - Cloud based...
- MapReduce's stable inputs and outputs are usually stored in a distributed file system.

Map Reduce Continued

- Google
  - Map reduce was used to create WWW indices
  - Now uses Percolator, Flume and MillWheel
    - These removed the need to rebuild the complete index
    - Live result streaming

MapReduce DataFlow



- Input reader
  - Reads from the distributed file system
  - Generates the (key,value) pairs
- Map function
  - Generates a zero or more pairs (the data types often differ from original)
- Partition function: the application allocates the map function output to a specific reducer
  - The partition function returns the index of the requested reduce
  - The data is exchanged.
- Comparison function sorts data
- Reduce function: called once for each unique (sorted) key to produce 0 or more outputs
- Output function: write to stable storage
- A thorough explanation is at http://developer.yahoo.com/blogs/hadoop/posts/2009/08/the_anatomy_of_hadoop_io_pipel/

Input reader – divides the data into appropriate splits and assigns each to a map function.

Map function – processes a list of (key,value) pairs

Partition function – the application allocates the map function output to a specific reducer – the function is giventhe key and the number of reducers, returns the index for the requested reduce family. The data is exchanged between nodes, to it moves from the map node to a shared where it will be reduced (a network is required).

Comparison function an application pulling the input for each Reduce from the machine where Map ran and sorting it

Reduce function works on a key value set in the sorted order and returns output for each .

The output functino writes the results to stable storage in the HDFS.

MapReduce Steps

- The Map step: the master node takes the input and divides it into smaller subsets, sending those out to worker nodes.
  - After completing the task the worker node passes the answer back to the master
- The Reduce step: the master node collects all of the answers and combines them to give the final output.
- Logically, if data exists in a (key,value) pair, Map takes one such pair from one data domain and returns a list of pairs in a different domain
  - Map(k1,v1) → list (k2, v2)
  - Reduce (k2, list(v2)) → list(v3)

```
void map(String name, String document):
   // name: document name
   // document: document contents
   for each word w in document:
       EmitIntermediate(w, "1");

void reduce(String word, Iterator partialCounts):
    // word: a word
    // partialCounts: a list of aggregated partial counts
    int sum = 0;
    for each pc in partialCounts:
       sum += ParseInt(pc);
    Emit(word, AsString(sum));
```

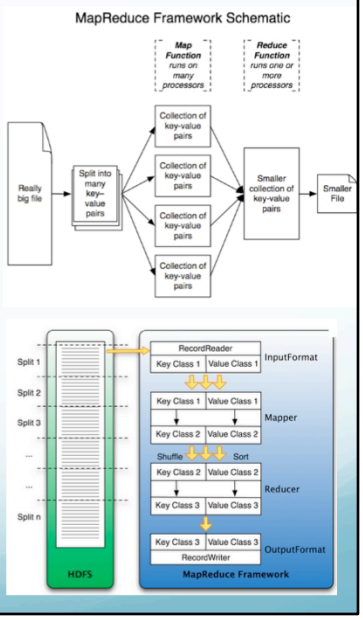
NoSQL includes filter/map reduce pipeline
If the mapping operation is independent of the others then the maps can be performed in parallel
If all outputs of the mapping operation that share the same key are presented to the reducer at the same time a set of reducers can operate in parallel.

The Map function is applied in parallel to every pair in the ouput data set, resulting in a list of pairs for each call. All pairs with the same key from all lists are grouped together so each different key ersults in a group. A list of key, value pairs becomes a list of values.

If the implementation is distributed then you have to have a distributed file system to make this work.
Note that again all output records from the Map phase with the same hash value are consumed by the same Reduce instance, regardless of which Map instance produced the data. Each Reduce processes or combines the records assigned to it in some way, and then writes records to an output file (in the distributed file system), which forms part of the computation's final output.
The input data set exists as a collection of one or more partitions in the distributed file system. It is the job of the MR scheduler to

MapReduce

- The MapReduce engine consists of one JobTracker
  - Knows where data is within the HDFS
  - Pushes the work out to TaskTracker nodes
    - Spawns a JVM process to keep the job running in the event of a crash (radar heard by Jetty)
    - Has 4 slots that can do either map or reduce tasks
  - First In, First Out – order of queued data manipulation
    - The first data added to the list will be the first data used (taken off) in processing
    - All Map instances use the same hash function so all output records with the same hash value are stored in the same output file.

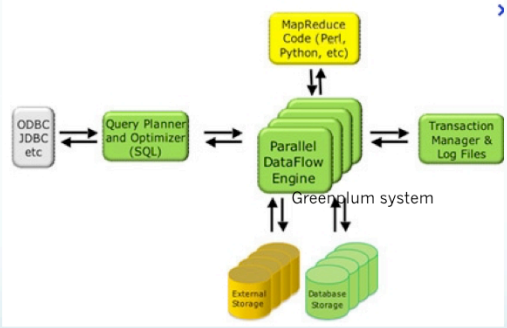


A client submits a job request/
The Job Tracker pushes the work to available Task Tracker nodes with a constraint on the locality to be as close as possible to the data needed (known in rack-aware file systems like HDFS)

The Map function reads a set of "records" from an input file, does any desired filtering and/or transformations, and then outputs a set of intermediate records in the form of new key/value pairs. As the Map function produces these output records, a "split" function partitions the records into R disjoint buckets by applying a function to the key of each output record. This split function is typically a hash function, though any deterministic function will suffice. Each map bucket is written to the processing node's local disk. The Map function terminates having produced R output files, one for each bucket. In general, there are multiple instances of the Map function running on different nodes of a compute cluster. We use the term *instance to mean a unique running invocation of either the Map or* Reduce function. Each Map instance is assigned a distinct portion of the input file by the MR scheduler to process.

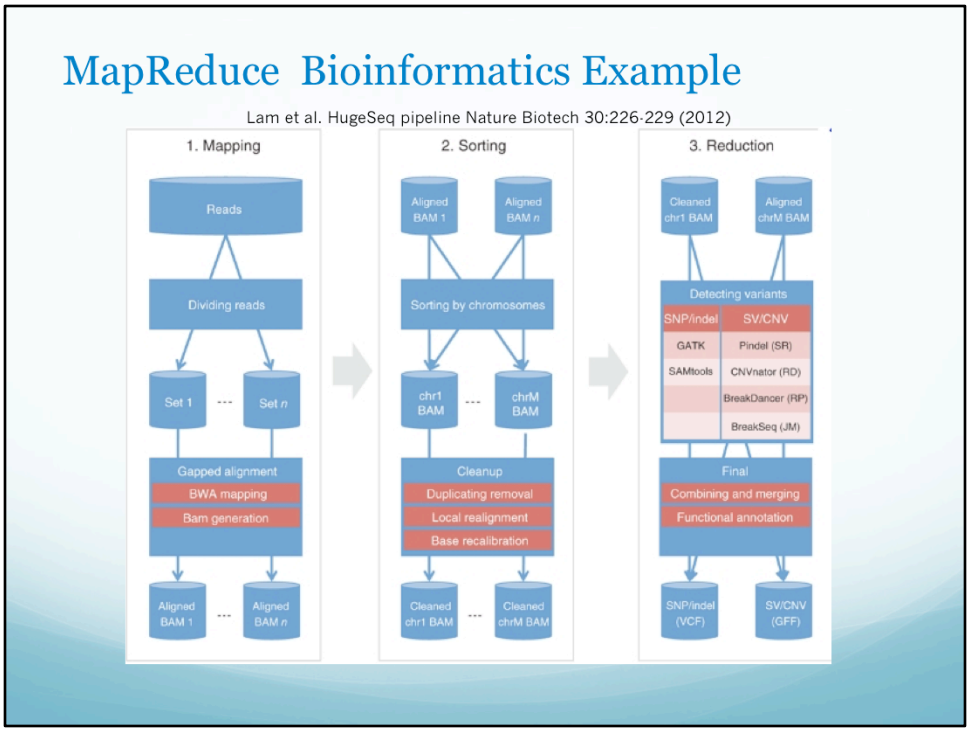
Parallel RDBMS VS MapReduce strategies

Cluster computing: when computing tasks are highly parallel and data sets are large.
Another solution: parallel DBMS (e.g. Teradata, ParAccel Bubba, Arbre etc.).

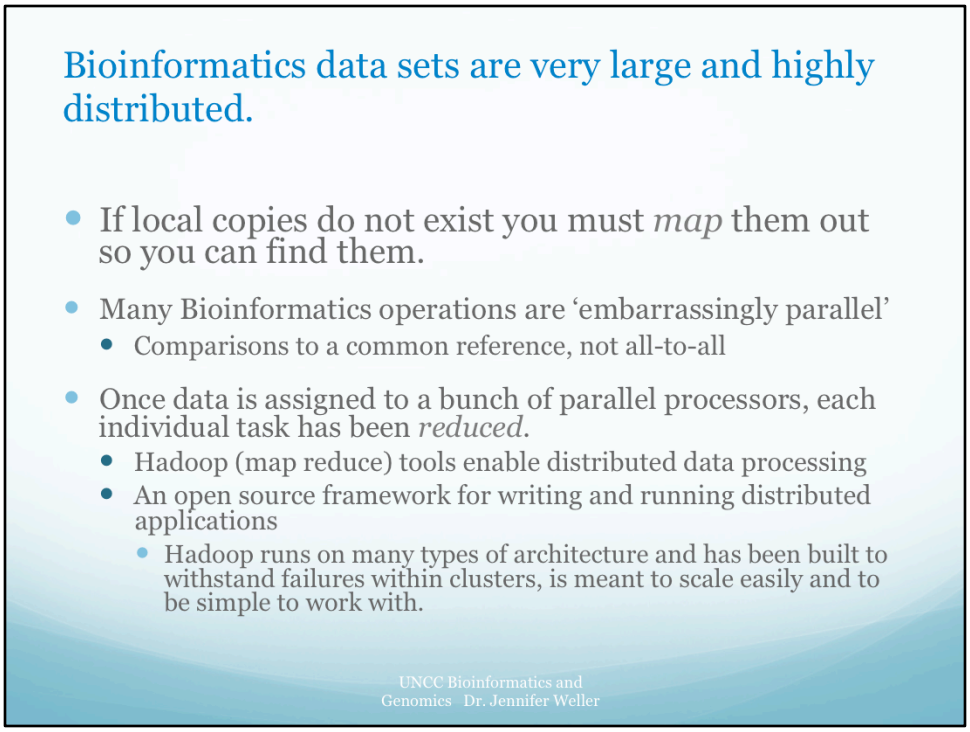


For a general review see: http://www.cubrid.org/blog/dev-platform/database-technology-for-large-scale-data/

You want to get to the answer quickly.
You have to create applications for these distributed systems.
The pDBMS have high-level programming environments.
Almost any task can be defined as either a set of queries (assuming user-defined functions are allowed) or MR jobs.



For simple and large data sets that are mostly about relative location, this is an efficient way to store the data and just retrieve the subsets needed for some operations.



The BioMart system and the DAS systems both provide ways to find the things you want – BioMart collects elements for you and DAS tells you the location.

What we are really saying is that you want to do the same thing over and over, keeping the results. Places where this does not work is where you are building a model and the whole data is used and the model has to be in memory in order to test hypotheses about data elements (many statistical tests are like this). A relational database with the associated application software is like this – keeping track of the intended structures and serializing in a way that you can keep track of takes a very large amount of extra memory. As the data sets get larger you want to use the space for the data itself.

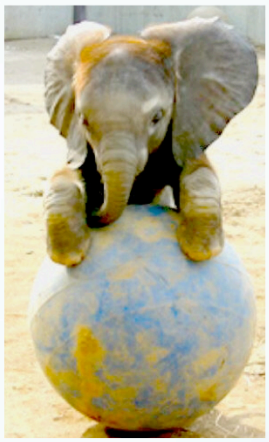
A solution is to maintain parallel systems and have a mapping rule. One system is the raw data and a location index, and the other system has a rule for using the location index depending on what you want to retrieve.

Comparing Parallel RDBMS and MapReduce strategies

- Differences: whether a schema is required, how indexing is performed, how data is distributed, how queries are executed.
- Similarity: run on 'shared nothing' set of computers (but do share network), divide data into partitions allocated to different nodes.
  - Raw memory and disk access are local
  - Processes can't interfere with each other – few resources are shared
  - With little network traffic scale-up is easier to estimate
    - Complex relational queries are close to linear in speedup and scaleup

Hadoop

- Apache Hadoop is a software framework
- Scheduling of (large amounts of) work is the central concern: each filesystem provides the network switch of the worker node.
  - Ideally work is run on that node (or one on the same switch).
  - HDFS is the Hadoop Distributed File system
- Architecture: master node
  - JobTracker
  - TaskTracker (* function a worker node can have)
  - NameNode (* function a worker node can have)
  - DataNode



The goal is to support data-intensive distributed applications.
Thousands of independent calculations
Peta-bytes of data.
It is written in Java.

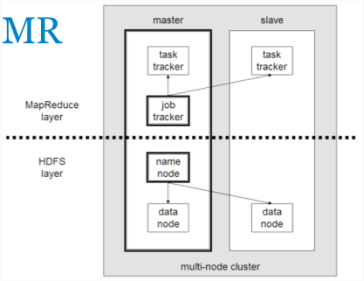
Location awareness must be published.
HDFS avoids the same location when replicating data.

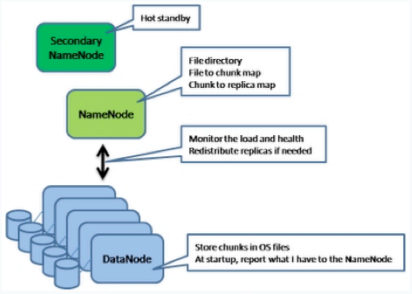
A dedicated NameNode server will host the filesystem index, a secondary NameNode generates snapshots of the memory structures

Hadoop

- Burden is on the implementer to understand the data.
- Organization must be pre-planned.
- NoSQL – All queries to the system must be coded
- Requires distributed compute system management skills

Hadoop/MR and the HDFS

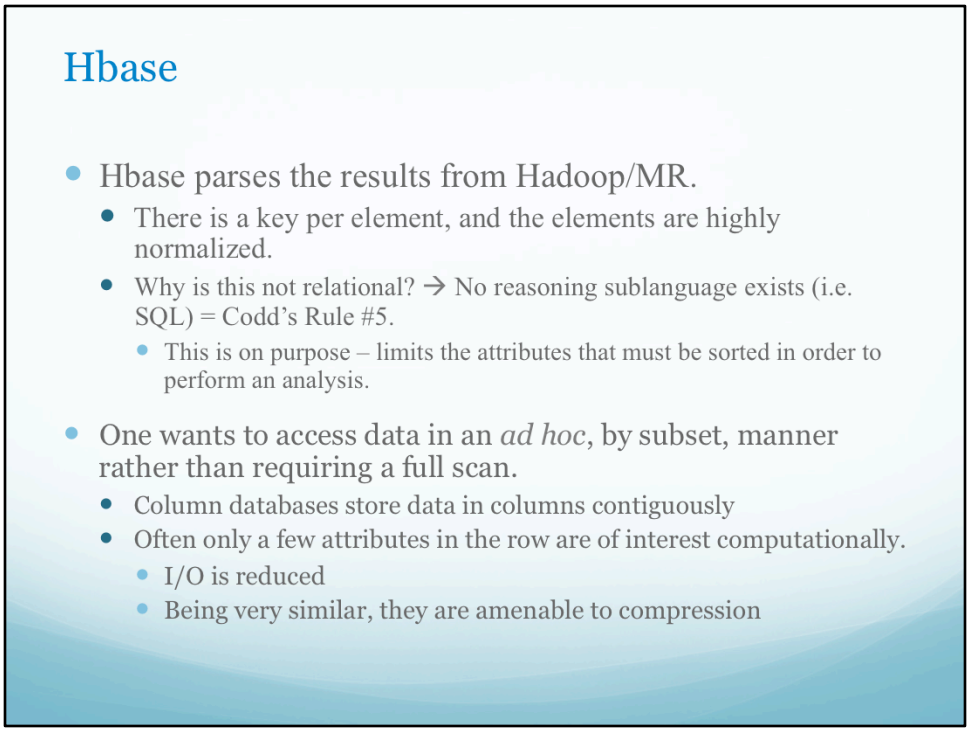




- Each Datanode serves up blocks of data over the network (HDFS_specific protocol)
- The Primary NameNode is the source of metadata storage about and management of the data files.
- The JobTracker schedules map/reduce jobs to the TaskTrackers, with a data location consideration built in.

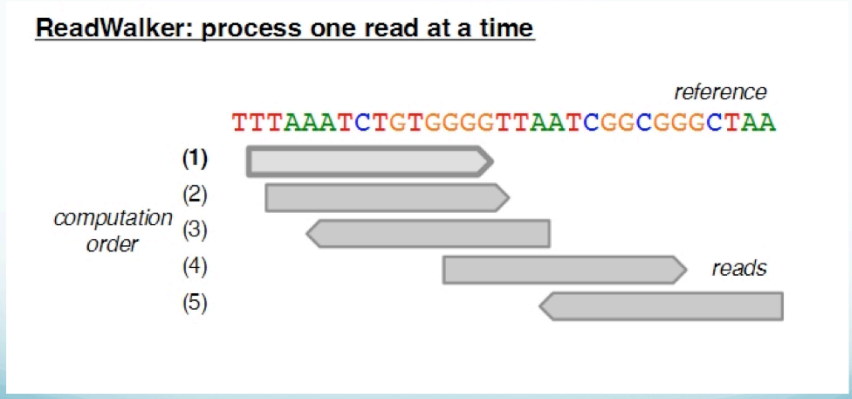
UNCC Bioinformatics and Genomics Dr. Jennifer Weller

HDFS is the Hadoop filesystem
Filesystem uses tcp/ip for communication
Clients use Remote Procedure Calls to communicate with each other.
Large blocks of data are stored across multiple hosts (no RAID needed).
Default is a replication level of 3, two on one rack and the third elsewhere.
It has facilities to change to replicate nodes when a main node goes down, and to take frequent snapshots of the main namenode so that if the primary NameNode goes down a recent update is available.
NameNode is the point of management for metadata storage/management. HDFS Federation allows a larger number of small files to be handled.
The filesystem cannot be mounted through existing OS. File access is through the native Java API or the Thrift API (allowing lots of other languages) os a user interface web application.



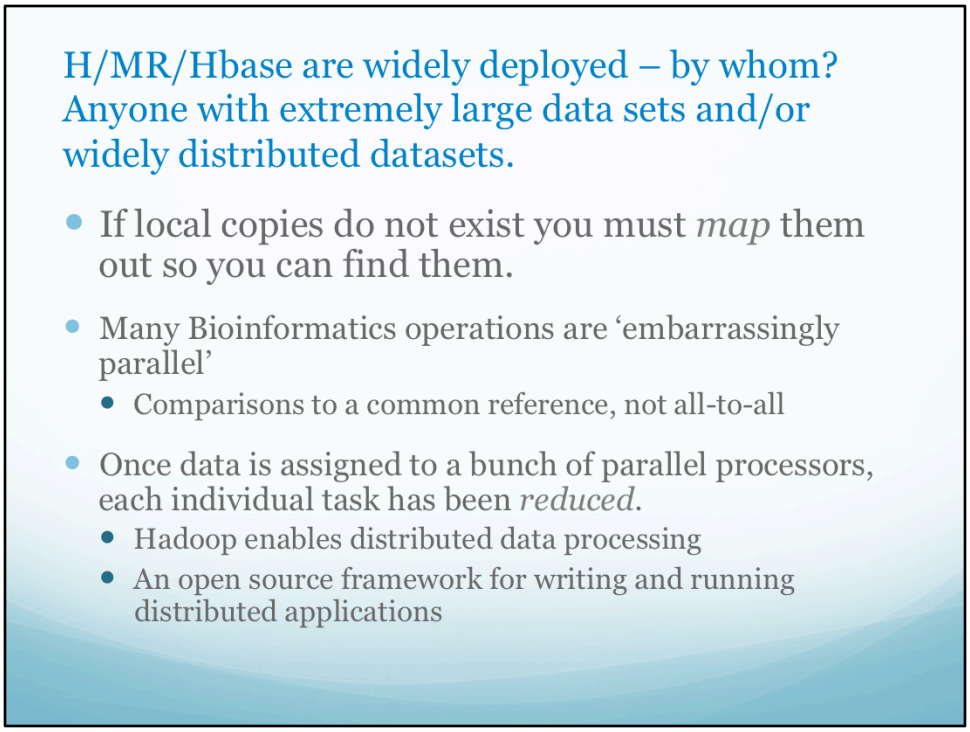
Hadoop and MapReduce produce a large chunk of results, which you then have to parse: random access to part of the data is needed.

Read Walker allows a computation to be performed on each read in turn (looping to next).



Set a metadata tag, search for specific reads to be excluded, identify indels

Existing algorithms: CycleQualityWalker, CountReadsWalker



Hadoop runs on many types of architecture and has been built to withstand failures within clusters, is meant to scale easily and to be simple to work with.

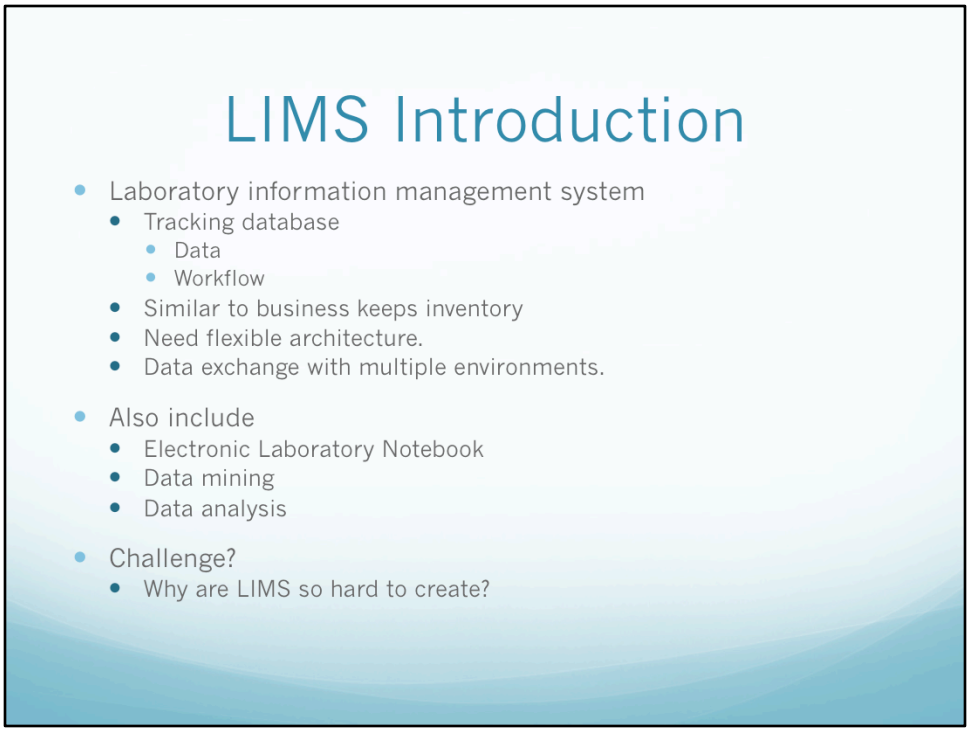
The GATK from the Broad Inst. Uses a MR framework.

- Genome Analysis Toolkit – a library of applications for handling NGS data.
  - Java code base
  - Map/Reduce framework
  - Support for BAM alignment files (Picard)
  - BED interval file format (Tribble)
  - Variance data (VCF format)
- Explanations and tutorials at Blue Collar Bioinformatics (and the Broad) http://bcbio.wordpress.com/

Did I waste my time learning about RDBMS?

- DeWitt and Stonebreaker published results of a benchmark study comparing Hadoop's MapReduce to RDBMS approaches for certain kinds of problems.
  - Databases come out ahead for complex processing and for use by many groups for multiple purposes.
  - MapReduce is a good extract-transform-load system

- M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, A. Pavlo, and A. Rasin, "MapReduce and Parallel DBMSs: Friends or Foes?," *Communications of the ACM*, vol. 53, iss. 1, pp. 64-71, 2010.
- A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, "A comparison of approaches to large-scale data analysis," in *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, New York, NY, USA, 2009, pp. 165-178



LIMS require flexibility.

Standardized processes are simple to implement. When new workflows and protocols are added to the system then new data needs to be able to be handled by the system.

Generic data types are needed. Each lab has its own requirements.

Instrument integration.

Systems need to communicate with other databases.