

BINF 8211/6211
Design and Implementation of
Bioinformatics Databases
Lecture #21

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

Today Topics

- Data and Ethics
- NoSQL
- Schedule of Dates for Presentations

ACID

- What is the RDBMS ACID test?

- Data management includes data gathering, storage and retention. Affecting this are issues of
 - Ownership
 - Collection
 - Storage/Security
 - Protection of subject confidentiality and data sharing
- When a clinical trial is funded by the NLM you must include a data safety and monitoring plan in your application. Generally the local IRB must approve it.

RCR and biomedical databases

- Original laboratory data must be retrievable. The person who conducted/oversaw the research does not always own the data.
 - A funding source may claim ownership and grant access and use rights at its discretion
 - Grants usually mean that the institution where research is conducted has ownership and control of the data.
 - Contracts usually involve transfer of all products (including data) back to the funding source.
 - A data source may demand some reach-through for ownership
- Custody of the data usually lies with an investigator – an institution may own the data (have full access and use).

1. To answer scientific questions you must have accurate and detailed records, organized to understand who collected them and under what circumstances – for human subjects it must be particularly clear that all research was conducted according to the IRB protocols, and for all research that IBC protocols were followed (and IACUC as warranted).
2. Most cases of alleged research fraud have resulted from incomplete (and unverifiable) record keeping. In this case the misconduct has to do with the process and not how the results were used and what conclusions were claimed.
3. People may want some ownership of data collected from samples they contribute. A country with unique natural resources may want ownership of data created from those samples. A database whose contents are integrated into a larger data set may demand some ownership based on the value added of their contribution.
4. Researchers must be sure they are clear on the issues of ownership and granted rights of access to data before using data. Those accessing data repositories must be aware of restrictions and obligations. Those generating data must be aware of obligations and restrictions.
5. Custody of all original data must be retained by the unit in which they are generated unless forbidden by the terms of the study. If a data collection facility is not a secure site for data storage then at least summaries must be maintained by the study owner. (for example in a hospital, patient records are not kept in a research or clinical lab).

- Data Collection
 - It is assumed that the method is appropriate.
 - The method of recording results must be subject to verification of accuracy.
 - Records related through experiments and studies must be unambiguously linked:
 - Provenance standards are met (dates, file locations and names are unambiguous, transformations are documented and can be identified, etc.).
 - There are guidelines for data management at this level at <http://ori.hhs.gov/education/products/clinicaltools/data.pdf>.
 - All appropriate authorizations must be documented and linked to the results.

1. Data collection may mean data generation or finding disparate on-line sources. Attribution of every element is required and you must be able to verify it.
2. Transcription of data by hand is notoriously subject to error, as is using default file names from a capture program and 'remembering' which is relevant if the basic data type is very similar to that of others.

- Data Storage and Security
 - Documents are safe from physical damage or misappropriation
 - Files are backed up and the backup device is safe from physical damage or inadvertent erasure.
 - Data subject to privacy restrictions must be accessible only to authorized personnel
 - Hack-proof if electronically stored (main storage on removable media so only ever in temporary memory on the analysis computer, that is disconnected from the network).
 - Never exchanged over a network, or only upon strong encryption (key is not exchanged over the network).
- Data Sharing
 - Many funding agencies and publication boards require submission of all data used to distill results and interpret them.
 - Developing a method to block associating data elements with individuals is a research area – some level of scrambling may be allowed so long as it is not irreversible if an audit is required.
 - Retention: usually the data must be available for a specific period of time after the final publication by the data collection group.

1. Lab notebooks are increasingly giving way to electronic notebooks – backup is not always considered though. Once this becomes a file or files – how do you locate them? Generally these are easier to over-write than paper note books so more care is needed to develop a system for retaining records.
2. Often a separate hard drive is acquired for a given project, labeled and stored in a safe place (cool, dark, fire and flood protected with limited personnel access, ideally one person who can check out the contents), often off-site from the primary research location.
3. Depending on the sensitivity of the data but also what could be done with the data in combination with other data sources. If you have all of your sequence data in one repository but no address or phone information, you microsatellite data will be included. If you have a relative whose DNA is in a forensic database that relationship can be discovered, then a Facebook or other social media site might indicate where you live and what your name is. So then a lot of information about your genetic history and disease predilections could be inferred. If only the gene sequences that turned out to be relevant to a study were published, and none contains a unique mutation and none includes a microsatellite used in forensics you might be OK, although still grouping a set of sequences together may well provide a unique signature. Separating each gene into a group with a common association is not a guarantee you won't be listed in a pool of possible contributors (there are only so many combinations of genotypes possible). This is

- Personal privacy and genetic sequence – see the study by Yaniv et al in Science (18Jan 2013, p 321).
 - Partial Y-chromosome sequence + age + state of residence and some shared genealogy data = identity of you AND your family (including some you didn't know about).
- Approaches to this problem
 - Remove identifying segments or separate them in a way that requires a key to reassemble.
 - Add a lot more data about studies and researchers, building a trust relationship by which study volunteers know a lot more about the environment in which their data is used.
 - Currently you must demonstrate that you have a secure system for storing and backing data up, and to prevent unauthorized access. Each University/institution must have a data plan and individuals must show they are aware of it and know how to comply with it and have the resources/access to carry out the requirements.

NoSQL

- Non SQL or non relational or not only SQL
 - Any system that provides a manner in which to store and retrieve data that is not modeled in a relational manner
 - Stems from Web 2.0
 - Facebook, Google, Amazon
 - Used for big data and real time web

NoSQL -- continued

- RDBMS – are not easily distributed
- NoSQL is more fluid and semi-structured allowing for distribution and greater “flexibility”
- NoSQL systems often forgo consistency for speed and partition tolerance.
 - Offer conceptual “eventual consistency”
 - Nodes are often behind in update
 - Thus can have read and data loss errors
 - Data consistency is a challenge in any distributed computation environment.

NoSQL

- Types and examples of NoSQL databases [[edit](#)]
- There have been various approaches to classify NoSQL databases, each with different categories and subcategories, some of which overlap. What follows is a basic classification by data model, with examples:
 - **Column:** Accumulo, Cassandra, Druid, HBase, Vertica
 - **Document:** Apache CouchDB, Clusterpoint, Couchbase, DocumentDB, HyperDex, Lotus Notes, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB
 - **Key-value:** Aerospike, CouchDB, Dynamo, FairCom c-treeACE, FoundationDB, HyperDex, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, Berkeley DB
 - **Graph:** AllegroGraph, InfiniteGraph, MarkLogic, Neo4J, OrientDB, Virtuoso, Stardog
 - **Multi-model:** Alchemy Database, ArangoDB, CortexDB, FoundationDB, MarkLogic, OrientDB

NoSQL

- **Key-value stores**
 - Key-value (KV) stores use the associative arrays.
 - Each pair occurs at least once
 - Lexicographic order is possible
 - Increase read and search speeds
 - Small segments can be searched in key ranges.
 - Eventual Consistency or Serializable
 - Examples: Oracle NoSQL, Redis

NoSQL

- **Document store**
- *Document-oriented database and XML database*
- The central concept of a document store is the notion of a "document".
 - Document definition varies
 - XML, YAML, JSON,
- API included for searching and maintaining.
- Each system is organized and grouped differently..
- Directory hierarchy
- Compared to relational databases, for example, collections could be considered analogous to tables and documents analogous to records. But they are different: every record in a table has the same sequence of fields, while documents in a collection may have fields that are completely different.

NoSQL

- **Graph**
- Nodes with infinite numbers of relations.
- Can include
 - Social networks
 - Public transportation
 - Maps
 - Network topologies

Where does an RDBMS fit in?

- If you want to retain persistent information about experiments, organisms, etc. you need storage backend for the frontend application server.
 - The size is limited (millions but not billions of rows)
 - Lots of prototyping tools, and many standards for common entities are in place.
 - Databases have stored procedures for updating, rollback, etc.
 - The management software has ACID (atomicity, consistency, isolation and durability) capabilities.
 - The system does not scale well to orders of magnitude faster reads and writes.

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.

Not-only SQL = no-SQL databases

- Many datasets are not structured
 - Creating the structures is very time-consuming.
 - If they are structured but not in the way you need them
 - Breaking up structured data gets complicated
 - Uploading to new structures causes I/O challenges.
 - Some data types don't fit the relational structure very well
 - Text, images, XML (hierarchical models in general) processes
 - Social networks
 - Maps

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.

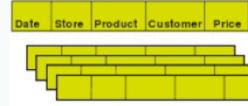
You can query data sets, just not with a consistent DML, but the consistency is the main shortcoming, but this is true of most distributed systems. You can have concurrency or consistency but not both.

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

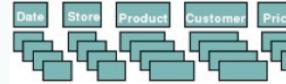
Comparing Row and Column-oriented Databases

row-store



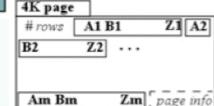
In this structure, several records are stored in a single disk page.

column-store

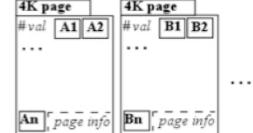


The column store is more efficient when most operations are 'read' and the row store when operations involve 'write'

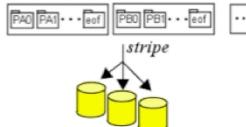
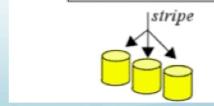
ROW STORAGE



COLUMN STORAGE



TABLE



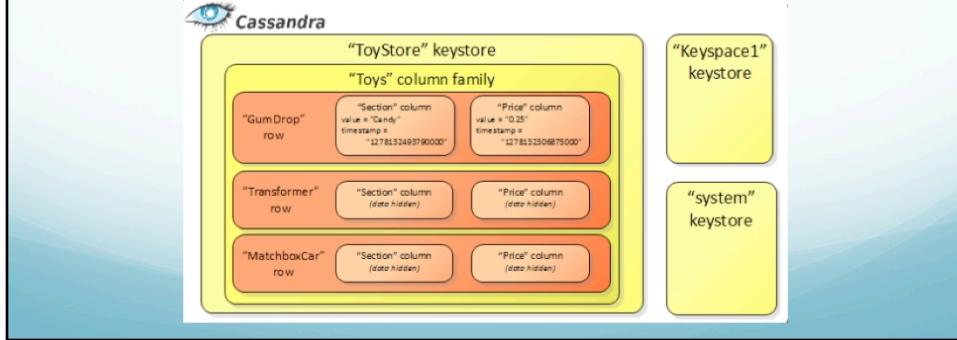
But in a column store several entries will need to be modified to change a record (multiple I/O)

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

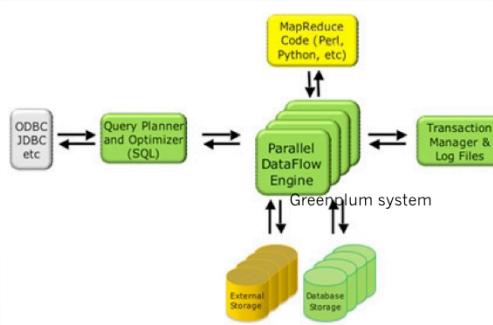
- Column: a tuple that is a triple containing a name, a value and a timestamp
 - The name and value are both binary (byte[]) and can be of any length.
 - SuperColumn is a tuple with a binary name and value which is a map containing an unbounded number of columns that are keyed by the Columns name
 -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



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.

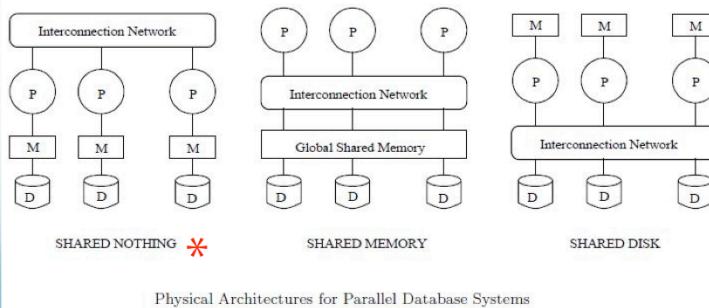
You have to create applications for these distributed systems.

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.

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



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

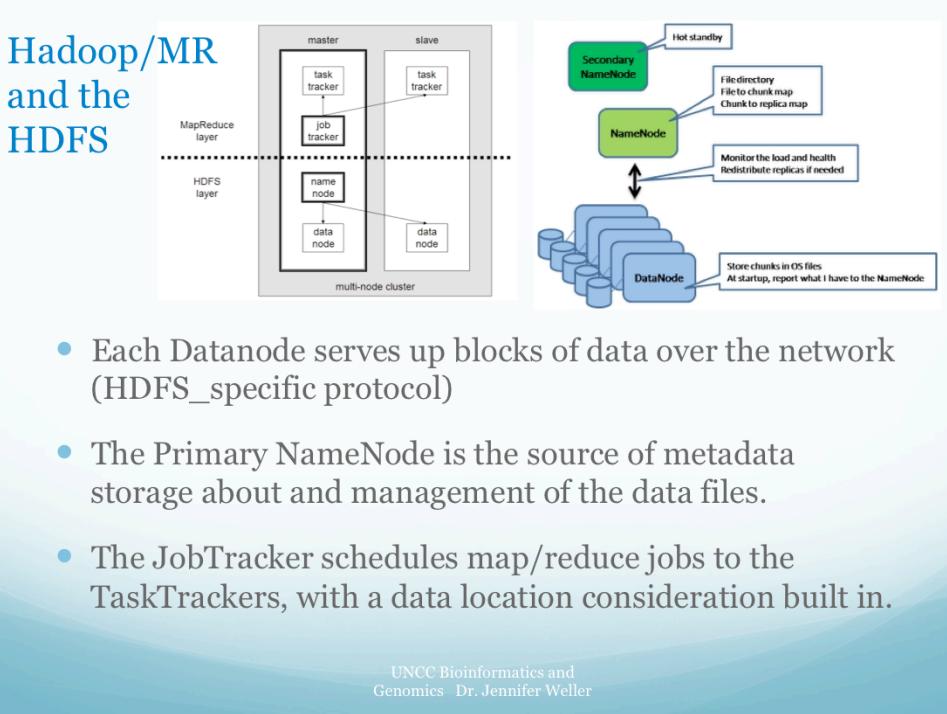


The goal is to support data-intensive distributed applications.
Thousands of independent calculations
Peta-bytes of data.
It is written in Java.

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/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.

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.

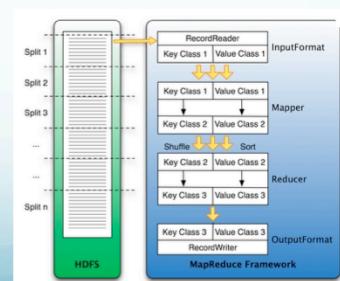
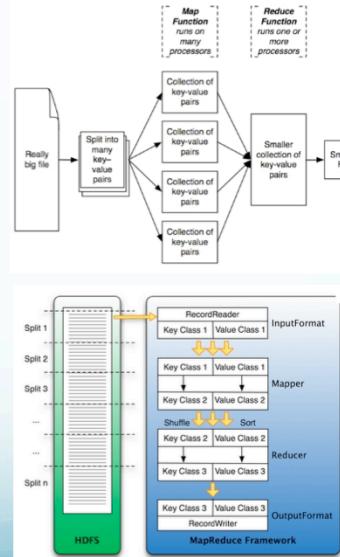
NameNode is the point of management for metadata storage/management. HDFS Federation allows a larger number of small files to be handled.

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) or a user interface web application.

- 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.

MapReduce

MapReduce Framework Schematic



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.

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
 - $\text{Map}(k_1, v_1) \rightarrow \text{list}(k_2, v_2)$
 - $\text{Reduce}(k_2, \text{list}(v_2)) \rightarrow \text{list}(v_3)$

```
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.

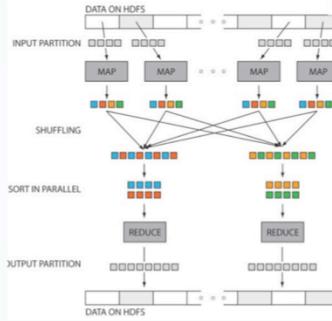
The Map function is applied in parallel to every pair in the ouput data set, resulting in a list of pairs for each call. All pairs with the same key from all lists are grouped together so each different key results in a group. A list of key, value pairs becomes a list of values.

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 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_pipeline/

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 given the key and the number of reducers, returns the index for the requested reduce family. The data is exchanged between nodes, so 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 function writes the results to stable storage in the HDFS.

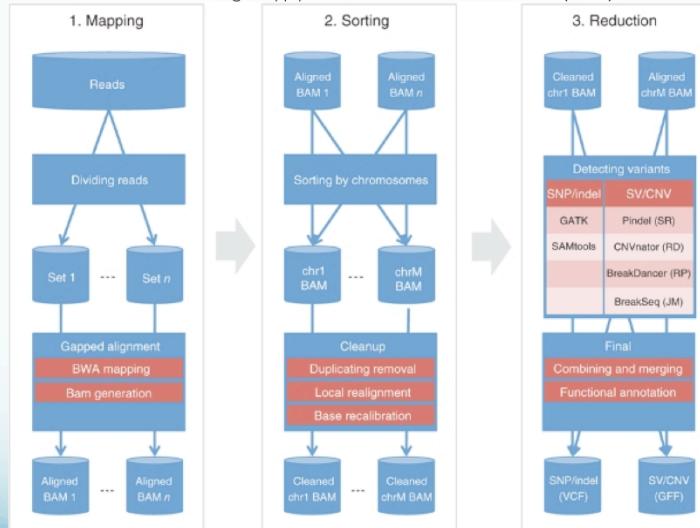
Hbase

- Hbase parses the results from Hadoop/MR.
 - There is a key per element, and the elements are highly normalized.
 - Why is this not relational? → No reasoning sublanguage exists (i.e. SQL) = Codd's Rule #5.
 - This is on purpose – limits the attributes that must be sorted in order to perform an analysis.
- One wants to access data in an *ad hoc*, by subset, manner rather than requiring a full scan.
 - Column databases store data in columns contiguously
 - Often only a few attributes in the row are of interest computationally.
 - I/O is reduced
 - Being very similar, they are amenable to compression

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

MapReduce Bioinformatics Example

Lam et al. HugeSeq pipeline Nature Biotech 30:226-229 (2012)



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 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/>

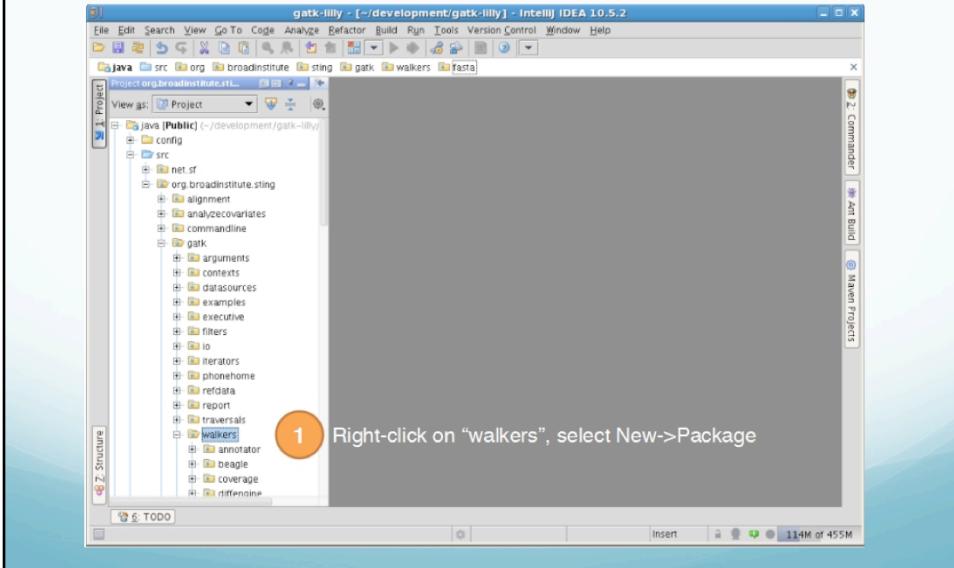
Read Walker allows a computation to be performed on each read in turn (looping to next).

ReadWalker: process one read at a time

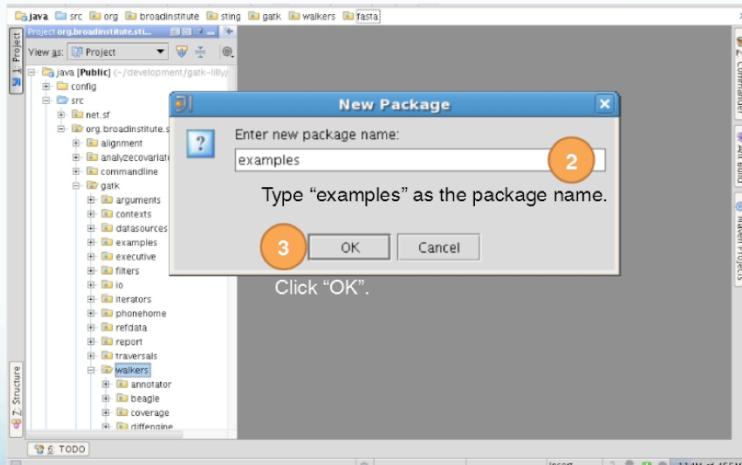


Set a metadata tag, search for specific reads to be excluded, identify indels
Existing algorithms: CycleQualityWalker, CountReadsWalker

Getting started with GATK in the application development environment

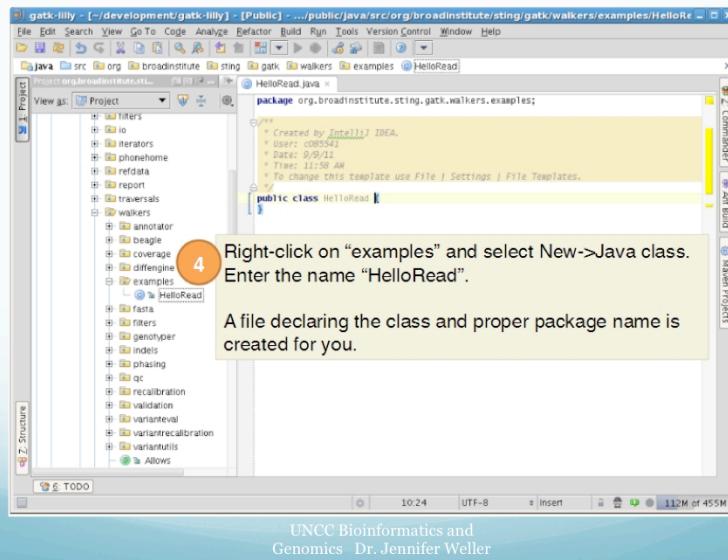


GATK is pre-loaded with a lot of tutorials and example data sets



UNCC Bioinformatics and
Genomics Dr. Jennifer Weller

Getting to the examples



The screenshot shows the IntelliJ IDEA interface with a project titled "gatk-lilly". The "Project" tool window on the left displays a tree structure of Java packages and classes. A right-click context menu is open over the "examples" package in the "org.broadinstitute.sting.gatk.walkers.examples" package. A callout bubble with the number "4" indicates the step: "Right-click on 'examples' and select New->Java class. Enter the name 'HelloRead'." The "HelloRead.java" code editor on the right shows the generated Java code:

```
package org.broadinstitute.sting.gatk.walkers.examples;  
  
/*  
 * Created by IntelliJ IDEA.  
 * User: c085941  
 * Date: 2/9/11  
 * Time: 11:25 AM  
 * To change this template, use File | Settings | File Templates.  
 */  
public class HelloRead {
```

A tooltip message states: "A file declaring the class and proper package name is created for you."

UNCC Bioinformatics and
Genomics Dr. Jennifer Weller

H/MR/Hbase are widely deployed – by whom?
Anyone with extremely large data sets and/or
widely distributed datasets.

- If local copies do not exist you must *map* them out so you can find them.
- Many Bioinformatics operations are ‘embarrassingly parallel’
 - Comparisons to a common reference, not all-to-all
- Once data is assigned to a bunch of parallel processors, each individual task has been *reduced*.
 - Hadoop enables distributed data processing
 - An open source framework for writing and running distributed applications

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.

Bioinformatics data sets are very large and highly distributed.

- If local copies do not exist you must *map* them out so you can find them.
- Many Bioinformatics operations are ‘embarrassingly parallel’
 - Comparisons to a common reference, not all-to-all
- Once data is assigned to a bunch of parallel processors, each individual task has been *reduced*.
 - Hadoop (map reduce) tools enable distributed data processing
 - An open source framework for writing and running distributed applications
 - 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.

UNCC Bioinformatics and
Genomics Dr. Jennifer Weller

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.

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

NoSQL

- “History Repeats Itself:
Sensible and NonsenSQL Aspects of the NoSQL
Hoopla “
 - 2013 by C. Mohan

Not-only SQL = no-SQL databases

- Many datasets are not structured
 - Creating the structures is very time-consuming.
 - If they are structured but not in the way you need them
 - Breaking up structured data gets complicated
 - Uploading to new structures causes I/O challenges.
 - Some data types don't fit the relational structure very well
 - Text, images, XML (hierarchical models in general)
- Hadoop uses name/value pairs but these don't have formal mathematical properties, this is a key-value (2-attribute) approach.
- MapReduce retrieves data subsets. It is not declarative (SQL is).
 - SQL states (declares) WHAT you want, and the program figures out how to give it to you.
 - MapReduce requires you to specify the steps needed to retrieve the results.
 - Designed for off-line processing not real-time (transactional) processes

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

No-SQL concepts

- Data model
 - Cluster: data is distributed across several machines (nodes)
 - Keyspace: a cluster has one or more keyspaces, which is a group of column families with a logical connection.
 - A given application tends to have a dedicated keyspace
 - Column family: a set of columns
 - An analogy is a hash map, keys are strings and values are column rows
 - Columns are sorted in a pre-defined way
 - The family is made up of EITHER standard columns OR super columns
 - Column: The unit of information in the model. A triplet of name, value and timestamp.
 - Sorting is always done on the name

The keyspace is similar to indexing on particular columns – again a 3-part structure is often used, with a name, value and timestamp. So you can convert data rows to a family of column sets.