CS-E4640 - Big Data Platforms
Lecture 4

Keijo Heljanko

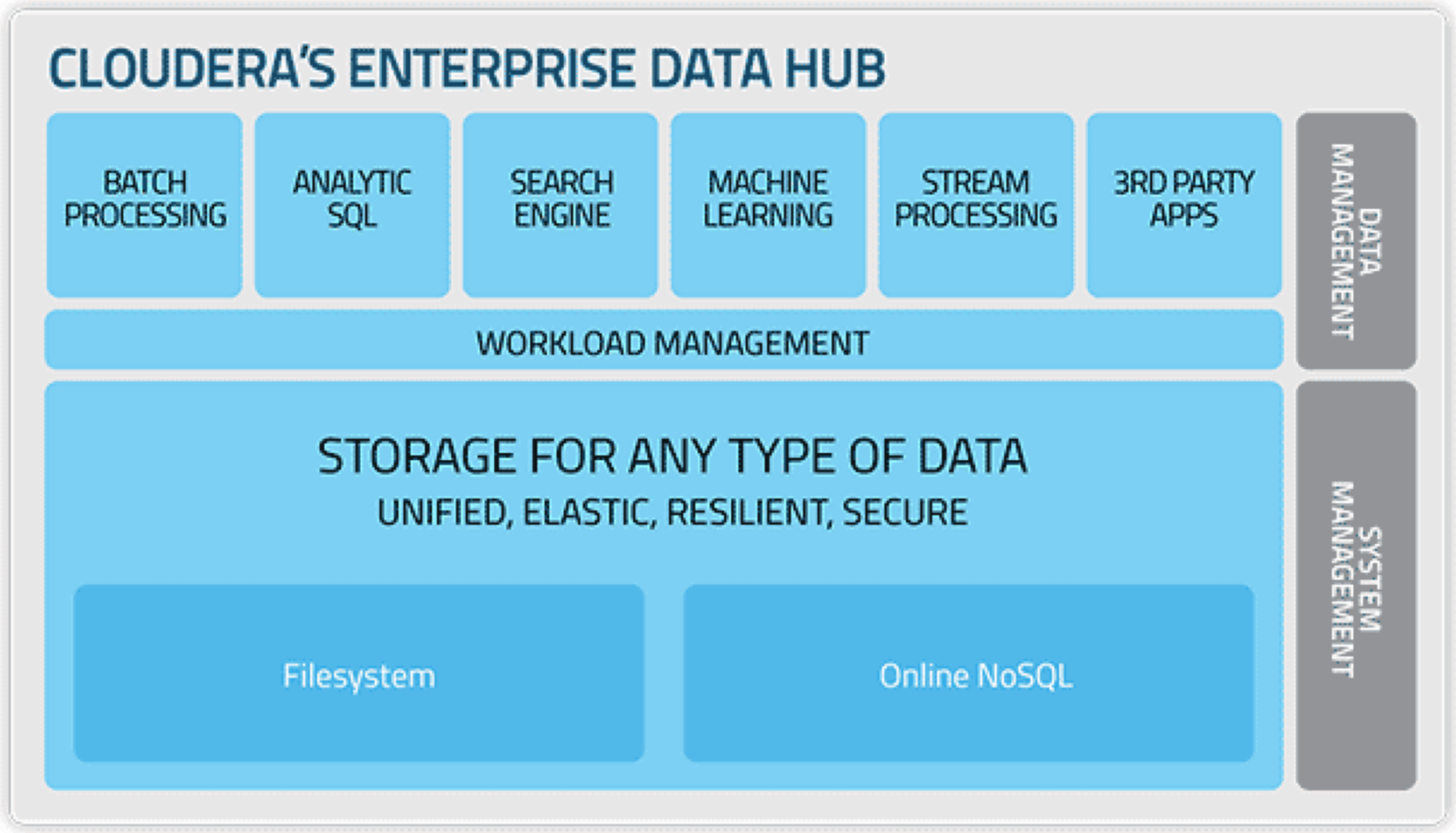
Department of Computer Science
University of Helsinki & Aalto University
keijo.heljanko@helsinki.fi

3.10-2018

Hadoop - Linux of Big Data

- Hadoop = Open Source Distributed Operating System Distribution for Big Data
  - Based on Google architecture design
  - Fault tolerant distributed filesystem: HDFS
  - Batch processing systems: Hadoop MapReduce and Apache Pig (HDD), Apache Spark (RAM)
  - Parallel SQL implementations for analytics: Apache Hive, Cloudera Impala, Spark SQL, Facebook Presto
  - Fault tolerant distributed database: HBase
  - Distributed machine learning libraries, text indexing & search, etc.
  - Hadoop Distributed Filesystem HDFS is often used as a filesystem for Spark, and MapReduce programs are also run on the same cluster as Spark jobs. Therefore understanding Hadoop commands is useful

Example: Cloudera's Hadoop Stack

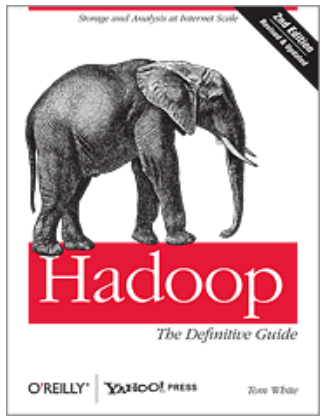


“SQL on Hadoop is the New Black”

- Many new analytics SQL implementations on top of Hadoop
  - Apache Hive
  - Cloudera Impala
  - Spark SQL
  - Presto from Facebook

Hadoop Books

- I can warmly recommend the Book:
  - Tom White: “Hadoop: The Definitive Guide, Fourth Edition”, O’Reilly Media, 2015. ISBN: 9781491901687.
    `http://www.hadoopbook.com/`



- An alternative book is:
  - Chuck Lam: “Hadoop in Action”, Manning, 2010. Print ISBN: 9781935182191.

Example Hadoop Commands

- `start-all.sh`
  - Starts all Hadoop daemons.

Example Hadoop Commands

- `hadoop namenode -format`
  - Formats the HDFS filesystem
- `hadoop fs -mkdir foo`
  - Creates a directory called "`foo`" in the HDFS home directory of the user
- `hadoop fs -rmr foo`
  - Removes the directory/file called "`foo`" in the HDFS home directory of the user
- `hadoop fs -copyFromLocal foo.txt bar`
  - Copies the file "`foo.txt`" to the HDFS directory/file called "`bar`" in the HDFS home directory of the user

Recap of Hadoop Commands Used (cnt.)

- `hadoop fs -ls foo`
  - Does a directory/file listing for directory called "`foo`" in the HDFS home directory of the user
- `hadoop jar wc.jar WordCount foo bar`
  - Runs MapReduce job in file "`wc.jar`" in the q class "`WordCount`" using the HDFS directory "`foo`" as input and stores the output in a new HDFS directory "`bar`" to be created by the job (i.e., "`bar`" should not yet exist in HDFS)
- See the following page on information on how to run Spark jobs on YARN:
  `https://spark.apache.org/docs/latest/running-on-yarn.html`

Recap of Hadoop Commands Used (cnt.)

- `hadoop fs -copyToLocal foo bar`
  - Copies the directory/file "`foo`" in the user HDFS home directory to the local file system directory/file "`bar`"
- `stop-all.sh`
  - Stops all Hadoop daemons

Hadoop Default ports

- The URL: `http://localhost:50070` can be used to access the NameNode through a Web browser. Hint: You can also browse HDFS filesystem through this URL.
- The URL: `http://localhost:8088` can be used to access the YARN resource manager to see the status of your MapReduce jobs, hit refresh to refresh the view

Hadoop NameNode and SecondaryNameNode Locking

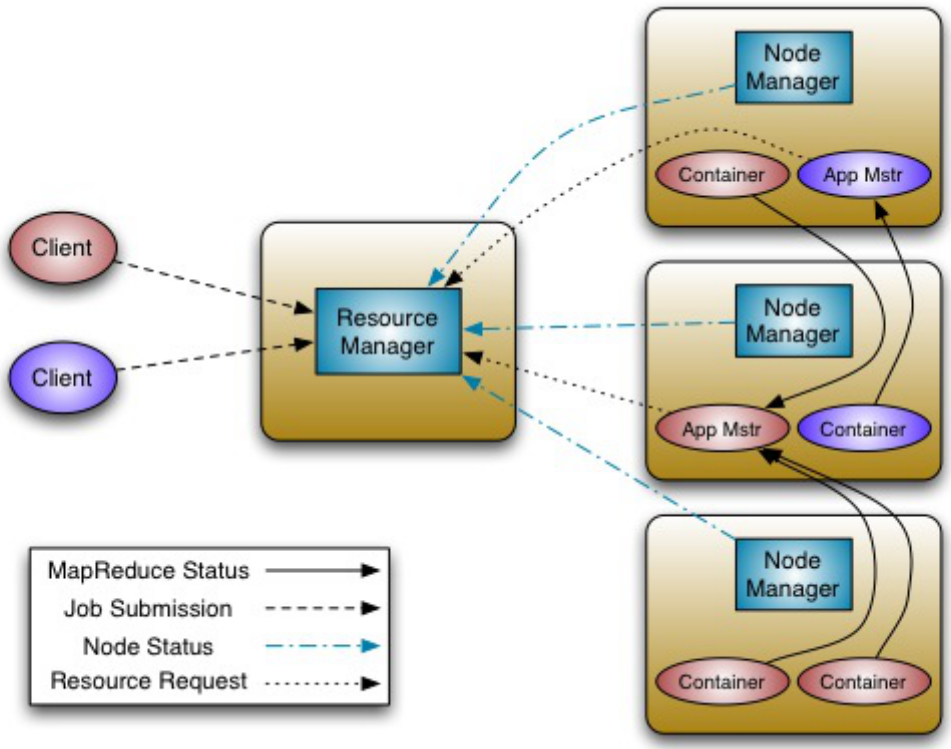
- Hadoop Namenode and SecondaryNameNode use locking to disallow the accidental starting of two daemons modifying the same files and leading to HDFS inconsistency
- If either daemon gets killed uncleanly, it they might leave lock files around and prevent a new NameNode or SecondaryNameNode from starting
- The lock files are called `data/in_use.lock`, `name/in_use.lock`, and `namesecondary/in_use.lock`.
- They are located in `dfs` subdirectory of `hadoop.tmp.dir` that can be set in `core-site.xml` (by default `/tmp/hadoop-${user.name}`)

Commercial Hadoop Support

- Cloudera: A Hadoop distribution based on Apache Hadoop + patches. Available from:
  `http://www.cloudera.com/`
- Hortonworks: Yahoo! spin-off of their Hadoop development team, will be working on mainline Apache Hadoop.
  `http://www.hortonworks.com/`
- MapR: A rewrite of much of Apache Hadoop in C++, including a new filesystem. API-compatible with Apache Hadoop.
  `http://www.mapr.com/`

YARN- Hadoop NextGen

- Jobtracker of MapReduce 1.0 is replaced by a general purpose cluster resource manager in YARN (Hadoop Nextgen 2.0+) to allow MapReduce to co-exists more nicely with other programming frameworks, Apache Spark in particular, on the same cluster



Fault Tolerance Strategies for Storage

- Hard disks are a common cause of failures in large scale systems with field reports giving around 3% average yearly replacement rates
- There are many ways to make more reliable storage out of unreliable hard disks. Here we list some examples only, from smaller scale to larger scale:
  - RAID - Redundant array of independent (originally: inexpensive) disks. Commonly used levels for improving reliability: RAID 1, RAID 5, RAID 6, RAID 10. Used inside servers & specialized SAN/NAS hardware
  - Replication and/or erasure coding of data inside datacenter: Example: Three way default replication in HDFS, with data residing on at least two racks
  - Geographical replication across datacenters: Example: Amazon S3 storage service offers geographic replication

RAID - Redundant Array of Independent Disks

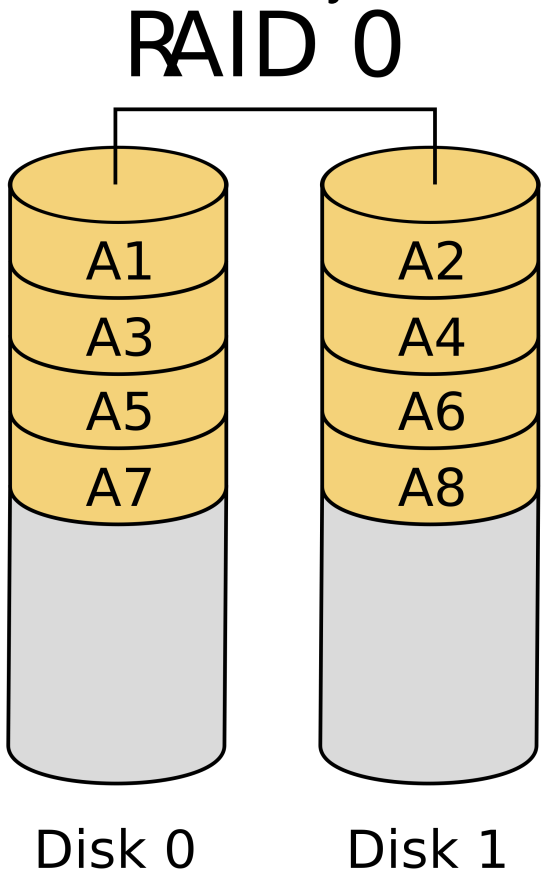
- Most commonly used fault tolerance mechanism in small scale
- RAID 0 : Striping data over many disks - No fault tolerance, improves performance
- Commonly used levels for fault tolerance:
  - RAID 1 : Mirroring - All data is stored on two hard disks
  - RAID 5 : Block-level striping with distributed parity
  - RAID 6 : Block-level striping with double distributed parity
  - RAID 10 : A nested RAID level: A stripe of mirrored disk pairs

RAID - Redundant Array of Independent Disks (cnt.)

- For additional info, see: `http://en.wikipedia.org/wiki/Standard_RAID_levels`, from where the RAID images in the following pages are obtained
- Terminology used: IOPS - I/O operations per second - the number of small random reads/writes per second

RAID 0 - Striping

- Stripes data over a large number of disks to improve sequential+random reads&writes
- Very bad choice for fault tolerance, only for scratch data that can be regenerated easily

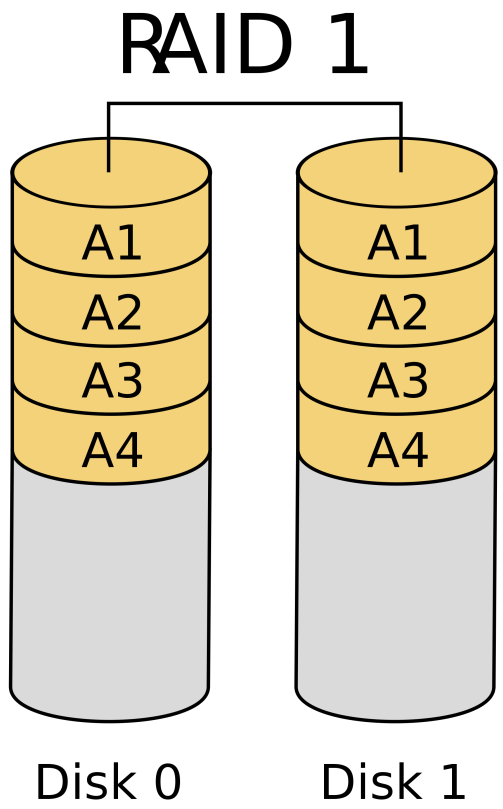


RAID 1 - Mirroring

- Each data block is written to two hard disks: first one is the master copy, and secondly a mirror slave copy is written after the master
- Reads are served by either one of the two hard disks
- Loses half the storage space and halves write bandwidth/IOPS compared to using two drives

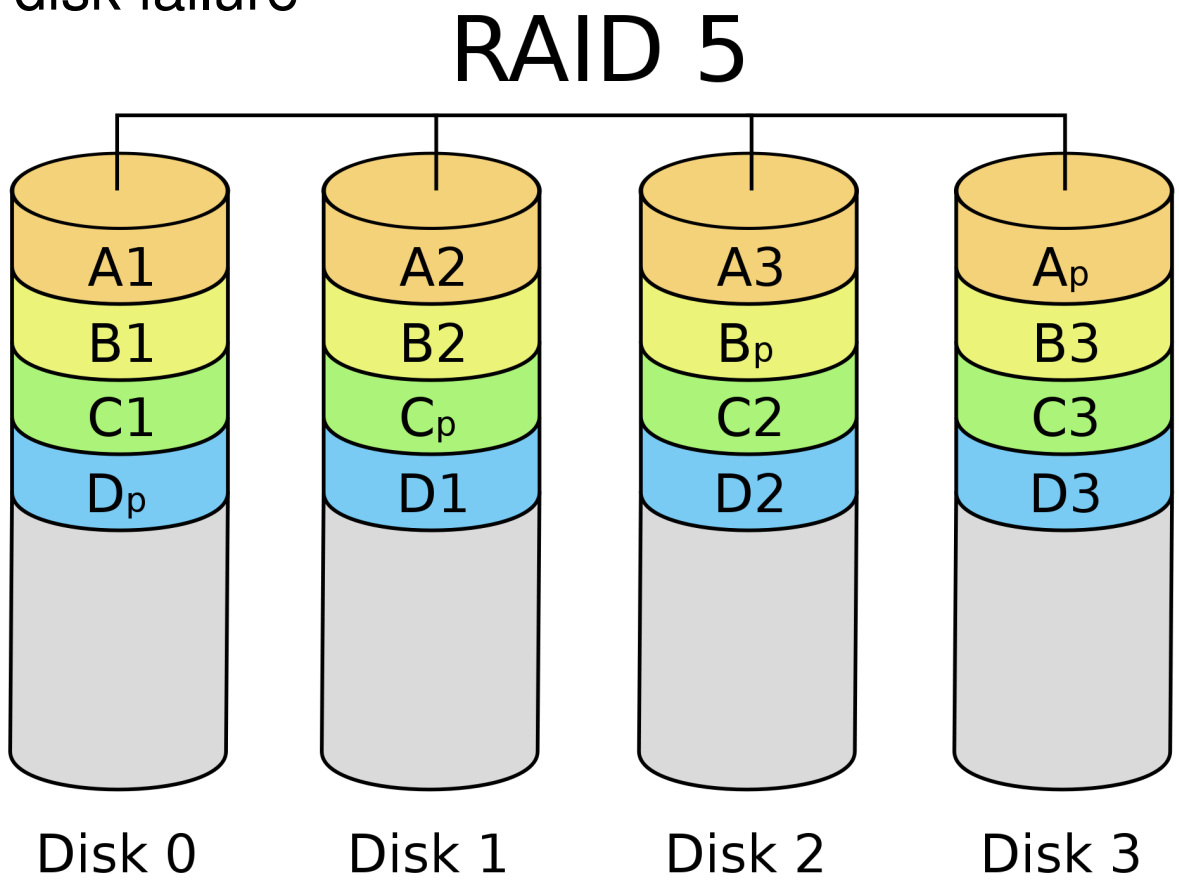
RAID 1 - Mirroring (cnt.)

- Data is available if either hard disk is available
- Easy repair: Replace the failed hard disk and copy all of the data over to the replacement drive



RAID 5 - Block-level striping with distributed parity

- Data is stored on $n + 1$ hard disks, where a parity checksum block is stored for each n blocks of data. The parity blocks are distributed over all disks. Tolerates one hard disk failure



RAID 5 - Properties

- Sequential reads and writes are striped over all disks
- Loses only one hard disk worth of storage to parity
- Sequential read and write speeds are good, random read IOPS are good
- Random small write requires reading one data block + parity block, and writing back modified data block + modified parity block, requiring 4 x IOPS in the worst case (battery backed up caches try to minimize this overhead.)

RAID 5 - Properties (cnt.)

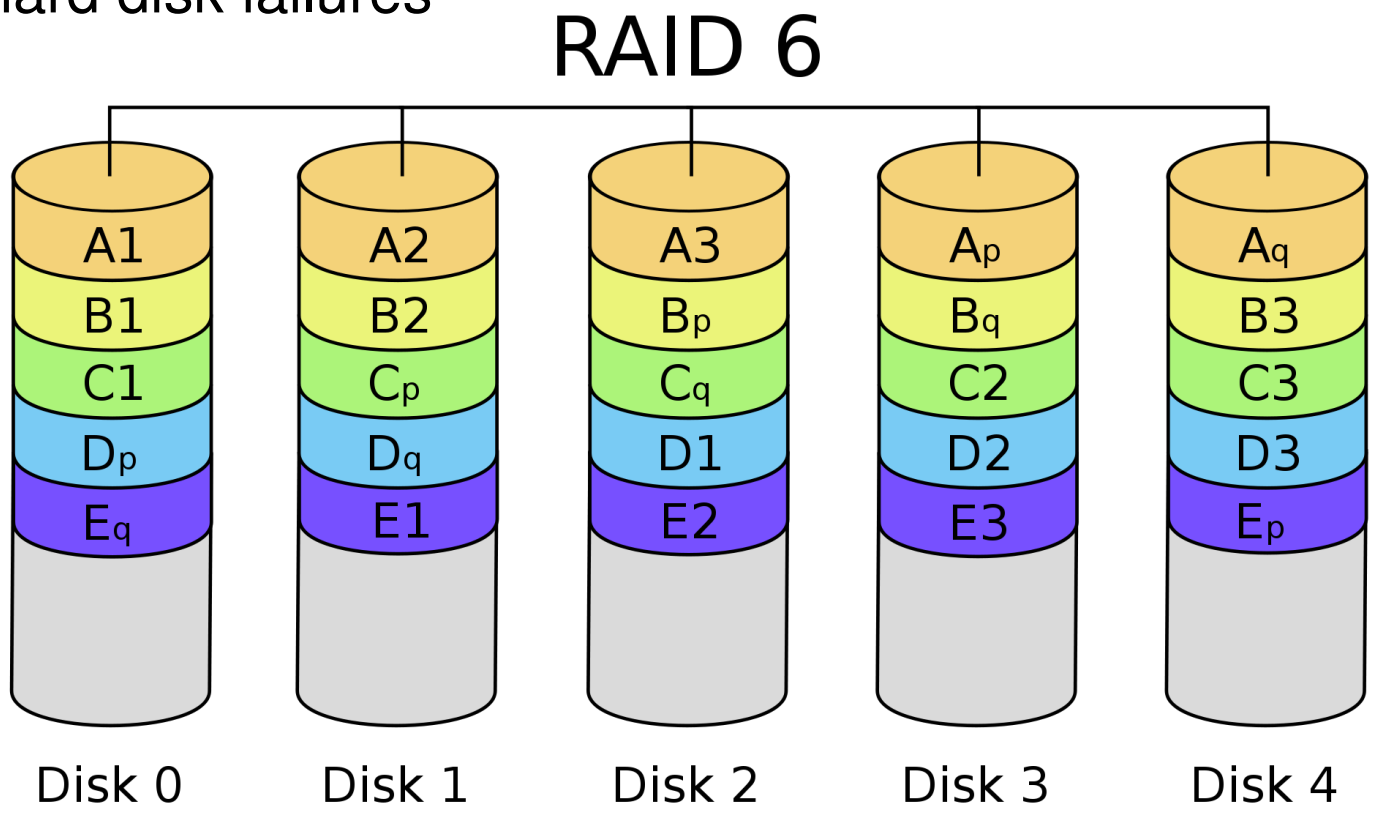
- Rebuilding a disk requires reading all of the contents of the other *n* disks to regenerate the missing disk using parity - this is a slow process
- Slow during rebuild: When one disk has failed, each missing data block read requires reading *n* blocks of data when rebuild is in progress
- Vulnerability to a second hard disk failure during array rebuild and long rebuild times make most vendors instead recommend RAID 6 (see two slides ahead) with large capacity SATA drives

RAID 5 - Properties (cnt.)

- "RAID 5 Write Hole": A power failure during writes to a stripe (remember, n +1 disks need to be updated together in the worst case for large writes) can leave the array in a corrupted state - battery backed up memory caches (and UPS) are used to try to get rid of this problem
- RAID 5 does not protect from a single drive with bad blocks from corrupting the array if a disk gives out bad data during parity calculations. Use of "hot spares" recommended to speed up recovery start

RAID 6 - Block-level striping with double distributed parity

- Data is stored on $n + 2$ hard disks, where two parity checksum blocks are stored for each n blocks of data. The parity blocks are distributed over all disks. Tolerates two hard disk failures



RAID 6 - Properties

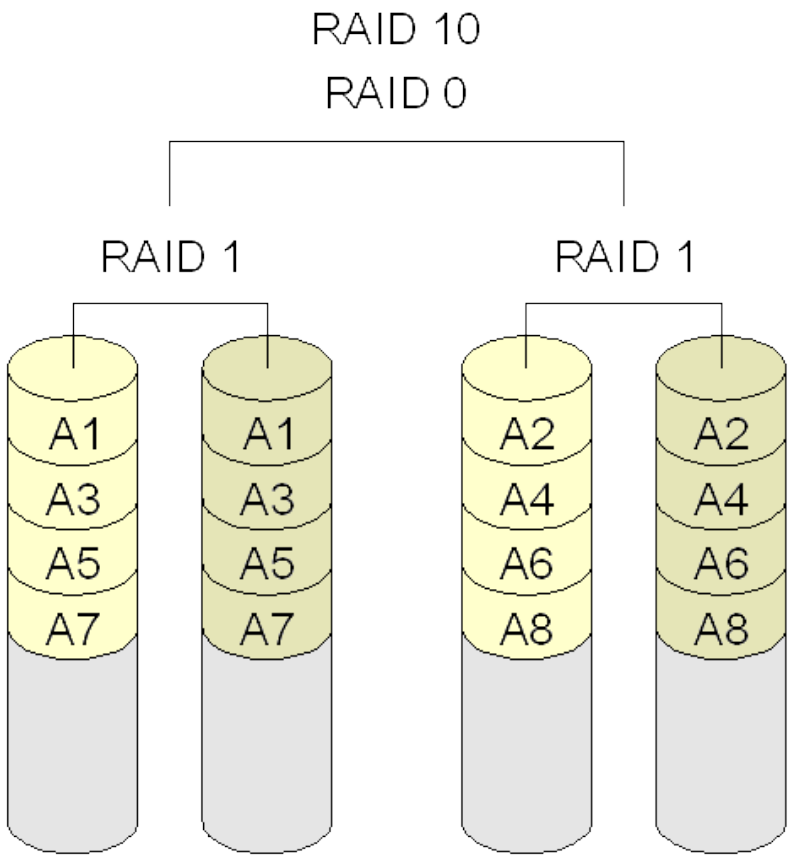
- Sequential reads and writes are striped over all disks
- Loses two hard disk worth of storage to parity
- Sequential read and write speeds are good, random read IOPS are good
- Random small write requires reading one data block + two parity blocks, and writing back modified data block + two modified parity blocks, requiring 6 x IOPS in the worst case (battery backed up caches try to minimize this overhead)
- Slow during rebuild: When 1-2 disks have failed, each missing data block read requires reading *n* blocks of data when rebuild is in progress

RAID 6 - Properties (cnt.)

- Can tolerate one additional disk failure during rebuild, contents of any two missing disks can be rebuilt using the remaining n disks
- "RAID 6 Write Hole": A power failure during writes to a stripe (remember, $n + 2$ disks need to be updated together in the worst case for large writes) can leave the array in a corrupted state - battery backed up memory caches (and UPS) are used to try to get rid of this problem
- In two disk failures: RAID 6 does not protect from a single drive with bad blocks from corrupting the array if a disk gives out bad data during parity calculations

RAID 10 - Stripe of Mirrors

- Data is stored on $2n$ hard disks, each mirror pair consists of two hard disks and the pairs are striped together to a single logical device (see: `http://en.wikipedia.org/wiki/Nested_RAID_levels`).



RAID 10 - Properties

- Tolerates only one hard disk failure in the worst case (n - one disk from each mirror pair - in the best case)
- Loses half of the storage space
- Sequential reads and writes are striped over all disks
- Sequential read and write speeds are good, random read and write IOPS are good

RAID 10 - Properties (cnt.)

- Each data block is written to two hard disks. Random small write require 2 x IOPS in the worst case
- Quicker during rebuild: Missing data is just copied over from the other disk of the mirror. Use of “hot spares” recommended to speed up recovery start
- Can not tolerate the failure of both disks in the same mirror

RAID 10 - Properties (cnt.)

- On unclean shutdown missing changes should be copied from the first master drive to the second slave drive - No “Write hole problem” if such ordering can be guaranteed. (Example: Solaris RAID1 with write policy “serial”.)
- If writes are unordered (for performance reasons), the “write hole problem” is re-introduced.
- RAID 10 does not protect from a single drive with bad blocks from corrupting the array during rebuild

RAID Usage Scenarios

- RAID 0: Temporary space for data that can be discarded
- RAID 1: Small scale installations, server boot disks, etc.
- RAID 5: Some fault tolerance with minimum storage overhead, small random writes can be problematic, RAID 5 Write Hole problem without special hardware (battery backed up cache / UPS)

RAID Usage Scenarios (cnt.)

- RAID 6: Fairly good fault tolerance with reasonable storage overhead, small random writes can be even more problematic than in RAID 5, RAID 6 Write Hole problem without special hardware (battery backed up cache / UPS)
- RAID 10: Less fault tolerant than RAID 6 but more fault tolerant than RAID 5. Loses half of the storage capacity. Good small random write performance makes this often the choice for database workloads. Can avoid the “Write hole problem” under write ordering guarantees.

Remember: RAID is no substitute for backups!

RAID Hardware Issues

- RAID 1 and RAID 10 can be made safe with proper configuration to use without special hardware support
- RAID 5 and RAID 6 are more storage efficient than RAID 10 but require specialized hardware to protect against corruption due to unclean shutdown (battery backed up cache / UPS)

More on RAID Write Hole Problem

- RAID 5 and RAID 6 need to atomically update all the drives in a stripe to remain consistent
- We know a way to do this from the world of Databases: Atomic Transactions
- How are transactions implemented in databases? By using a persistent log of modifications to be performed and replaying any missing modifications in case of a shutdown in the middle of an atomic sequence of modifications
- Thus RAID 5 and RAID 6 need a persistent transaction log in order to offer atomic transactions for updating a stripe

More on RAID Write Hole Problem (cnt.)

- Battery backed up cache is one way to implement a persistent transaction log
- Another way is to implement a log structured filesystem that is aware of the underlying storage subsystem: Example: Sun (Oracle) ZFS: `http://en.wikipedia.org/wiki/ZFS`
- ZFS implements RAID 5 and RAID 6 equivalent fault tolerance without specialized hardware
- It a filesystem that is given full control over the raw disk devices, it uses a copy-on-write transactional object model to achieve this
- ZFS also has data integrity checksums

Example of Hadoop Use: Hadoop-BAM + SeqPig

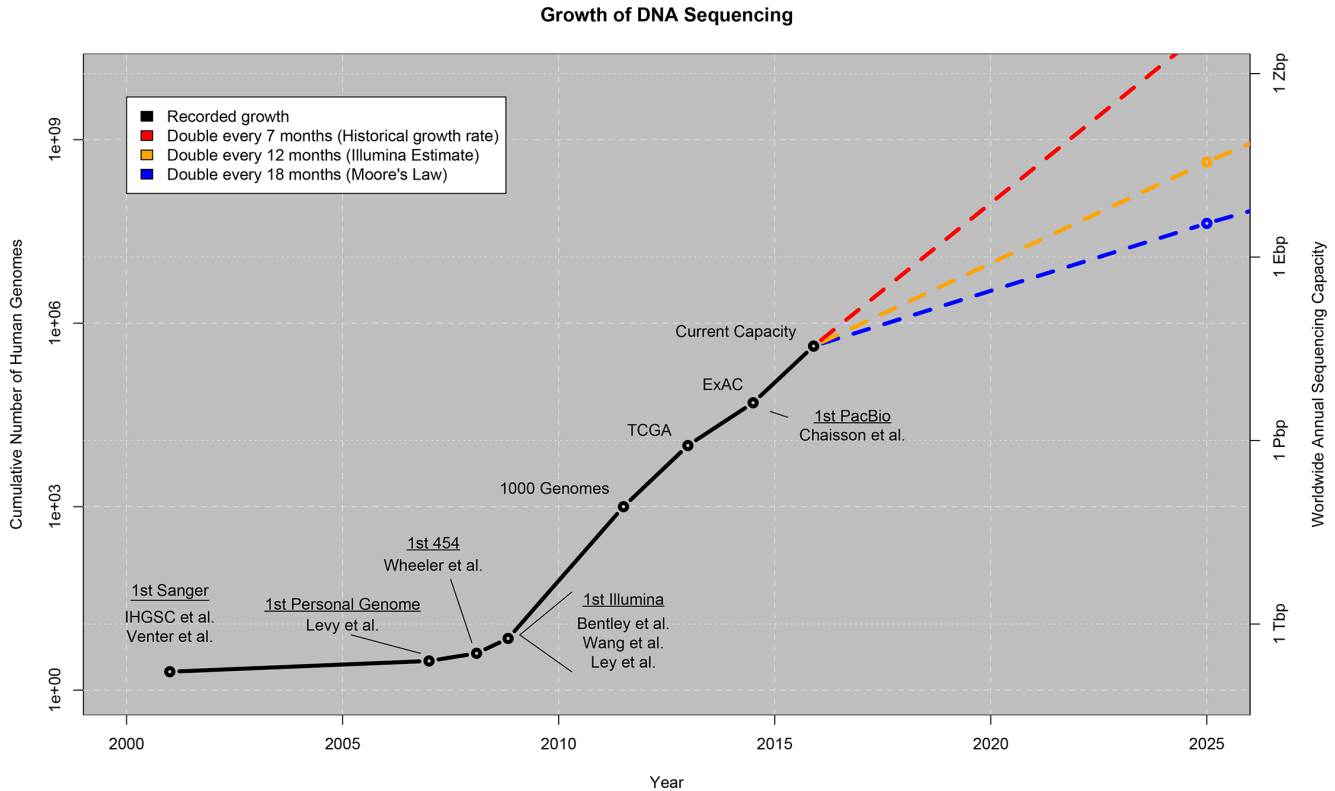
The rest of Lecture 4 will be an example of Hadoop research use and is not be part of the exam requirements.

Genomics Research

- Genomics research has high economic relevance: It is used by biomedicine researchers, hospital diagnostics, food industries, agronomy, pharmaceutical industries, ...
- Example use cases of NGS data analytics include:
  - Human genetics (involving the detection of genotype variations that relate to diseases)
  - Personalized medicine (which will rely critically on the ability to rapidly and reliably map the genetic make-up of individual patients)
  - Genomics selection for agriculture (which will exploit genomics data in farmed livestock to select the next generation of breeding animals and crop plants)
- All these use cases require advanced analytics methods to be developed to cope with the data growth rate

Big Data: Astronomical or Genomical?

- Genomics data is outgrowing Moore's law: Stephens ZD et al. (2015) Big Data: Astronomical or Genomical?. PLOS Biology 13(7): e1002195. `https://doi.org/10.1371/journal.pbio.1002195`



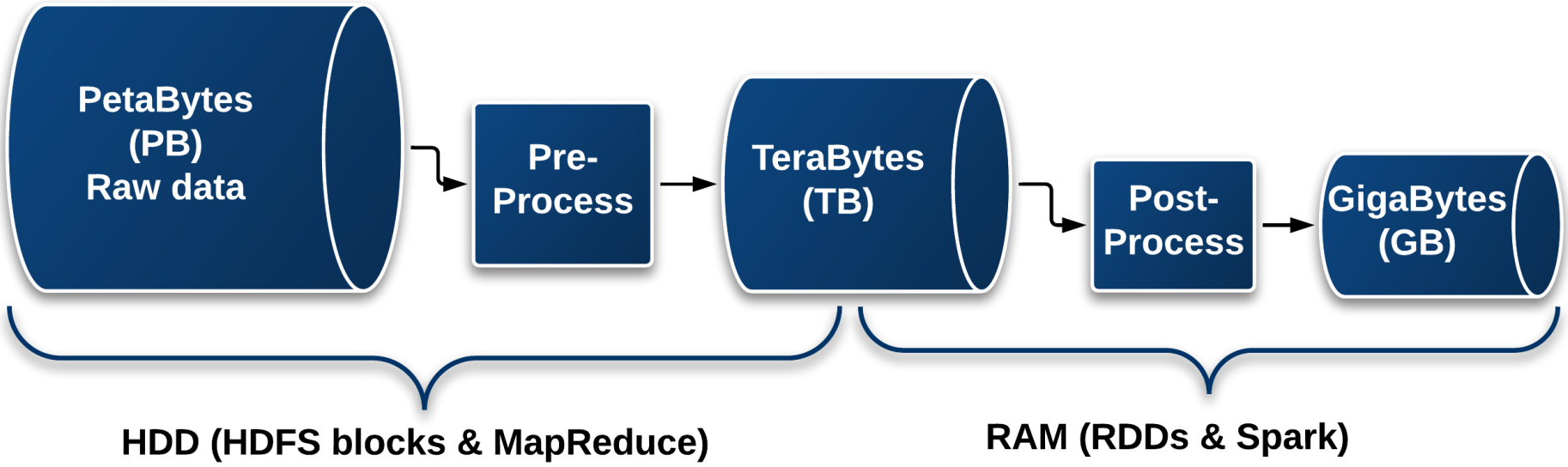
Next Generation Sequencing and Big Data

- The amount of NGS data worldwide is predicted to double every 5 (historical growth) to 12 months (Illumina estimate)
  - This growth is much faster than Moore's law (was) for the growth rate of computing (historically transistor counts have doubled every 18-24 months until now)
- 1000 Genomes project has Petabytes of human genomes data sets
- In many NGS projects multiple large files (100+ Gigabytes) has to be processed
- NGS analytics methods has to cope with the data growth rate $\Rightarrow$ Towards distributed computing methods and parallel algorithms to avoid hitting computational limits

Hadoop-BAM

- A library to interface NGS data formats with Apache Spark and Apache Hadoop
- Includes tools for e.g., sorting of reads, as needed by merging results of parallel read aligners
- Supported fileformats: BAM, SAM, FASTQ, FASTA, QSEQ, BCF, and VCF
- Version 7.9 of the `hadoop-BAM` released: `https://github.com/HadoopGenomics/Hadoop-BAM`
- 3700+ Downloads of the library
- Niemenmaa, M., Kallio, A., Schumacher, A., Klemelä, P., Korpelainen, E., and Heljanko, K.: Hadoop-BAM: Directly Manipulating Next Generation Sequencing Data in the Cloud. Bioinformatics 28(6):876-877, 2012. (`http://dx.doi.org/10.1093/bioinformatics/bts054`).

HDD and RAM based NGS parallelization



- Processing data in main memory instead of files in hard disks $\Rightarrow$ minimal I/O operations. Map/Reduce data from Petabytes to Gigabytes (million times less in the end!)

General parallel pipelines for genomics

- Broad Institute's GATK4 integrates widely-used tools to be run in parallel on clusters using Apache Spark. Relies on Hadoop-BAM for parallel I/O.
  - Current implementation has been focusing mostly on Variant discovery functionalities
  - Alignment and variant files can be processed in parallel, includes e.g. sorting, duplicate marking, realignment and variant calling.
- ADAM from UC Berkeley includes basic tools for file transformations, k-mer counting, allele frequencies on Apache Spark. Uses Hadoop-BAM for I/O.
- Halvade uses Broad Institute's best practices pipeline on Hadoop MapReduce. Hadoop-BAM I/O.

SeqPig

- Parallel scripting language for NGS data sets based on the Apache Pig language
- Compiles into Java, executed by Hadoop MapReduce
- SQL-like functionality with helper functions for NGS data: Filtering data, computing aggregate statistics, doing joins
- Supported fileformats: BAM, SAM, FASTQ, QSEQ, and FASTA
- Schumacher, A., Pireddu, L., Niemenmaa, M., Kallio, A., Korpelainen, E., Zanetti, G., and Heljanko, K.: SeqPig: Simple and scalable scripting for large sequencing data sets in Hadoop. Bioinformatics 30 (1): 119-120, 2014. (dx.doi.org/10.1093/bioinformatics/btt601.)
- See also supplement: http://bioinformatics.oxfordjournals.org/content/suppl/2013/10/17/btt601.DC1/supplement.pdf

Scalability of SeqPig

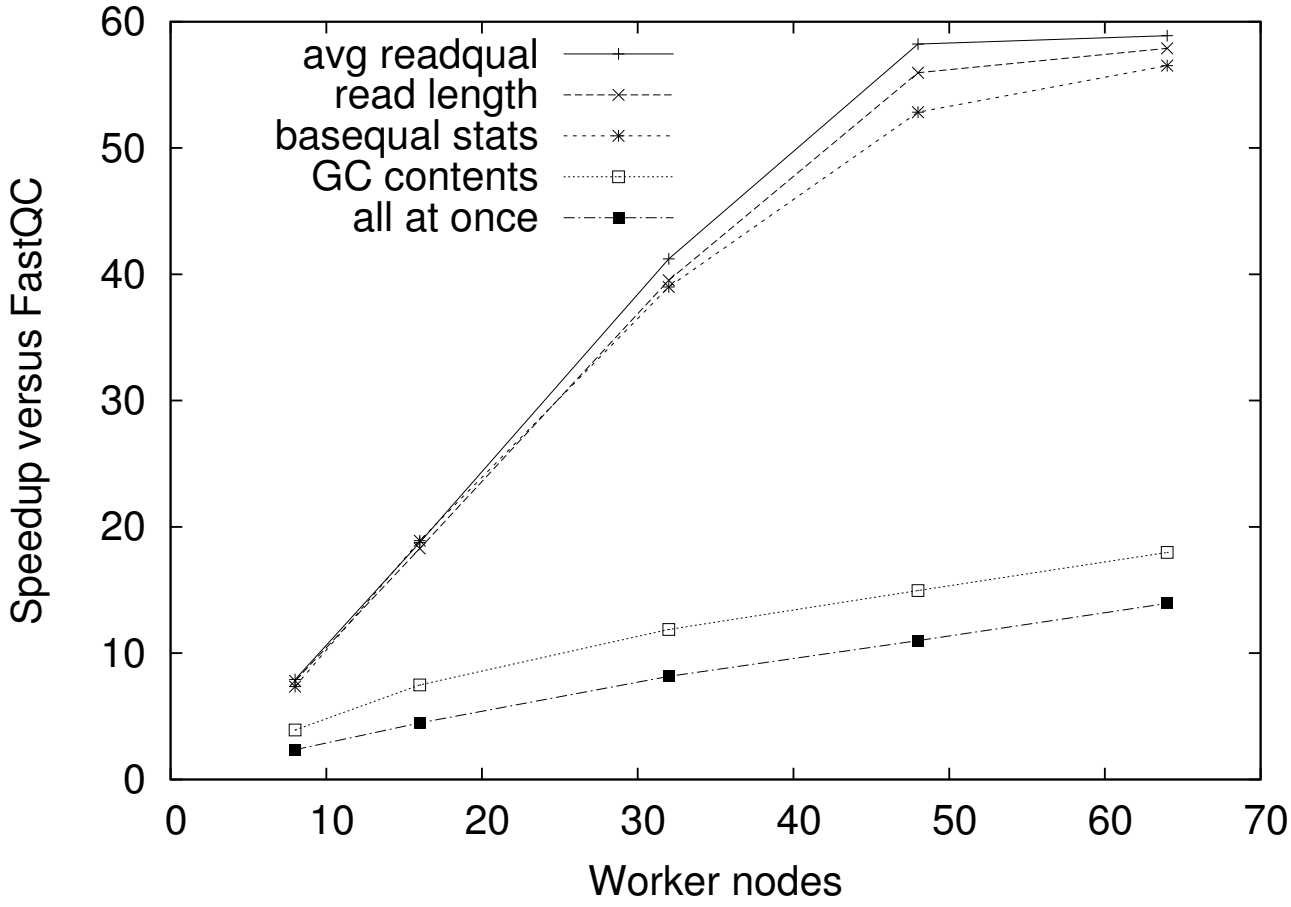


Figure: Scalabilty of SeqPig vs sequential FastQC. Computing statistics on 61.4 GB input file with up to 63 computer Hadoop cluster

SeqPig Benefits and Drawbacks

- Benefits:
  - Automatic parallelization of data processing scripts
  - Easy to learn scripting language with full power of MapReduce
  - Most scripts are at most tens of lines of code vs. hundreds to thousands of lines of Java
  - Also allows calling back user defined functions written in Java/Python
  - Implements SQL like functionality
- Drawbacks:
  - MapReduce has 10+ second startup delay: No for interactive use
  - A specialized language instead of a standard like SQL

MapReduce for Read Alignment

- The SEAL system is a parallelization of the BWA read alignment tool on top of Hadoop (`http://biodoop-seal.sourceforge.net/`)
- It allows multiple BWA instances running on the same physical machine to share the large index of the reference sequence
- For more details, see the paper: "Luca Pireddu, Simone Leo, Gianluigi Zanetti: SEAL: a distributed short read mapping and duplicate removal tool. Bioinformatics 27(15): 2159-2160 (2011)"
- Hadoop-BAM is used to sort the output of SEAL

Current Research Topics

- Focus on plaforms for Big Data Analytics
- Hadoop and Apache Spark scalable batch processing technologies
- Scalable datastores for Genomics and Internet of Things data
- Parallel SQL engines for analytics: Hive, Spark SQL, Cloudera Impala, Facebook Presto
- Adding real-time processing into Hadoop/Spark stack and analyzing its fault tolerance