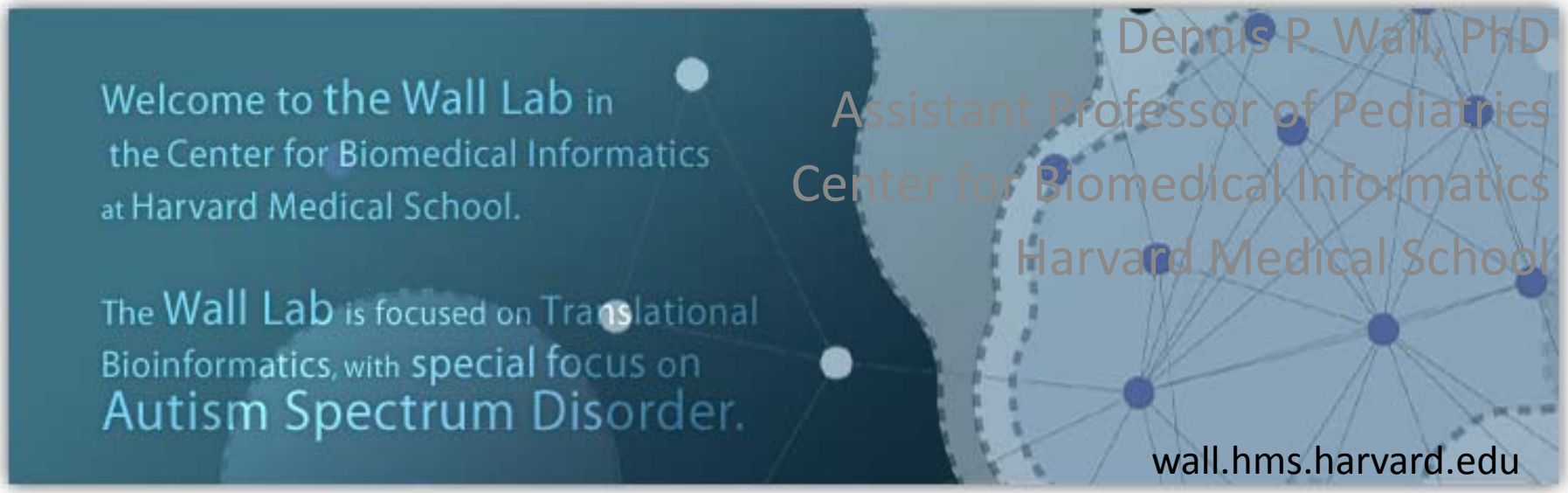
Next-Generation Sequencing Data Analysis and Computation I

- ☑ BMI 714
- ☑ Directors Tonellato and Wal
- ☑ Today's Instructor: Wall



NGS Data Analysis and Computation I

- ☐ Overview
- ☐ Cloud Approaches
- ☐ Quality Control
- ☐ NGS Data Formats
- ☐ Alignment
- ☐ Variant Calling

NGS Data Analysis and Computation I

- ☑ Overview
- ☐ Cloud Approaches
- ☐ Quality Control
- ☐ NGS Data Formats
- ☐ Alignment
- ☐ Variant Calling

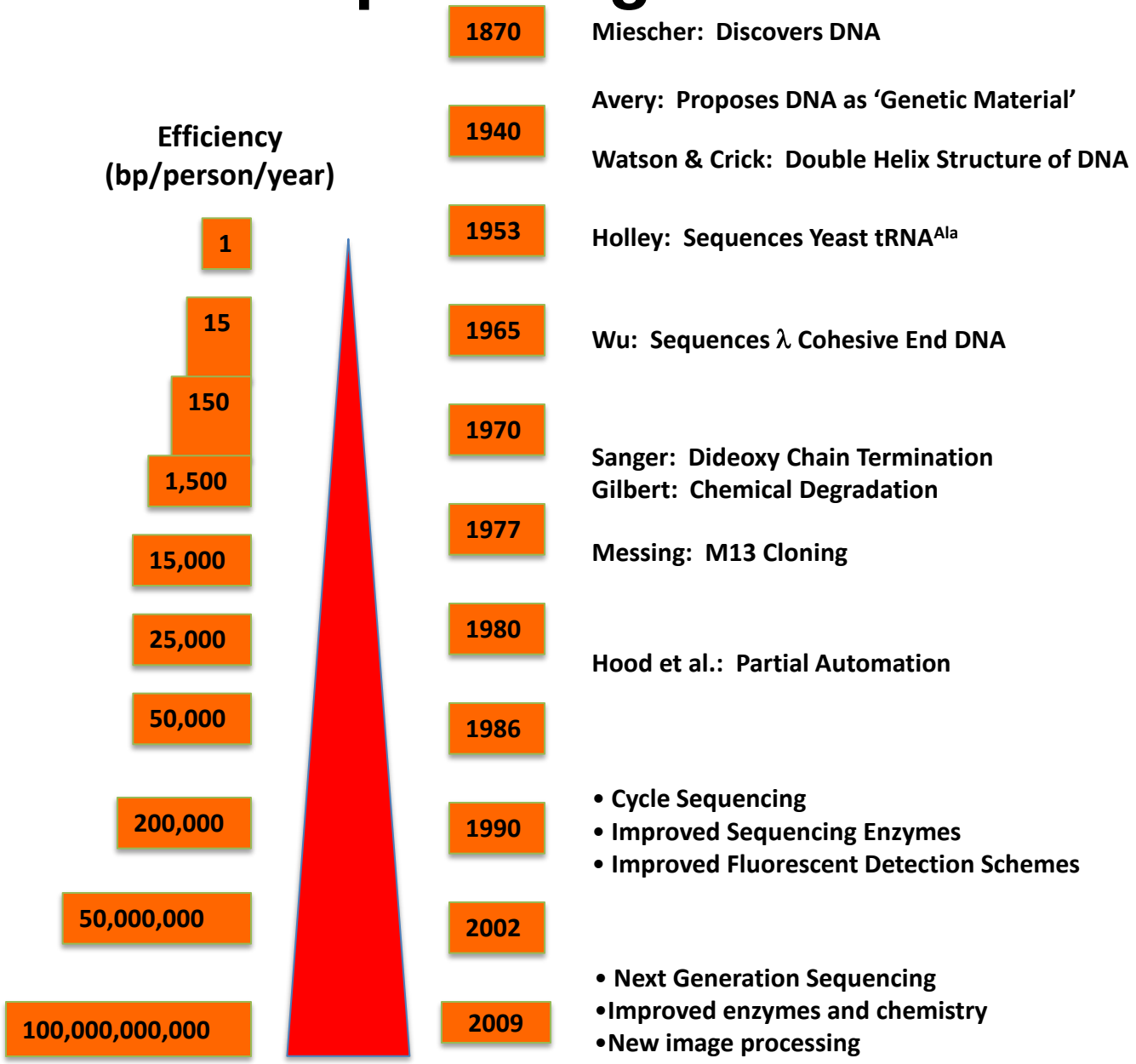
A New Era for Translational Science

- Explosion of experimental data
- Big difficulties in data interpretation
- Driving the need for new paradigms for data computation and knowledge extraction

Challenges in Translational Science

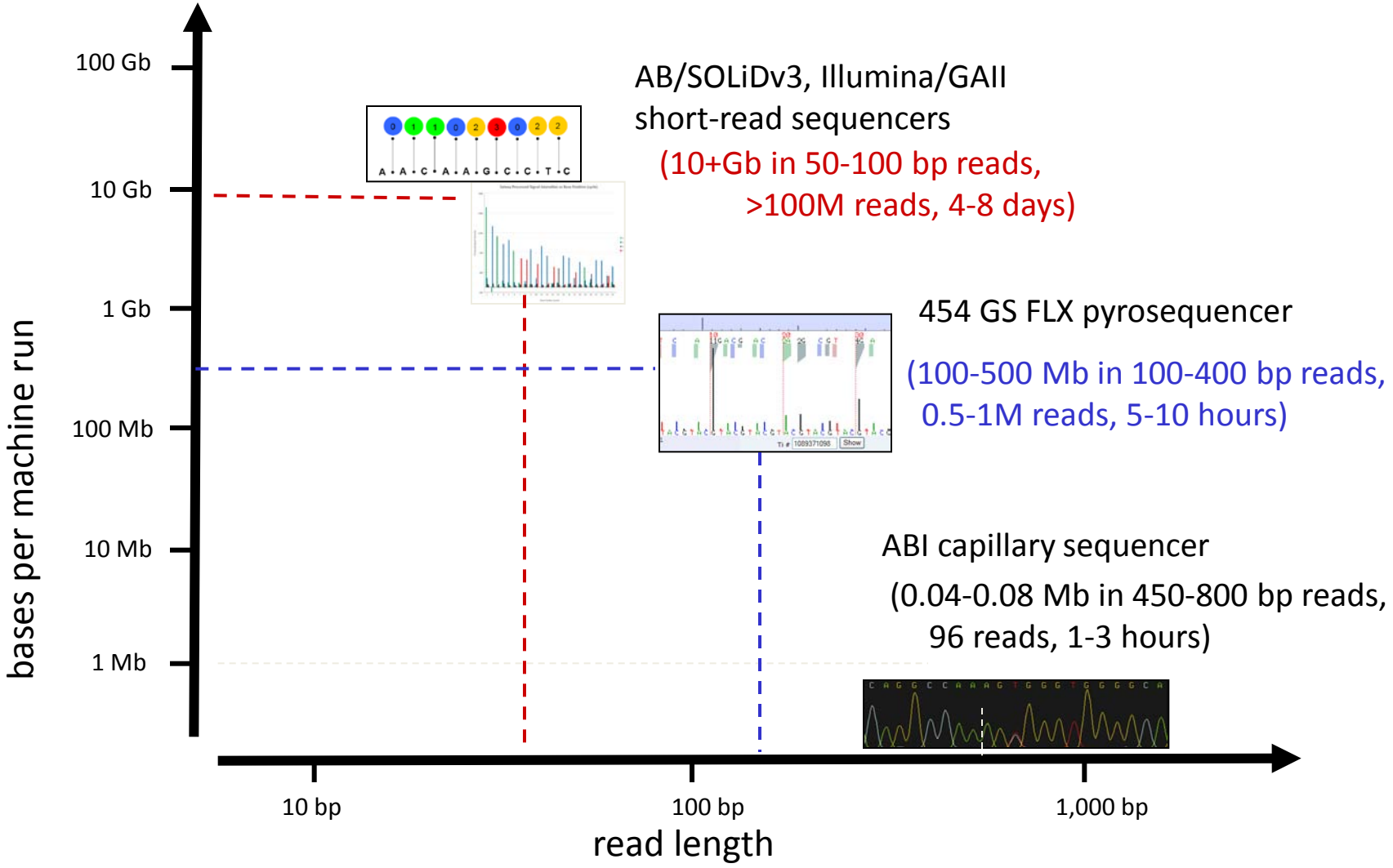
- Mining clinical databases to associate molecular markers to disease, or to reclassify a disease
- Designing information systems to support genome-informed personalized medicine
- Using high-throughput genomic measurements for improved disease profiling, diagnosis/prognosis, and target identification

History of DNA Sequencing

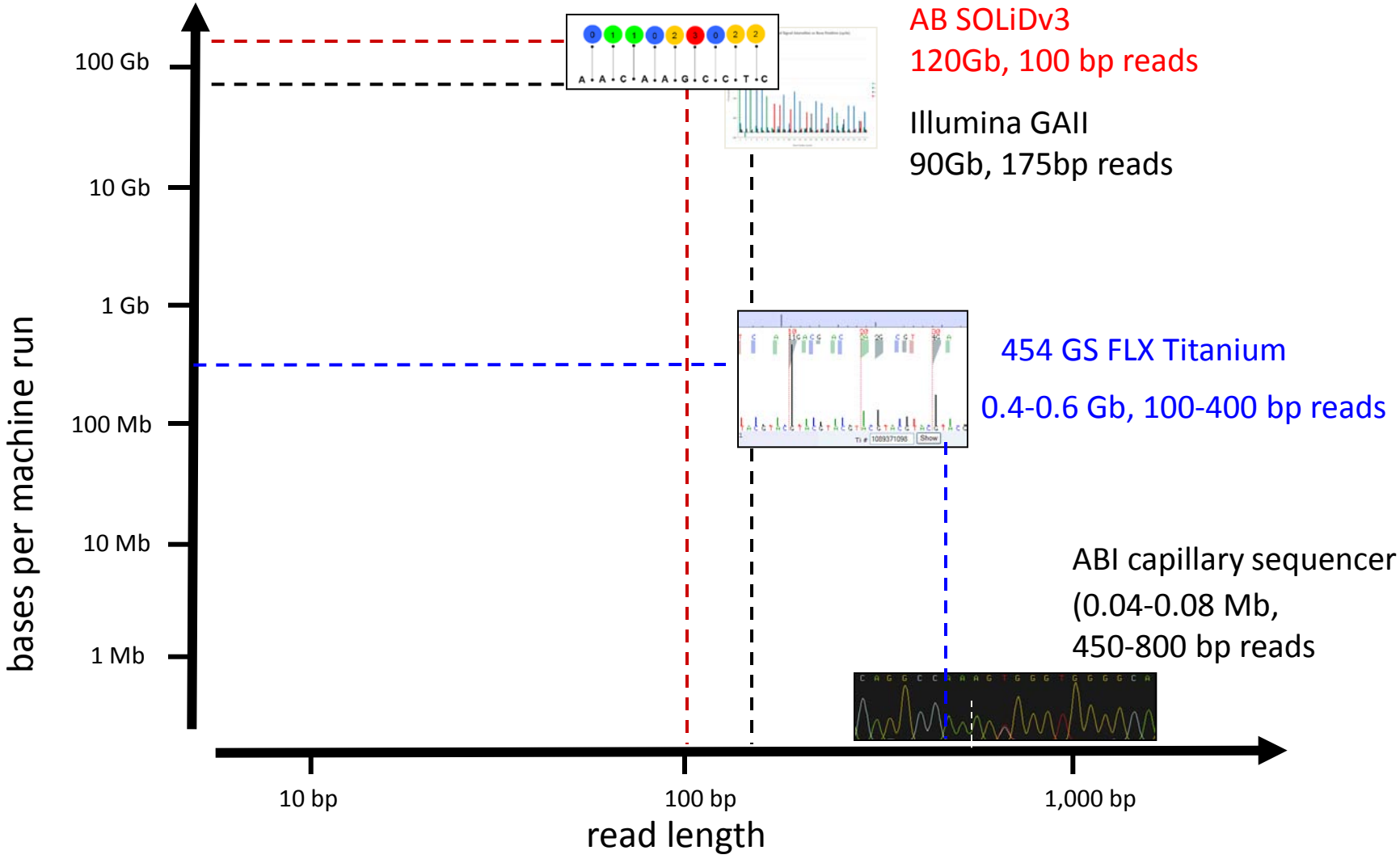


Adapted from Eric Green, NIH; Adapted from Messing & Llaca, *PNAS* (1998)

Next-generation sequencing



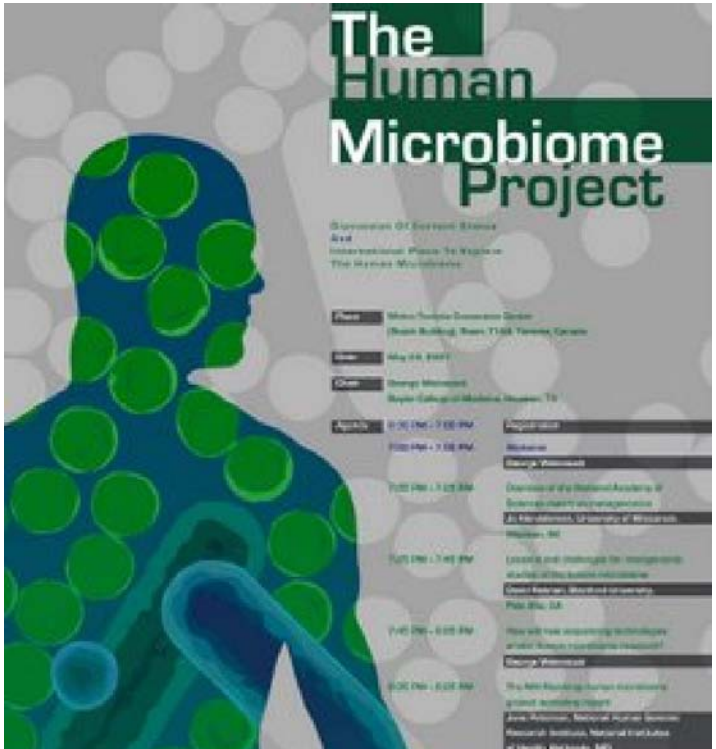
Looking ahead: 2012 and beyond

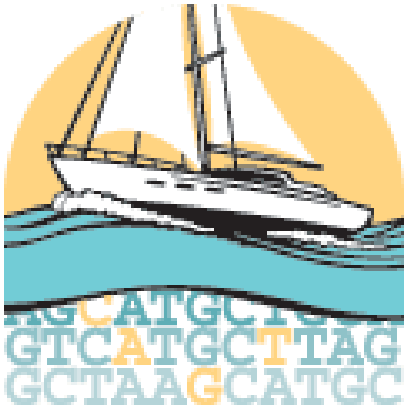


Next Generation Projects



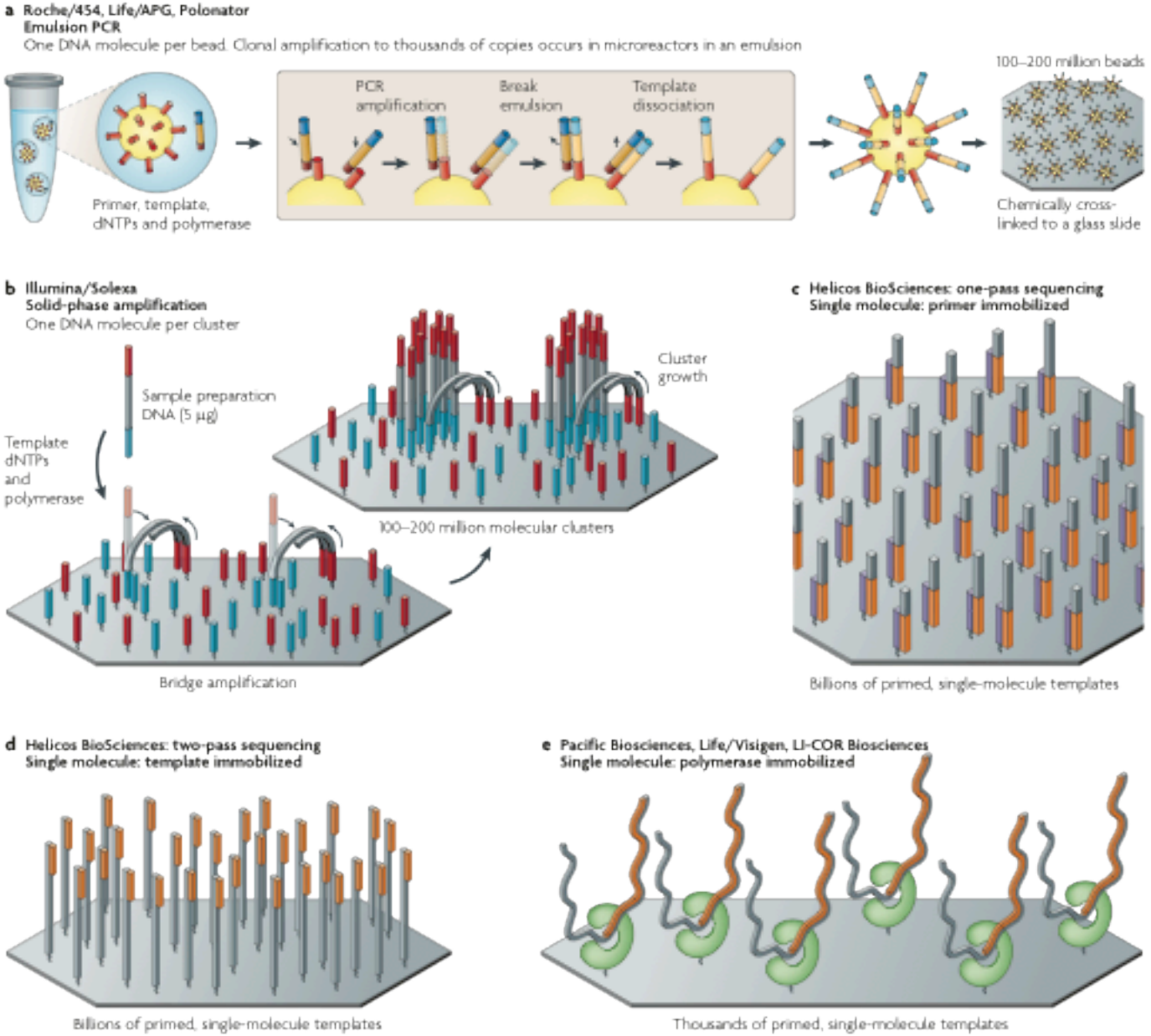






Next Generation Projects

- 1000 Genomes Project
- MalariaGEN - sequencing thousands of malaria isolates
- ICGC – International Cancer Genome Consortium
- Mouse Genomes Project – sequencing 17 common laboratory mouse strains
- 1001 Genomes Project - Arabidopsis whole-genome sequence variation
- UK10K – sequencing of 10,000 healthy and disease affected individuals


a Roche/454, Life/APG, Polonator
Emulsion PCR
One DNA molecule per bead. Clonal amplification to thousands of copies occurs in microreactors in an emulsion
Primer, template, dNTPs and polymerase
PCR amplification
Break emulsion
Template dissociation
100–200 million beads
Chemically cross-linked to a glass slide
b Illumina/Solexa
Solid-phase amplification
One DNA molecule per cluster
Sample preparation DNA (5 μg)
Template dNTPs and polymerase
Cluster growth
100–200 million molecular clusters
Bridge amplification
c Helicos BioSciences: one-pass sequencing
Single molecule: primer immobilized
Billions of primed, single-molecule templates
d Helicos BioSciences: two-pass sequencing
Single molecule: template immobilized
Billions of primed, single-molecule templates
e Pacific Biosciences, Life/Visigen, LI-COR Biosciences
Single molecule: polymerase immobilized
Thousands of primed, single-molecule templates

Sequencing Technologies

Illumina



Capillary (e.g. AB 3730)



Roche 454



AB SOLiD



Helicos



PACIFIC BIOSCIENCES™

Products

- Helicos® Genetic Analysis System
  - HeliScope™ Sequencer
  - HeliScope™ Sequencer Operation
  - HeliScope™ Analysis Engine
  - HeliScope™ Sample Loader
  - HeliScope™ Control Center
- tSMS™ Reagent Kits
- Flow Cells
- Data & Analysis

Helicos® Genetic Analysis System

print page | email page

The Helicos® Genetic Analysis System is the first technology capable of true direct DNA measurement available to the scientific community. The system is comprised of the HeliScope™ Single Molecule Sequencer, the HeliScope™ Analysis Engine, and the HeliScope™ Sample Loader. Novel reagent packaging and intuitive software provide for easy operation, automated tracking of consumables, and overall management of the system.

Download the Sales Specification Sheet (pdf).

The Helicos Genetic Analysis System includes:

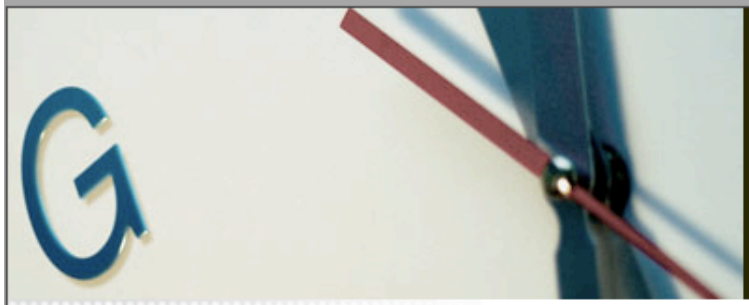


The HeliScope™ Single Molecule Sequencer
A powerful genetic analyzer that performs the tSMS™ chemistry and captures images of single molecules, producing accurate sequences of billions of templates at a time. Images are then sent to the HeliScope Analysis Engine for processing.

Get more information on the HeliScope Single Molecule Sequencer.

Questions?

For information regarding the HeliScope™ Sequencer, or any of our products, call **877-2-HELICOS** or send us a note.



PACIFIC BIOSCIENCES™

Search

Register for More Information

ABOUT US | TECHNOLOGY | SMRT™ BIOLOGY | NEWS & EVENTS | CAREERS | CONTACT US

Single Molecule Real Time

OVERVIEW
- Introduction
- SMRT™ At-a-Glance
- Looking Ahead

SMRT™ TECHNOLOGY
- Observation Window
- Synthesis Of Long DNA
- Real-Time Detection
- High Quality Sequence Data
- Massive Scalability

PUBLICATIONS

Long Reads, Fast Results, and More Informative Data at Lower Overall Costs

Pacific Biosciences was founded in 2004 with the goal of developing Single Molecule Real Time (SMRT™) DNA sequencing technology. This technology enables, for the first time, the observation of natural DNA synthesis by a DNA polymerase as it occurs. Our approach is based on eavesdropping on a single DNA polymerase molecule working in a continuous, processive manner. Distinguished by its long reads, fast time to result, more informative data, and lower overall costs, SMRT DNA sequencing promises to be a transformative technology that will enable a new paradigm in genomic analysis.

PacBio's SMRT technology is built upon three key innovations that overcome major challenges facing the field of DNA sequencing:

The SMRT Cell, which enables observation of individual fluorophores against a dense background of labeled nucleotides by maintaining a high signal-to-noise ratio.
Phospholinked nucleotides, which produce a completely natural DNA strand through fast, accurate, and processive DNA synthesis. A novel detection platform that enables single molecule, real-time detection, as well as flexibility in run configurations and applications.

Basic NGS workflow

DNA Prep

Randomly shear DNA + end repair + size select

Library Prep

Append sequencing adapters

Chip Prep

Layout of library on sequencing slide or wells

Sequencing

For each library fragment – determine the order and identity of bases at either end of the fragment



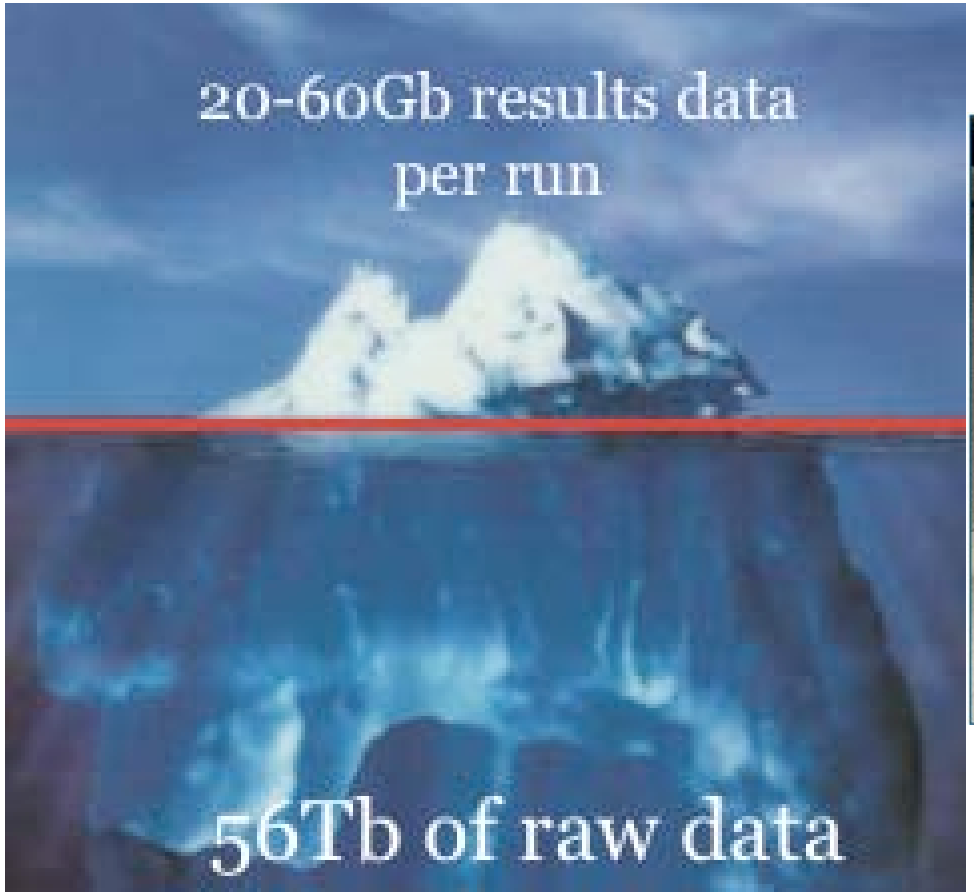
Raw Analysis

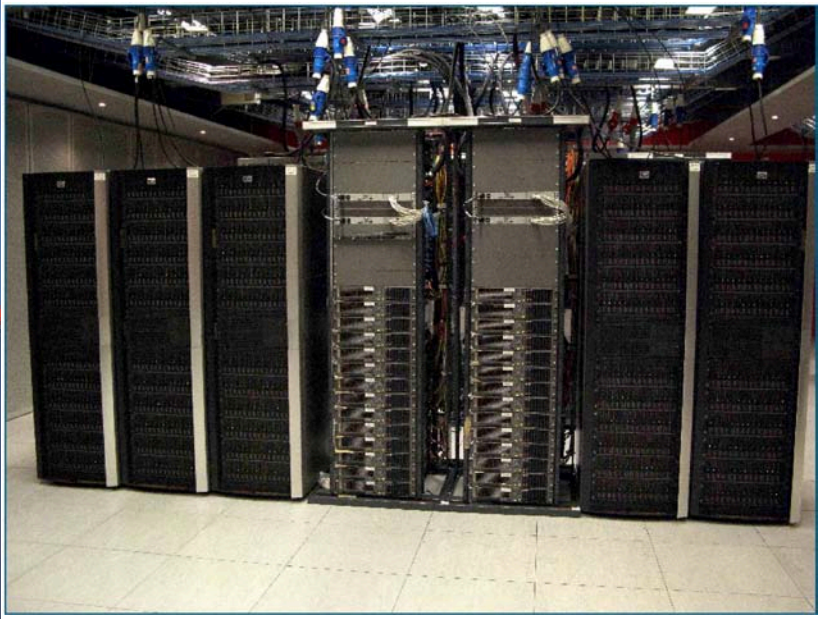
Image processing + base calling

- Base calls + associated quality: Fastq/BAM

From eccb-tutorial-thomas1-101009130915

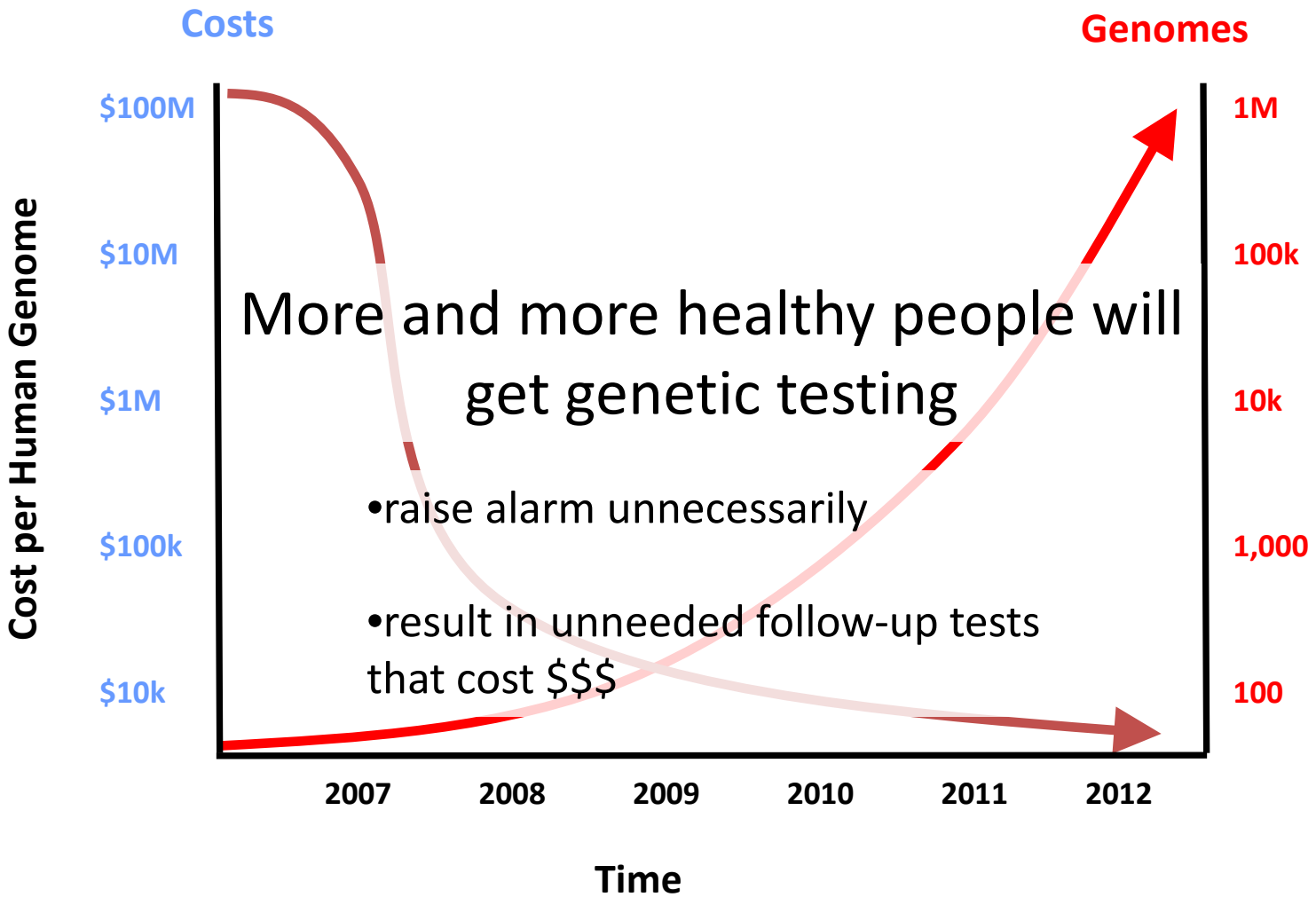
Huge data storage demands



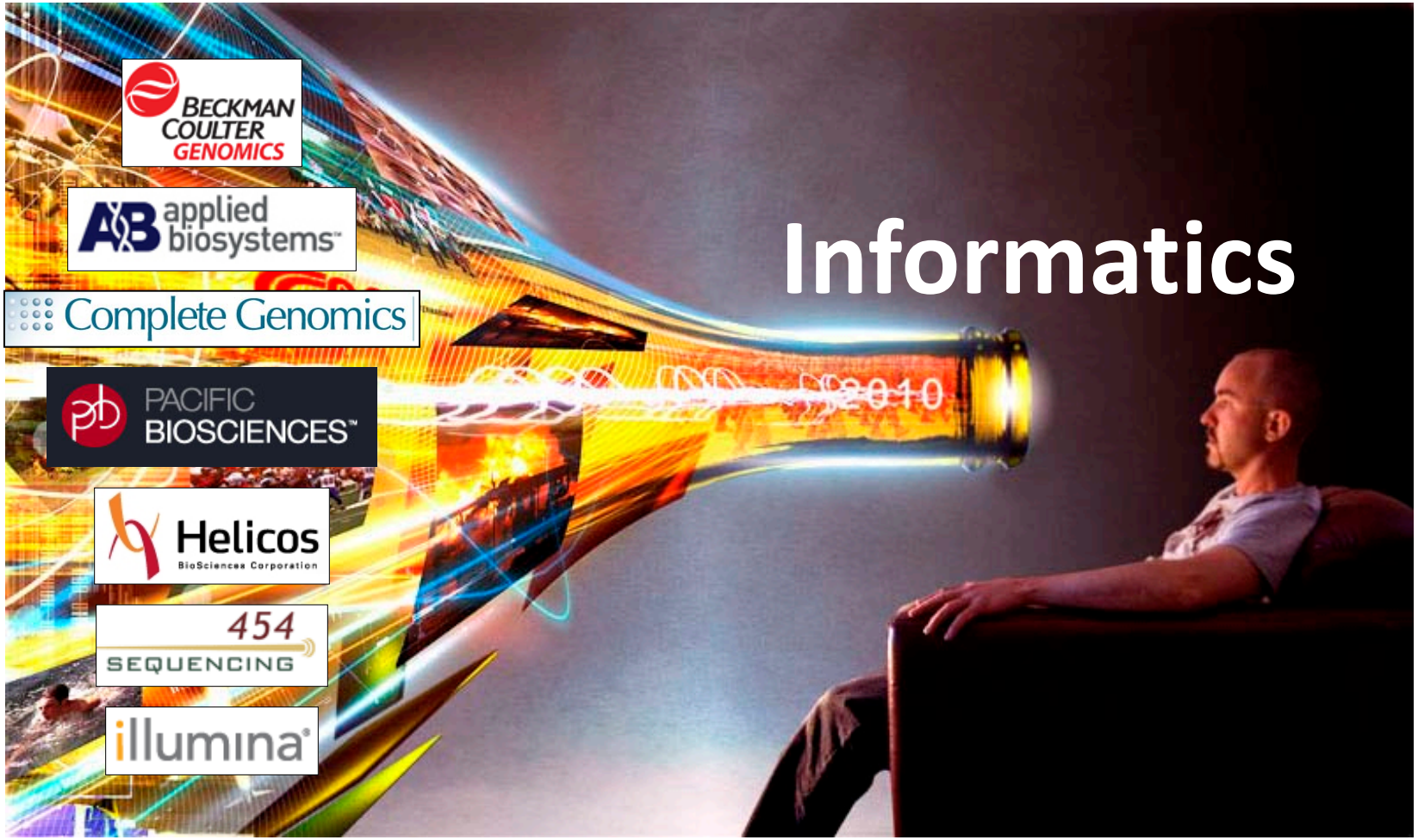


From eccb-tutorial-thomas1-101009130915

Thousands of individual genomes by 2012



The interpretation bottleneck



NGS Data Analysis and Computation I

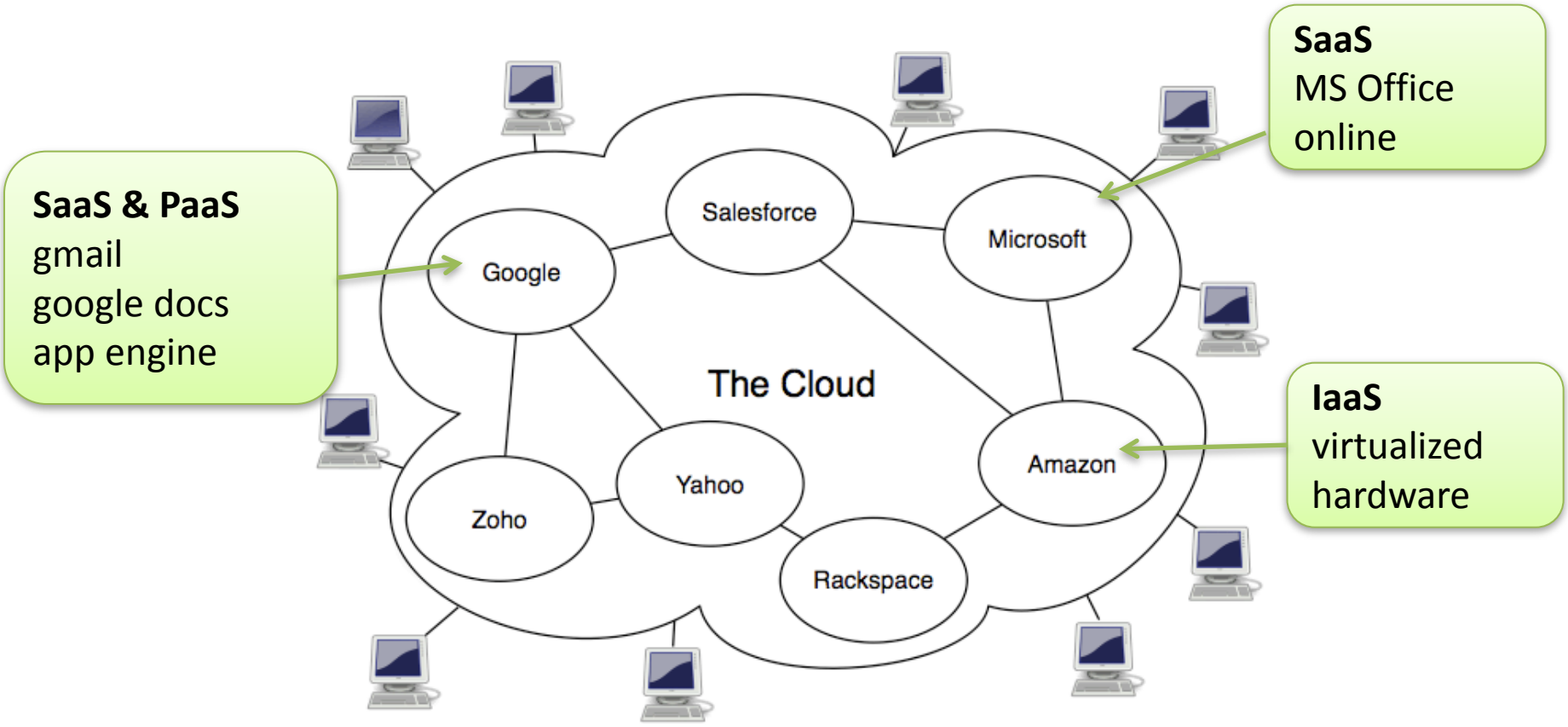
- ☑ Overview
- ☑ Cloud Approaches
- ☐ Quality Control
- ☐ NGS Data Formats
- ☐ Alignment
- ☐ Variant Calling

Will Computers Crash Genomics?



11 FEBRUARY 2011 VOL 331 SCIENCE www.sciencemag.org

Cloud Computing



Definition: Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms, and/or services). These resources can be dynamically re-configured to adjust to a variable load (scale), allowing also for an optimum resource utilization.

Why cloud computing?

1. Ability to scale
   - Job that takes 10 hours on a single server can be done in 1 hour on 10 servers
2. Pay-per-use
   - Only pay for what you use, when you use it
   - Reduces the need to purchase hardware
3. Increased flexibility
   - Variety of server types and operating systems

Cost of Amazon Web Services - instances

- Small ($0.10/hr -> $0.085/hr)
  - 1.7 GB memory & 160 GB storage
  - 1 core
- Large ($0.40/hr -> $0.34/hr)
  - 7.5 GB memory & 850 GB storage
  - 4 cores
- High-CPU Extra Large ($0.80/hr -> $0.68/hr)
  - 7 GB memory & 1.7 TB storage
  - 8 cores

15% price drop in a year. We expect the prices to continue to decline over time

Cloud Management Strategy



Amazon Web Services (AWS)/RightScale Main Account

Project Deployment

Server Configuration

SSH Key Pair



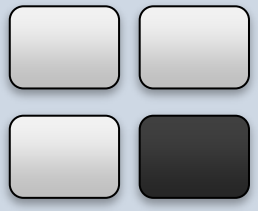
Inelastic

RightScale Sub Account

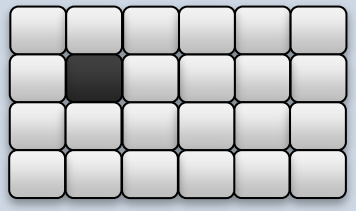
Project Deployment

Server Configuration

RightScale SSH



Managed Elastic



Elastic

Whole genome analysis in the cloud

- Analysis of publicly available NGS short reads from the genome of an African individual (http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?study=SRP000239)
  - Use short read alignment tool for mapping sequence data to the NCBI reference genome
  - Take advantage of the elasticity of the cloud

Where to Start?

Launch AWS Linux Instance



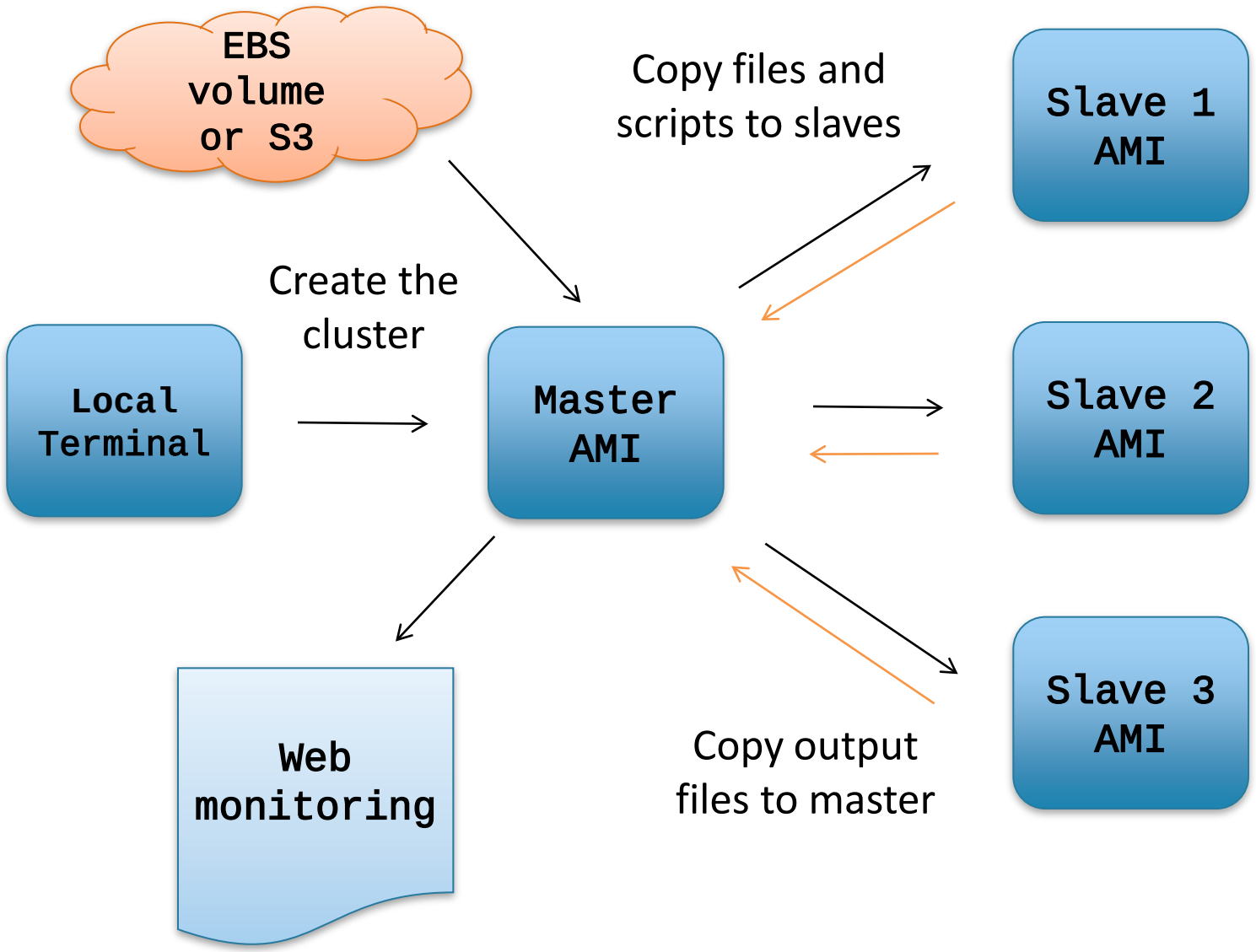
Package into an AMI



- Install short read alignment software
- NCBI reference genome
- Apache web server

Launch identical copies of our instance

Job Distribution Architecture



Costs and Results

- African Genome NGS Data: 370 GB
- Compute Nodes: 25 (5 x 5-core)
- Total Cost: $2,600
  - Cost can be spread over more instances to decrease computation time
  - Cost reduction is expected when using recently developed less computationally intensive alignment algorithms
- Resulting annotated variant file: 22 MB

Best Practices

1. Analyze the details of the project to take full advantage of the cloud
   - Type of OS required, size of data, CPU, memory
2. Create a backup strategy before you launch an instance
3. Only launch an instance when you are ready to start working and shut down the instance when the work is complete
4. Access an instance using a secure connection
5. Actively monitor your account

NGS Data Analysis and Computation I

- ☑ Overview
- ☑ Cloud approaches
- ☑ Quality Control
- ☐ NGS Data Formats
- ☐ Alignment
- ☐ Variant Calling

Basic NGS workflow

DNA Prep

Randomly shear DNA + end repair + size select

Library Prep

Append sequencing adapters

Chip Prep

Layout of library on sequencing slide or wells

Sequencing

For each library fragment – determine the order and identity of bases at either end of the fragment



Raw Analysis

Image processing + base calling

- Base calls + associated quality: Fastq/BAM

From eccb-tutorial-thomas1-101009130915

Quality Control is Essential

Sequencing runs cost money

-Sequencing a poor library on multiple runs – throwing money away!
-Cost of analyzing data $
-CPU time required for alignment and assembly $$
-Cost of storing raw sequence data $$$
-Hours of analysis could be wasted $$$$

Two aspects of sequencing QC

Run/Lane QC

-Is the data off this lane useful?

Library QC

-Is this library worth sequencing more lanes off?

E.g. Have 2x but want another 8x in total
Slightly different set of metrics

Duplicates and False SNPs



Phred

Genome Research **8**: 175-185, 1998

RESEARCH

Base-Calling of Automated Sequencer Traces Using *Phred.* I. Accuracy Assessment

Brent Ewing,[1] LaDeana Hillier,[2] Michael C. Wendl,[2] and Phil Green[1,3]

[1]Department of Molecular Biotechnology, University of Washington, Seattle, Washington 98195-7730 USA;
[2]Genome Sequencing Center, Washington University School of Medicine, Saint Louis, Missouri 63108 USA

8:175–185 ©1998 by Cold Spring Harbor Laboratory Press ISSN 1054-9803/98 $5.00; www.genome.org

Phred

Genome Research **8**: 186-194, 1998

RESEARCH

Base-Calling of Automated Sequencer Traces Using *Phred.* II. Error Probabilities

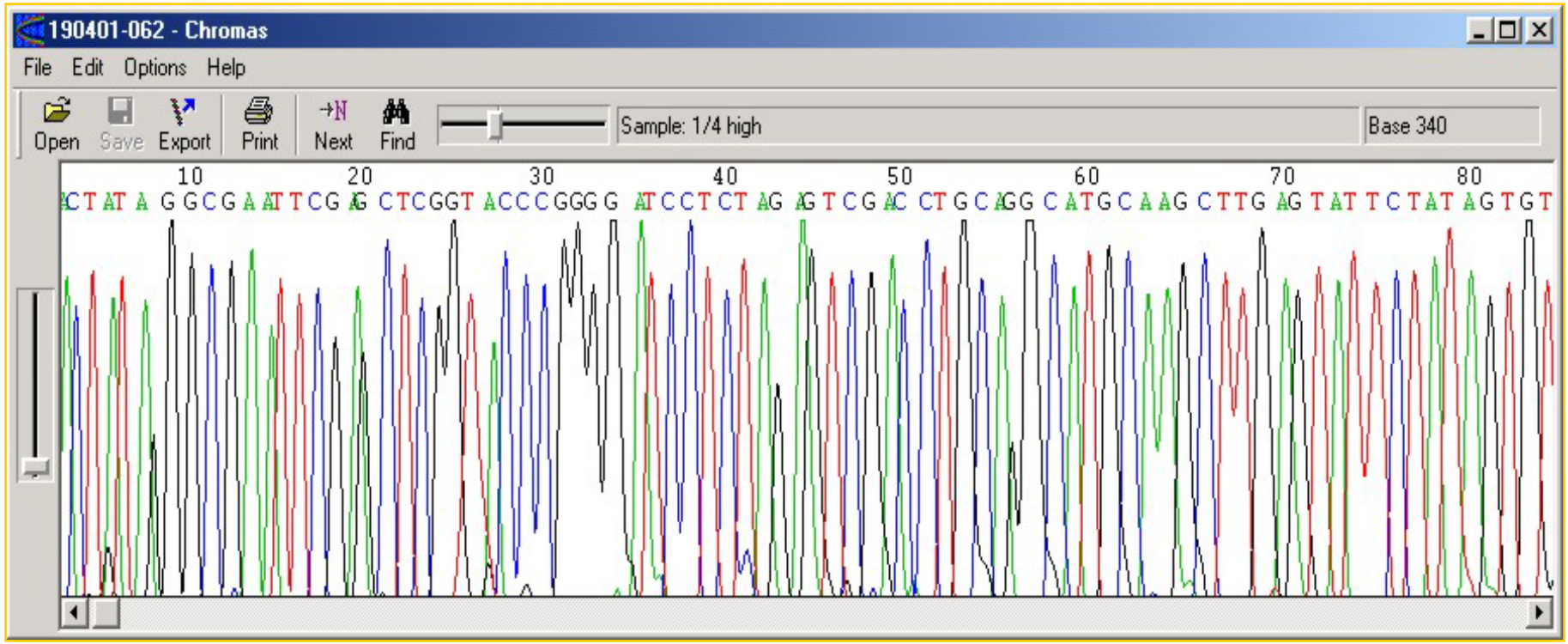
Brent Ewing and Phil Green[1]

Department of Molecular Biotechnology, University of Washington, Seattle, Washington 98195-7730 USA

8:186–194 ©1998 by Cold Spring Harbor Laboratory Press ISSN 1054-9803/98 $5.00; www.genome.org

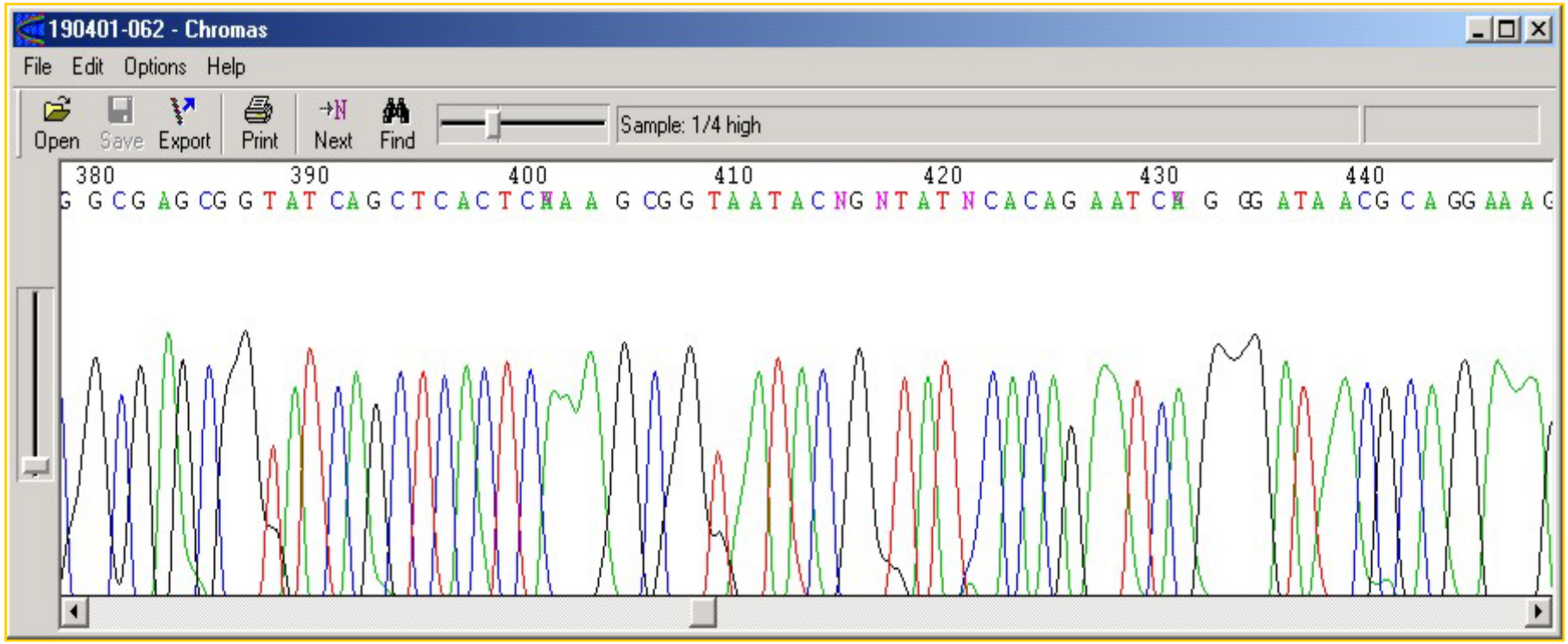
Trace File

High quality region – no ambiguities (Ns)



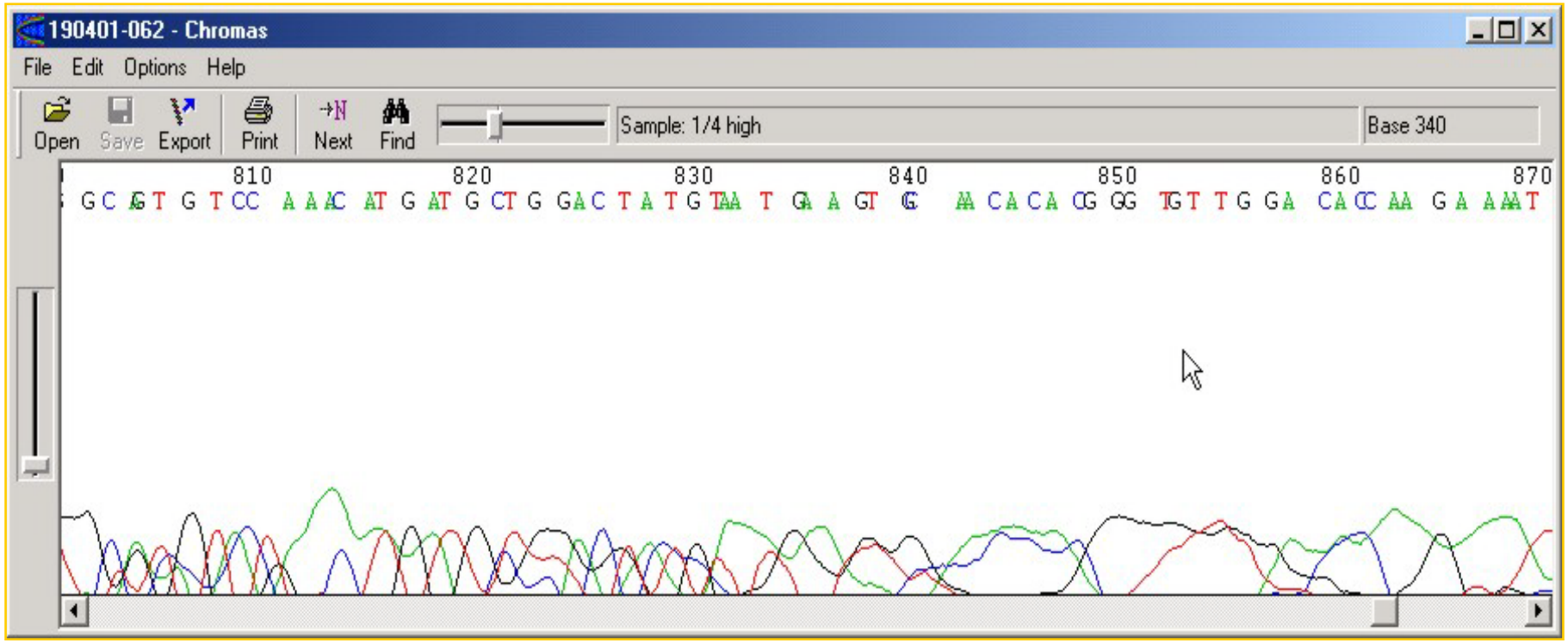
Trace File

Medium quality region – some ambiguities (Ns)



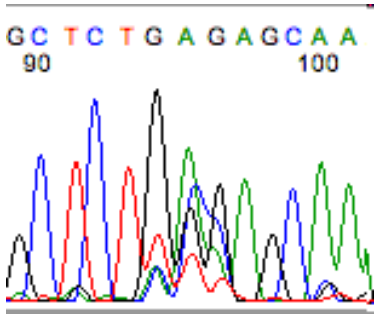
Trace File

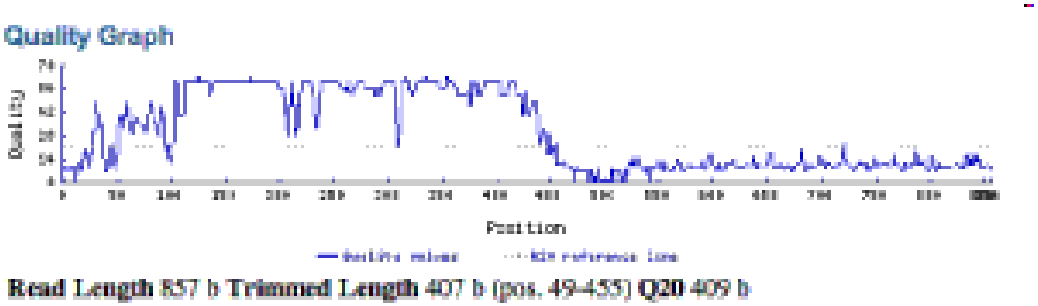
Poor quality region – low confidence



Phred Quality Scores

- Phred is a program that assigns a quality score to each base in a sequence. These scores can then be used to trim bad data from the ends, and to determine how good an overlap actually is.
  - there are much improved algorithms now, but the phred quality score is still widely used.
- Phred scores are logarithmically related to the probability of an error: a score of 10 means a 10% error probability; 20 means a 1% chance, 30 means a 0.1% chance, etc.
  - Q = -10 log P, where Q is the phred score and P is the probability that the base was called incorrectly.
  - A score of 20 is generally considered the minimum acceptable score.





Phred Algorithm

- Phred uses Fourier analysis (decomposing the data into a series of sine waves) to examine chromatogram trace data.
  - First, find the expected position of each peak, assuming they are supposed to be evenly spaced, with no compressions or other factors altering peak positions.
  - Next, find actual center and area of each peak.
  - Finally, match observed peaks to expected. This involves splitting some peaks and ignoring others.
- Assigning a quality score involves comparing various parameters determined for each peak with data that was generated from known sequences run under a wide variety of conditions (i.e., based on ad hoc observations and not theory).
- Since all four traces (A, C, G, T) are examined separately, phred generates a best-guess sequence. Phred output is a file where each line contains one base and its quality score.

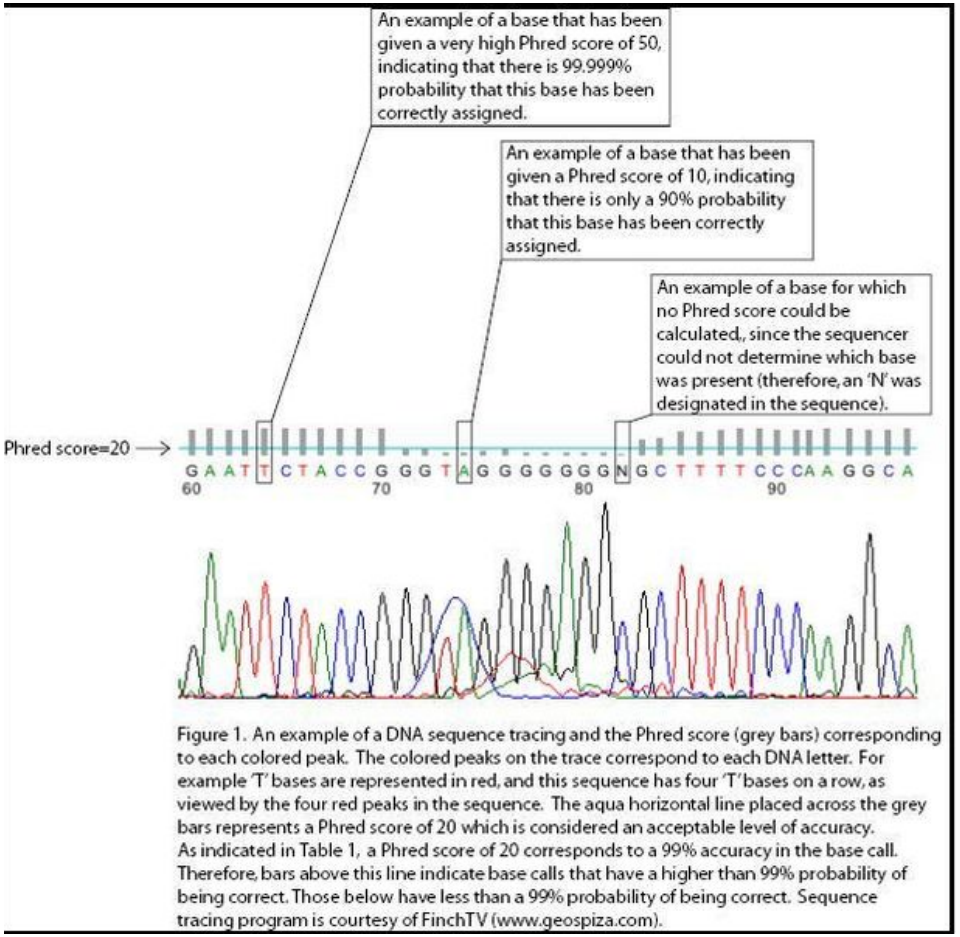


Figure 1. An example of a DNA sequence tracing and the Phred score (grey bars) corresponding to each colored peak. The colored peaks on the trace correspond to each DNA letter. For example 'T' bases are represented in red, and this sequence has four 'T' bases on a row, as viewed by the four red peaks in the sequence. The aqua horizontal line placed across the grey bars represents a Phred score of 20 which is considered an acceptable level of accuracy. As indicated in Table 1, a Phred score of 20 corresponds to a 99% accuracy in the base call. Therefore, bars above this line indicate base calls that have a higher than 99% probability of being correct. Those below have less than a 99% probability of being correct. Sequence tracing program is courtesy of FinchTV (www.geospiza.com).

NGS Data Analysis and Computation I

- ☑ Overview
- ☑ Cloud Approaches
- ☑ Quality Control
- ☑ NGS Data formats
- ☐ Alignment
- ☐ Variant Calling

FASTQ format

```
@HWI-EAS225:3:1:2:854#0/1
GGGGGGGAAGTCGGCAAAATAGATCCGTAACTTCGGG
+HWI-EAS225:3:1:2:854#0/1
a`abbbbbabaabbbababb^``[aaa`_N]b^ab^``a
@HWI-EAS225:3:1:2:1595#0/1
GGGAAGATCTCAAAAACAGAAGTAAAACATCGAACG
+HWI-EAS225:3:1:2:1595#0/1
a`abbbababbbabbbbbbbabb`aaababab\aa_`
```

FASTQ format

- Each read is represented by four lines

(1) @ followed by read ID
(2) Sequence
(3) + optionally followed by repeated read ID
(4) Quality line
    - Same length as sequence
    - Each character encodes the quality of the base

FASTQ phred-based call qualities

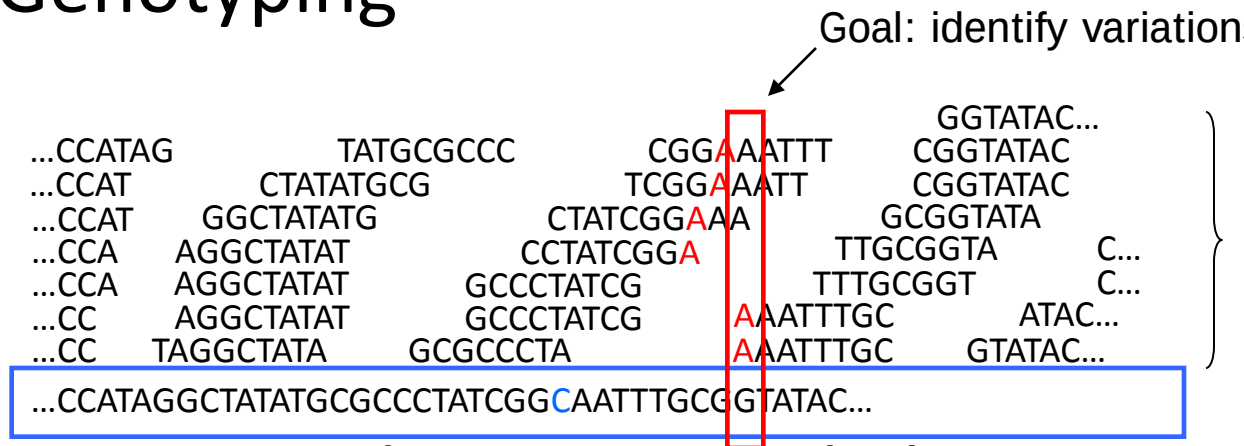
quality score Q_{phred}	error prob. p	characters
0 .. 9	1 .. 0.13	!"#$%&'()*
10 .. 19	0.1 .. 0.013	+,-./01234
20 .. 29	0.01 .. 0.0013	56789:;<=>
30 .. 39	0.001 .. 0.00013	?@ABCDEFGH
40	0.0001	I

NGS Data Analysis and Computation I

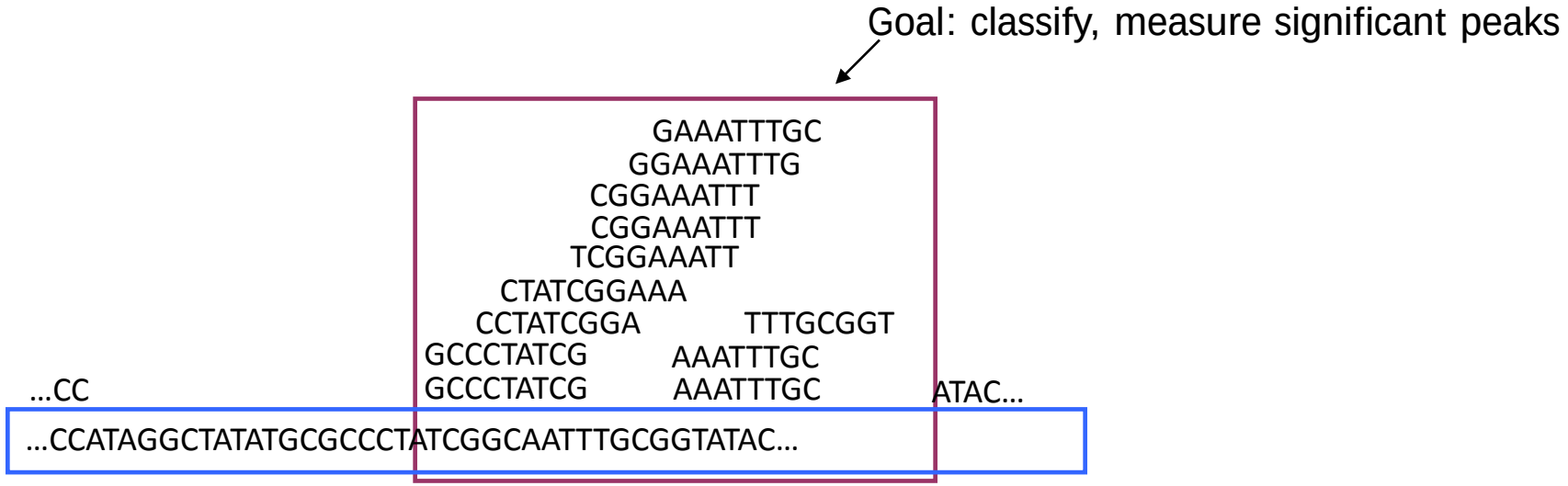
- ☑ Overview
- ☑ Cloud Approaches
- ☑ Quality Control
- ☑ NGS Data Formats
- ☑ Alignment
- ☐ Variant Calling

Short Read Applications

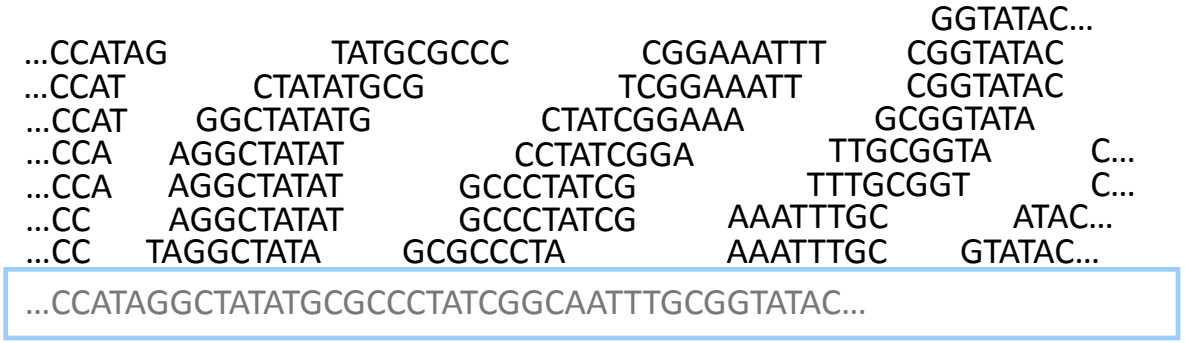
- Genotyping

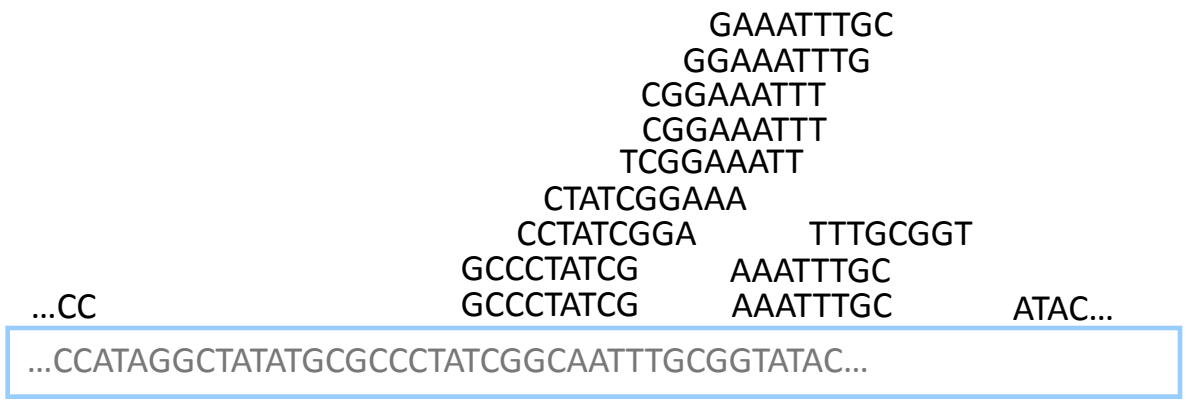


- RNA-seq, ChIP-seq, Methyl-seq



Short Read Applications





Finding the alignments is a performance bottleneck

Aligner	Description	URL
Illumina platform		
ELAND	Vendor-provided aligner for Illumina data	http://www.illumina.com
Bowtie	Ultrafast, memory-efficient short-read aligner for Illumina data	http://bowtie-bio.sourceforge.net
Novoalign	A sensitive aligner for Illumina data that uses the Needleman–Wunsch algorithm	http://www.novocraft.com
SOAP	Short oligo analysis package for alignment of Illumina data	http://soap.genomics.org.cn/
MrFAST	A mapper that allows alignments to multiple locations for CNV detection	http://mrfast.sourceforge.net/
SOLiD platform		
Corona-lite	Vendor-provided aligner for SOLiD data	http://solidsoftwaretools.com
SHRiMP	Efficient Smith–Waterman mapper with colorspace correction	http://compbio.cs.toronto.edu/shrimp/
454 Platform		
Newbler	Vendor-provided aligner and assembler for 454 data	http://www.454.com
SSAHA2	SAM-friendly sequence search and alignment by hashing program	http://www.sanger.ac.uk/resources/software
BWA-SW	SAM-friendly Smith–Waterman implementation of BWA for long reads	http://bio-bwa.sourceforge.net
Multi-platform		
BFAST	BLAT-like fast aligner for Illumina and SOLiD data	http://bfast.sourceforge.net
BWA	Burrows-Wheeler aligner for Illumina, SOLiD, and 454 data	http://bio-bwa.sourceforge.net
Maq	A widely used mapping tool for Illumina and SOLiD; now deprecated by BWA	http://maq.sourceforge.net

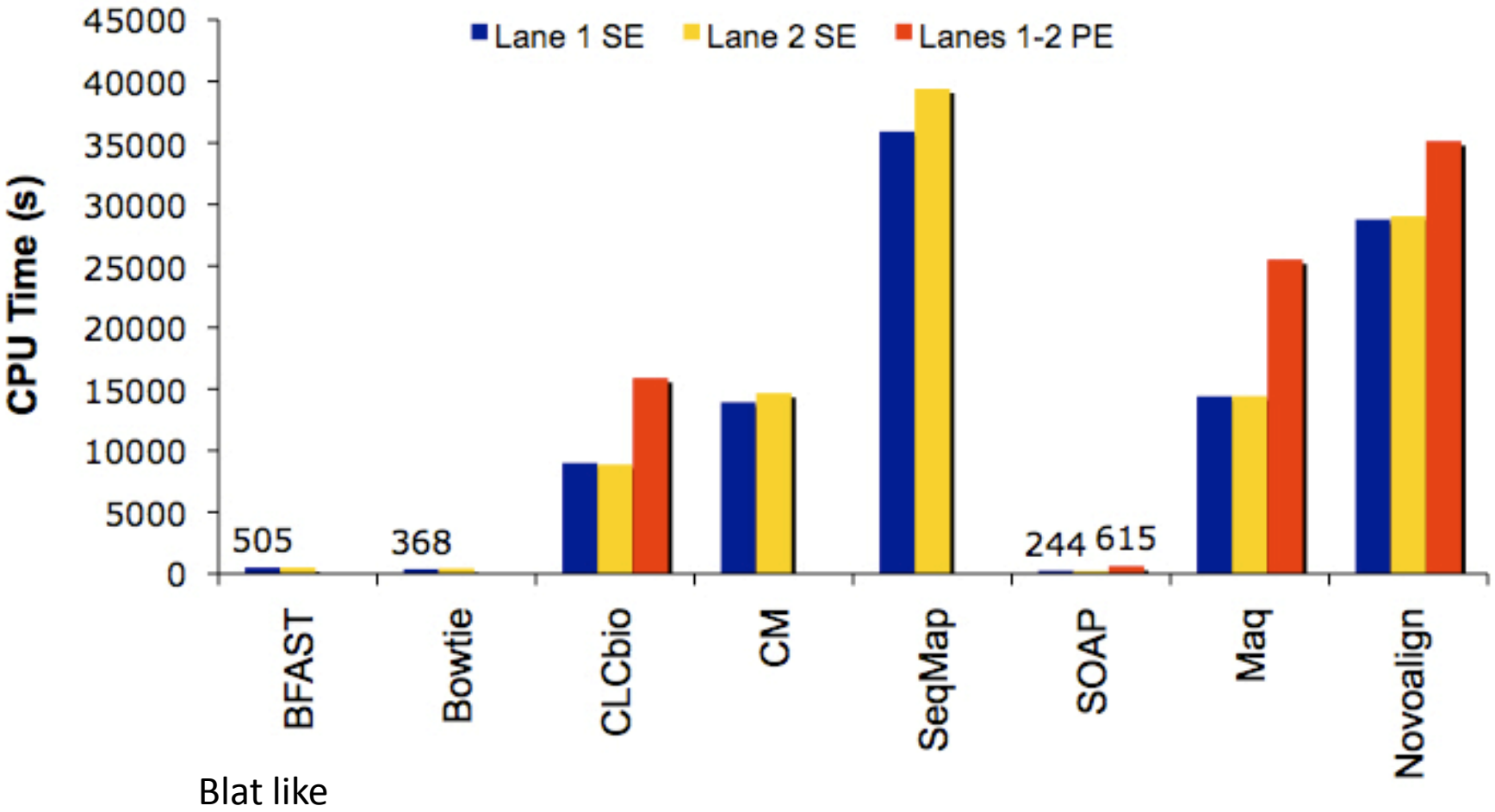
Koboldt DC et al. Brief Bioinform 2010 Sep;11(5):484-98

Aligners Evaluated

Aligner	Version	Developer	License
BFAST	0.3.1	UCLA	Academic
Bowtie	0.9.8	Salzberg/UMD	GNUPL
cross_match	1.080721	U. Wash.	Academic
CELL	2.0	CLC bio	Commercial
Maq	0.7.1	Sanger	GNUPL
Novoalign	2.00.12	Novocraft	Academic
RMAP	0.41	CSHL	Public
SeqMap	1.0.12	Stanford	Academic
SHRiMP	1.1.0	U. Toronto	Public
SOAP	2.01	BGI	Academic

See www.massgenomics.org

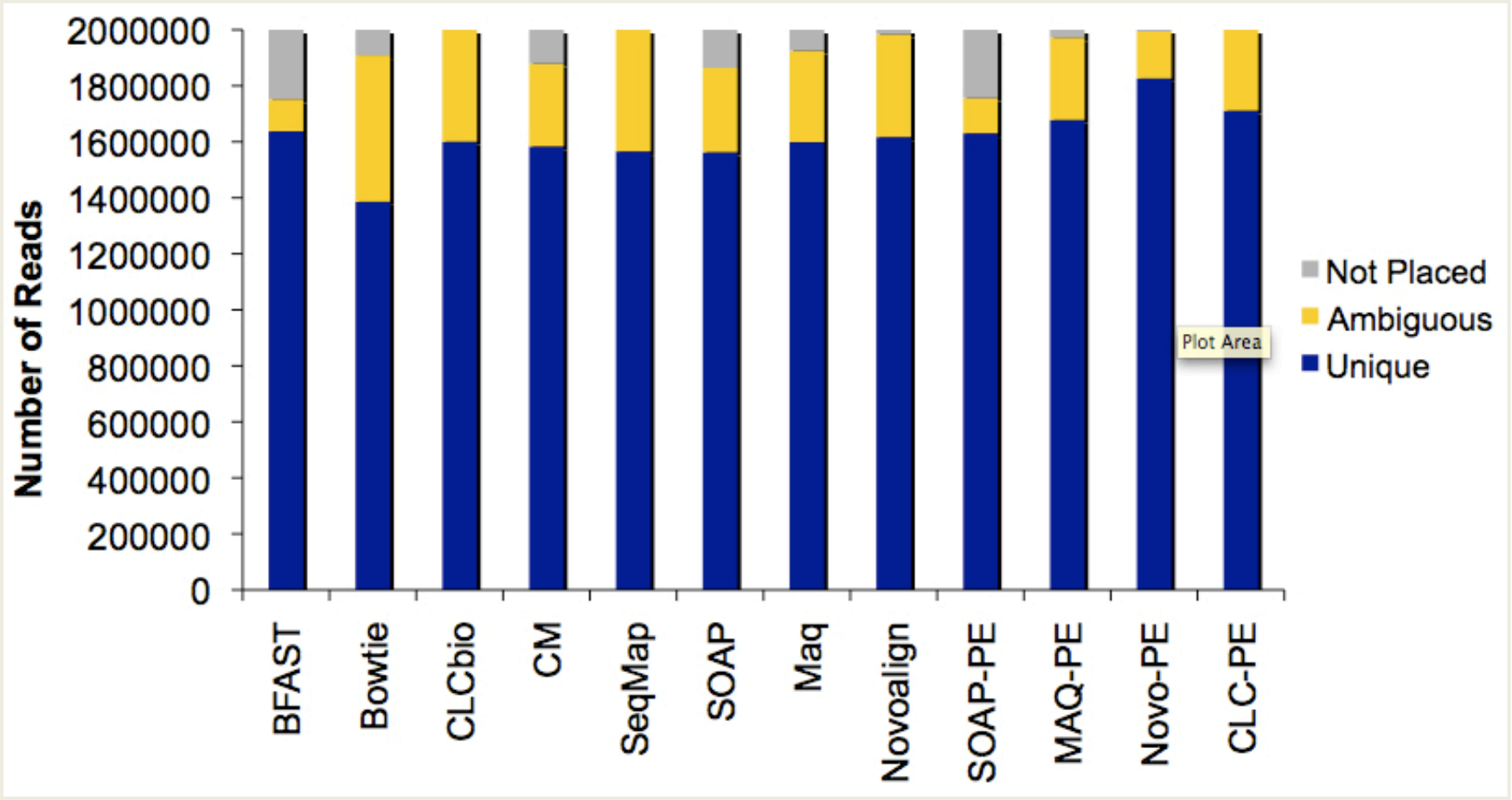
CPU Time 2M Reads to Hs36: SE/PE



Blat like

- Benchmark: Maq (~8 hours)

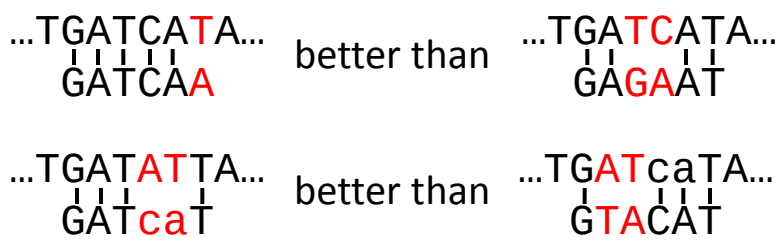
Placement: 2M Reads on Hs36



- Most aligners place ~80% of reads uniquely in Hs36.

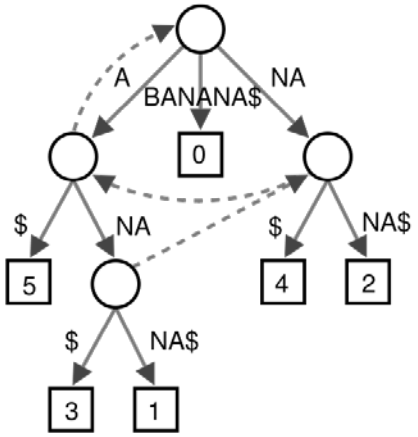
Short Read Alignment

- Given a reference and a set of reads, report at least one "good" local alignment for each read if one exists
  - Approximate answer to question: **where** in genome did read originate?
- What is "good"? For now, we concentrate on:
  - Fewer mismatches = better
  - Failing to align a low-quality base is better than failing to align a high-quality base

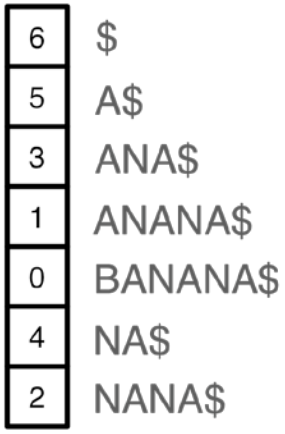


Indexing

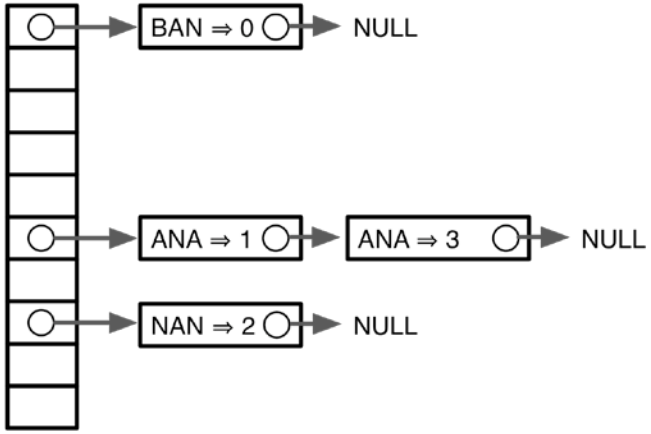
- Genomes and reads are too large for direct approaches like dynamic programming
- *Indexing* is required



Suffix tree



Suffix array



Seed hash tables

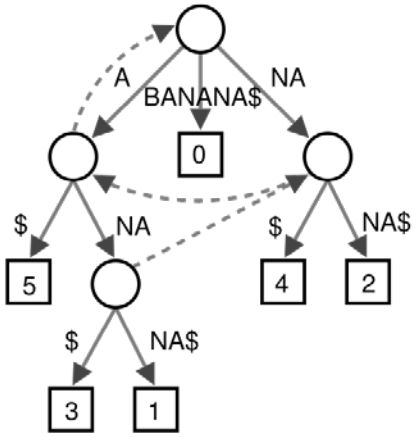
Many variants, incl. spaced seeds

- Choice of index is key to performance

Indexing

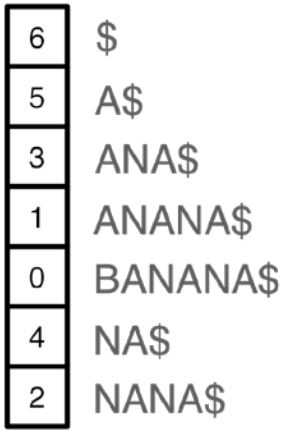
- Genome indices can be big. For human:

Suffix Tree



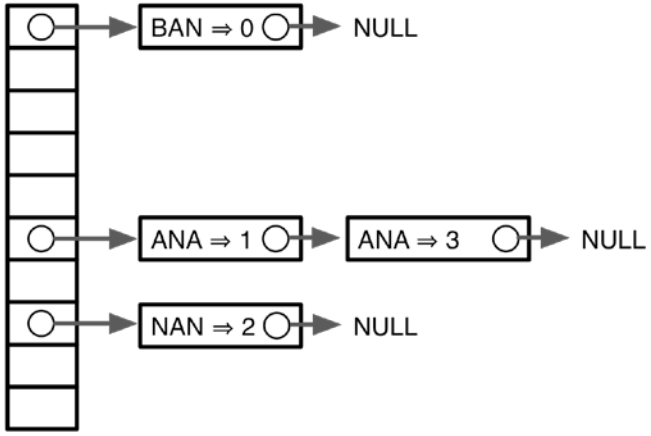
> 35 GBs

Suffix Array



> 12 GBs

k-mer Hash Tables



> 12 GBs

- Large indices necessitate painful compromises
  1. Require big-memory machine
  2. Use secondary storage
  3. Build new index each run
  4. Subindex and do multiple passes

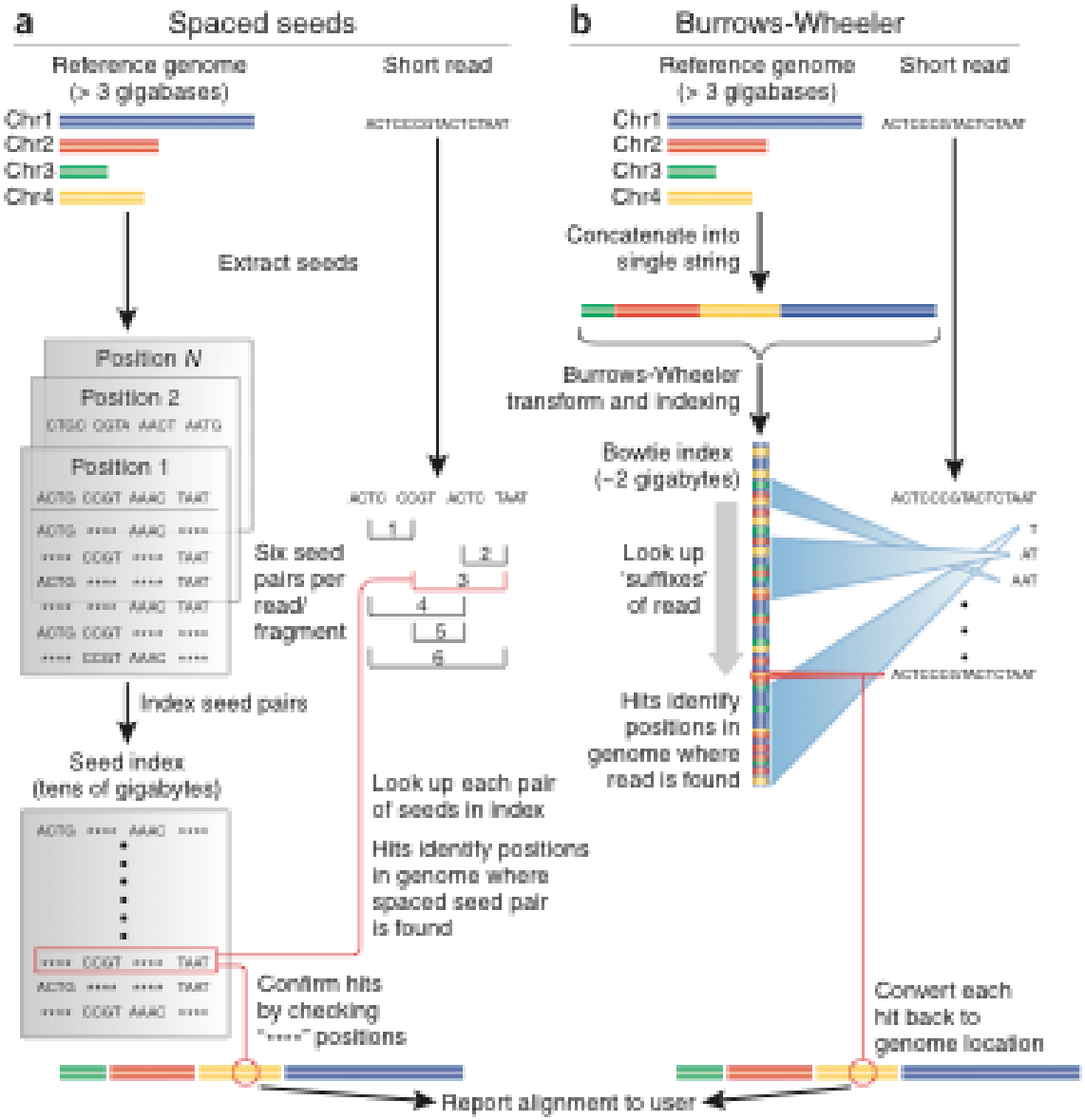
Mapping Reads Back

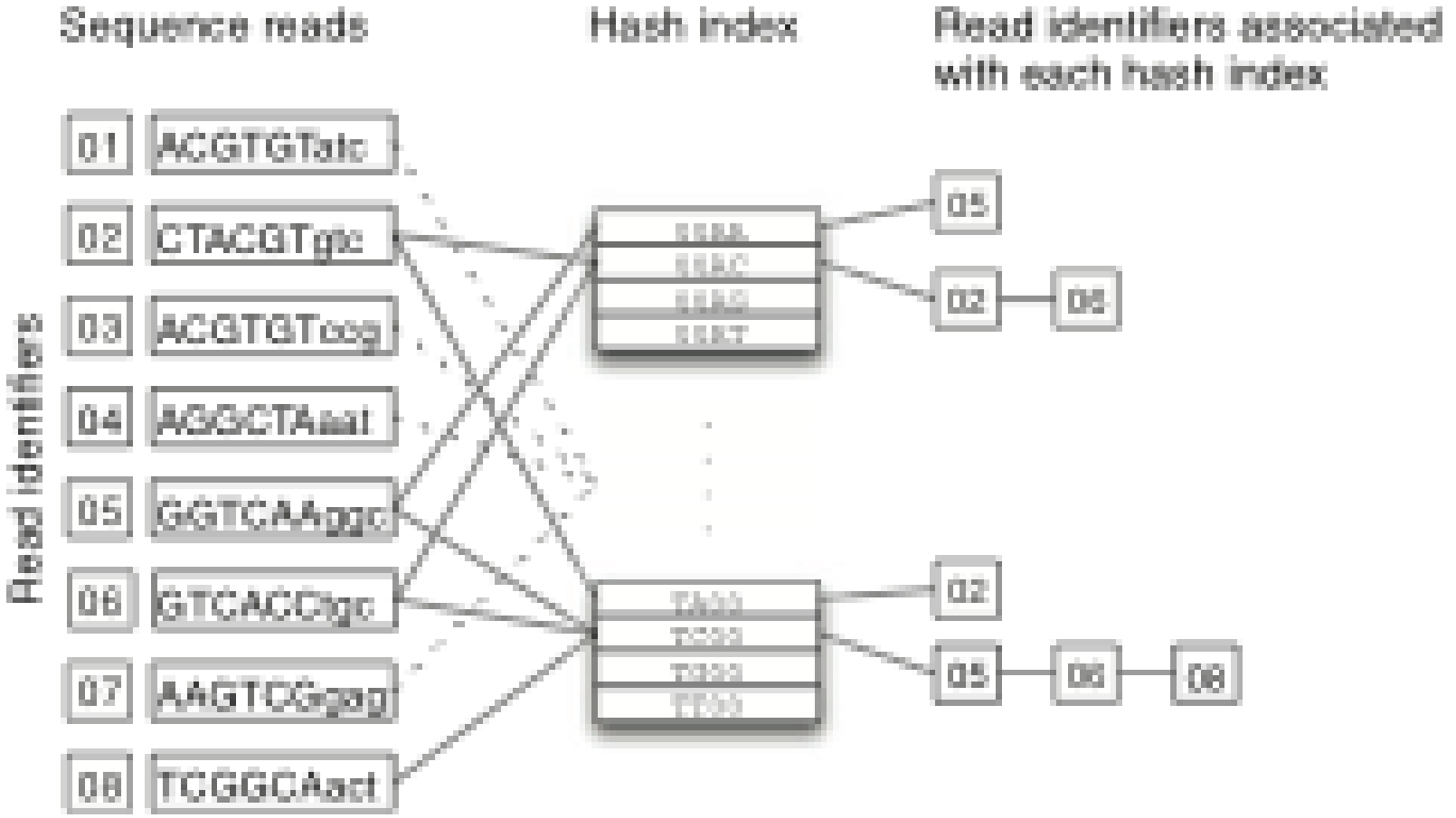
- Hash Table (Lookup table)
  - FAST, but requires perfect matches
- Array Scanning
  - Can handle mismatches, but not gaps
- Dynamic Programming (Smith Waterman, Forward, Viterbi)
  - Indels
  - Mathematically optimal solution
  - Slow (most programs use Hash Mapping as a prefilter)
- Burrows-Wheeler Transform (BW Transform)
  - FAST (memory efficient)
  - But for gaps/mismatches, it lacks sensitivity

Short read alignment

- Large and ever growing number of implementations for short-read sequence alignment
- Fundamental algorithms can be divided into two approaches
  - Hash table based implementations
  - Burrows-Wheeler transform (BWT)
- Both approaches apply to sequence and color space and all technologies
- Heuristics to find potential locations on the genome
- Slower more accurate alignment run on a subset of potential locations
- Similar strategy to traditional read alignment algorithms: Blat, Blast, SSAHA2
- Constant trade-off: speed vs. sensitivity
- Guaranteed high accuracy will take longer

MAQ vs Bowtie



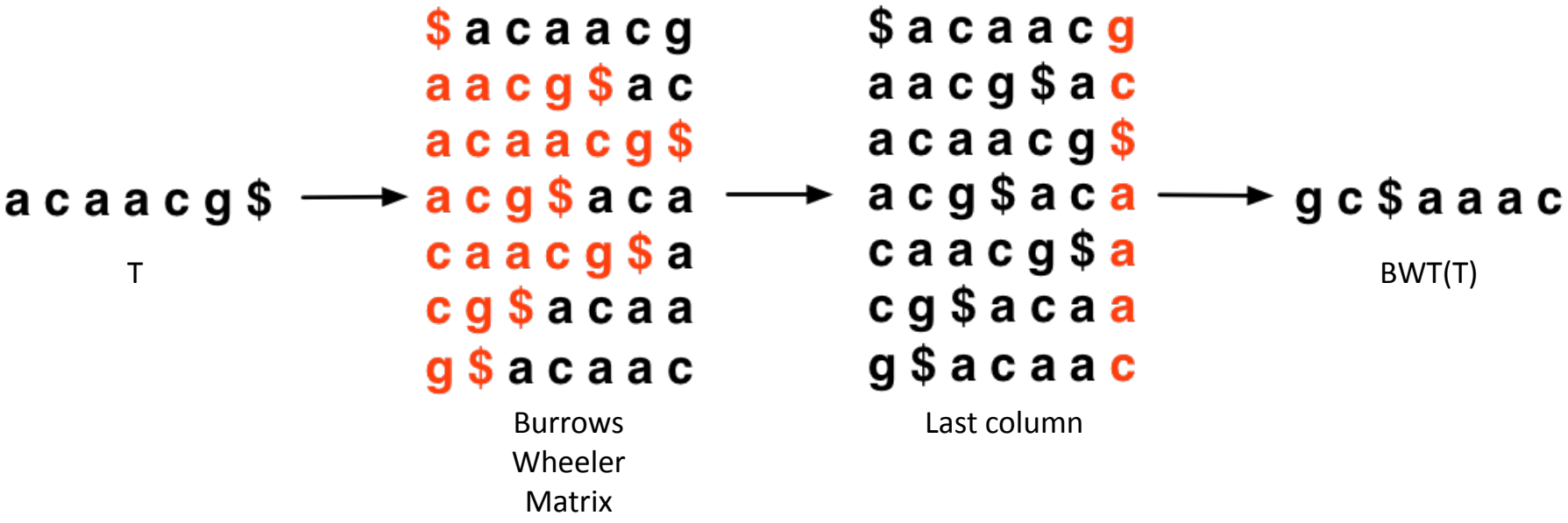

Sequence reads
Hash index
Read identifiers associated with each hash index
Read identifiers
01
ACGTGTatc
02
CTACGTgtc
03
ACGTGTccg
04
AGGCTAaat
05
GGTCAAggc
06
GTCACCtgc
07
AAGTCGgag
08
TCGGCAact
05
02
06
02
05
06
08

Hash table tidbits

- Hash is typically built from a set of seeds spaced along the reference or read sequence
- Seeds used in the hash table creation and the reads have been associated with the region of the genome
- Specialized and accurate alignment algorithm is used to determine the exact placement
  - Gapped and ungapped versions of Smith-Waterman
  - Use the base qualities in order to determine the most likely alignment of a read
  - Use information about where the mate pair aligns to
  - Maq: Require 1 mate to align ungapped and can do full Smith-Waterman to align its mate (e.g. indel in the mate)
- Mapping quality
  - A probability measure of the alignment being incorrect
  - Single end mapping quality vs. paired end mapping quality
  - Low complexity regions typically have low mapping qualities

Burrows-Wheeler Transform

- Reversible permutation used originally in compression

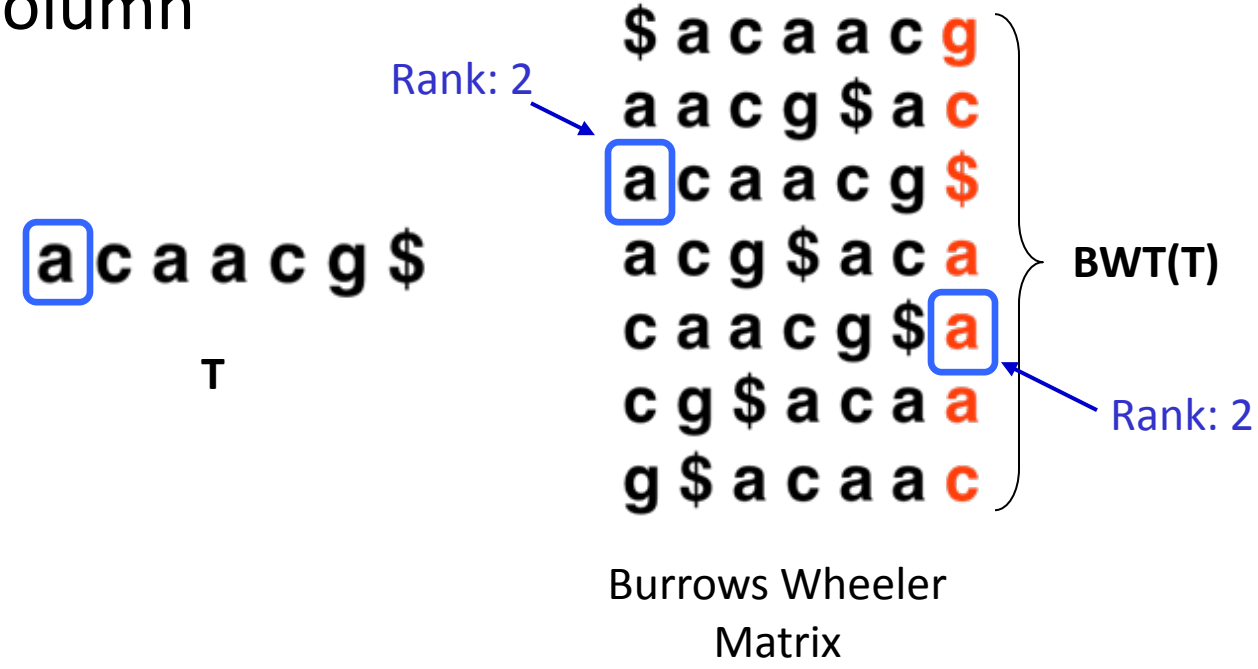


- Once BWT(T) is built, *all else shown here is discarded*
  - Matrix will be shown for illustration only

Burrows M, Wheeler DJ: A block sorting lossless data compression algorithm. *Digital Equipment Corporation, Palo Alto, CA* 1994, Technical Report 124; 1994

Burrows-Wheeler Transform

- Property that makes BWT(T) reversible is "LF Mapping"
  - i^{th} occurrence of a character in **L**ast column is same *text* occurrence as the i^{th} occurrence in **F**irst column



Burrows-Wheeler Transform

- To recreate T from BWT(T), repeatedly apply rule:

  T = **BWT**[**LF**(i)] + **T**; i = **LF**(i)

  – Where **LF**(i) maps row i to row whose first character corresponds to i's last per LF Mapping



- Could be called "unpermute" or "walk-left" algorithm

- Consider transforming a long English text frequently containing the word "the". Sorting the rotations of this text will often group rotations starting with "he " together, and the last character of that rotation (which is also the character before the "he ") will usually be "t", so the result of the transform would contain a number of "t" characters along with the perhaps less-common exceptions (such as if it contains "Brahe ") mixed in. So it can be seen that the success of this transform depends upon one value having a high probability of occurring before a sequence, so that in general it needs fairly long samples (a few kilobytes at least) of appropriate data (such as text).
- The remarkable thing about the BWT is not that it generates a more easily encoded output—an ordinary sort would do that—but that it is *reversible*, allowing the original document to be re-generated from the last column of data.

FM Index

- Ferragina & Manzini propose "FM Index" based on BWT
- Observed:
  - LF Mapping also allows *exact matching* within T
  - **LF**(i) can be made fast with *checkpointing*
  - …and more (see FOCS paper)

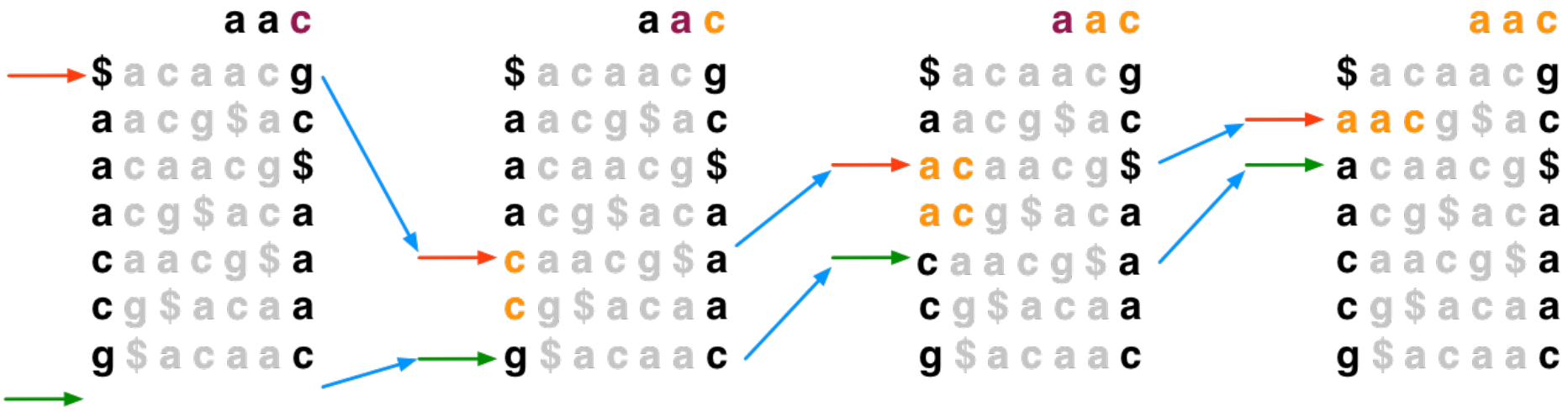
- Ferragina P, Manzini G: Opportunistic data structures with applications. *FOCS. IEEE Computer Society; 2000.*
- Ferragina P, Manzini G: An experimental study of an opportunistic index. *SIAM symposium on Discrete algorithms*. Washington, D.C.; 2001.

Exact Matching with FM Index

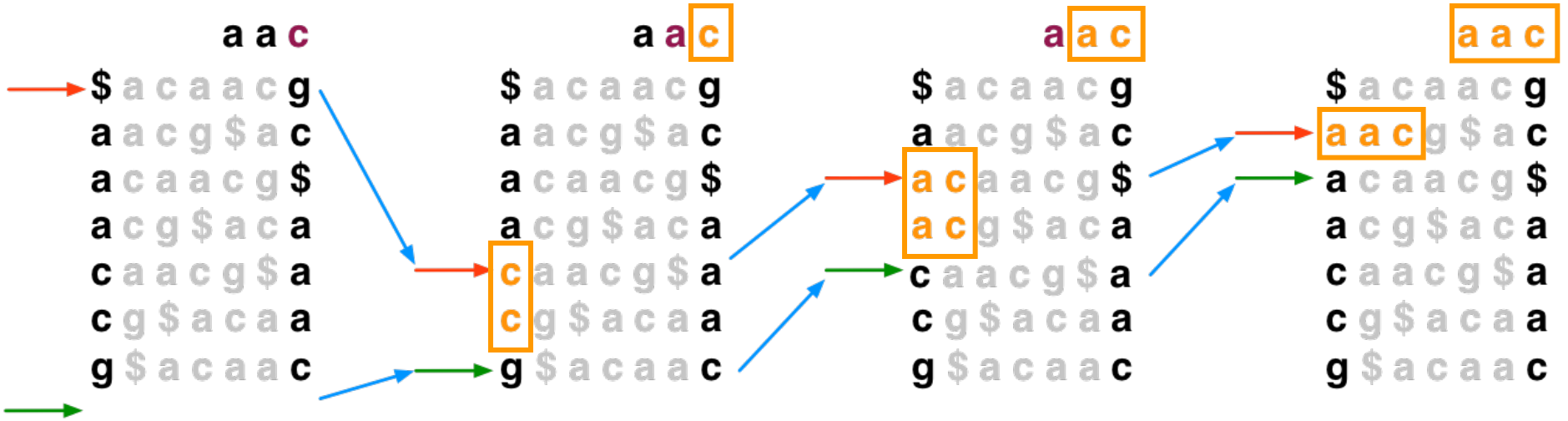
- To match Q in T using BWT(T), repeatedly apply rule:

  top = **LF**(**top**, **qc**); **bot** = **LF**(**bot**, **qc**)

  – Where **qc** is the next character in Q (right-to-left) and **LF**(i, **qc**) maps row i to the row whose first character corresponds to i's last character *as if it were* ***qc***

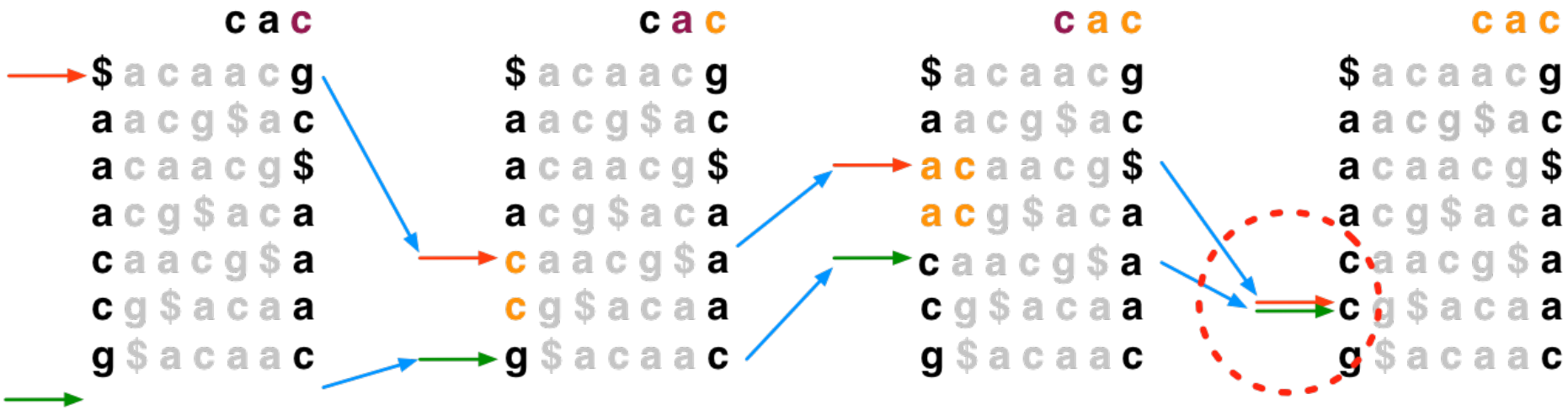


Exact Matching with FM Index



- In progressive rounds, **top** & **bot** delimit the range of rows beginning with progressively longer suffixes of Q

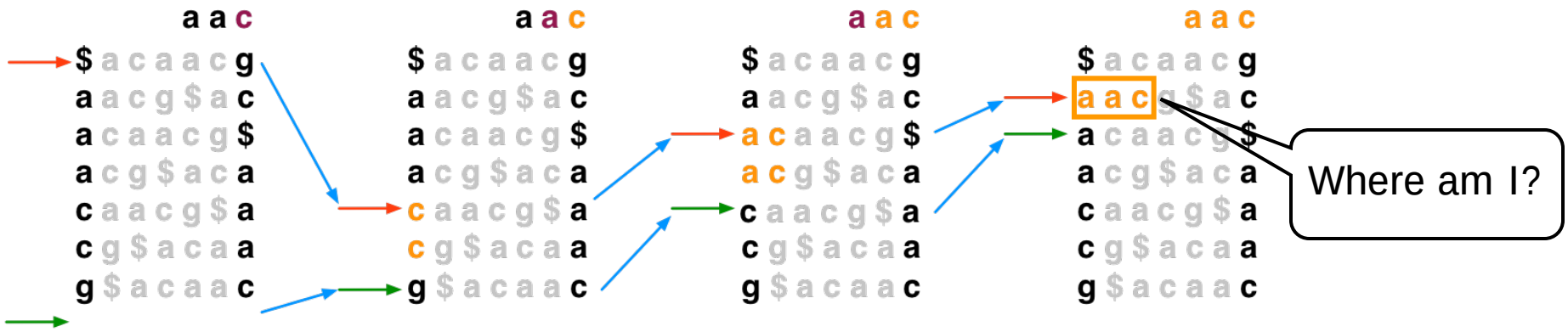
Exact Matching with FM Index



- If range becomes empty (**top** = **bot**) the query suffix (and therefore the query) does not occur in the text

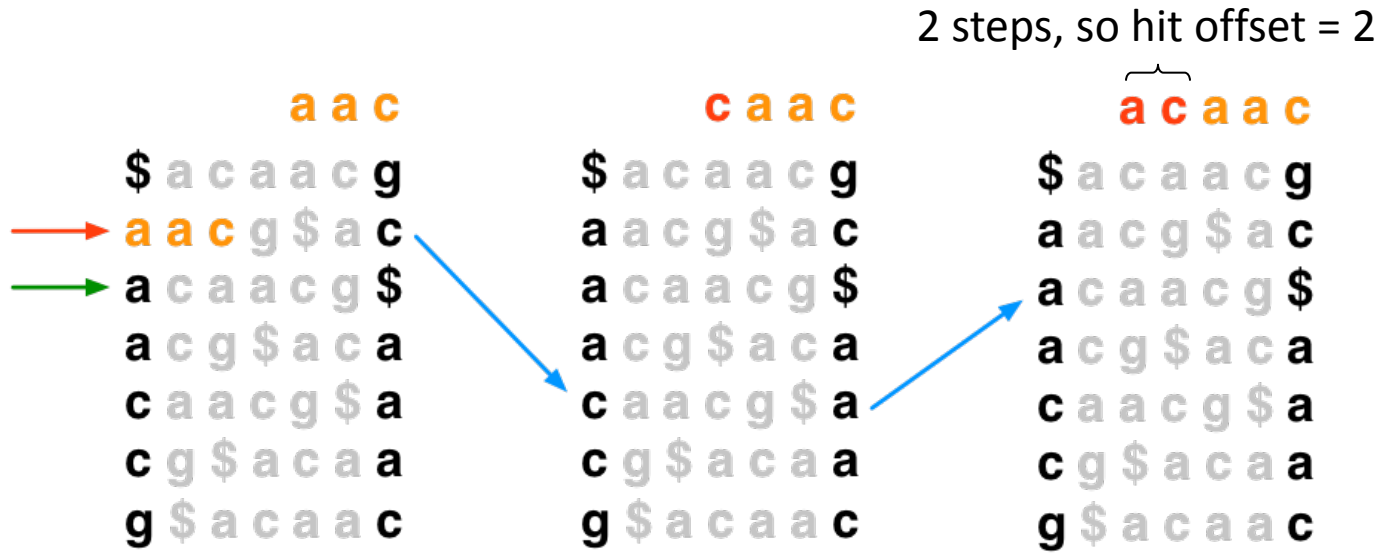
Rows to Reference Positions

- Once we know a row contains a legal alignment, how do we determine its position in the reference?



Rows to Reference Positions

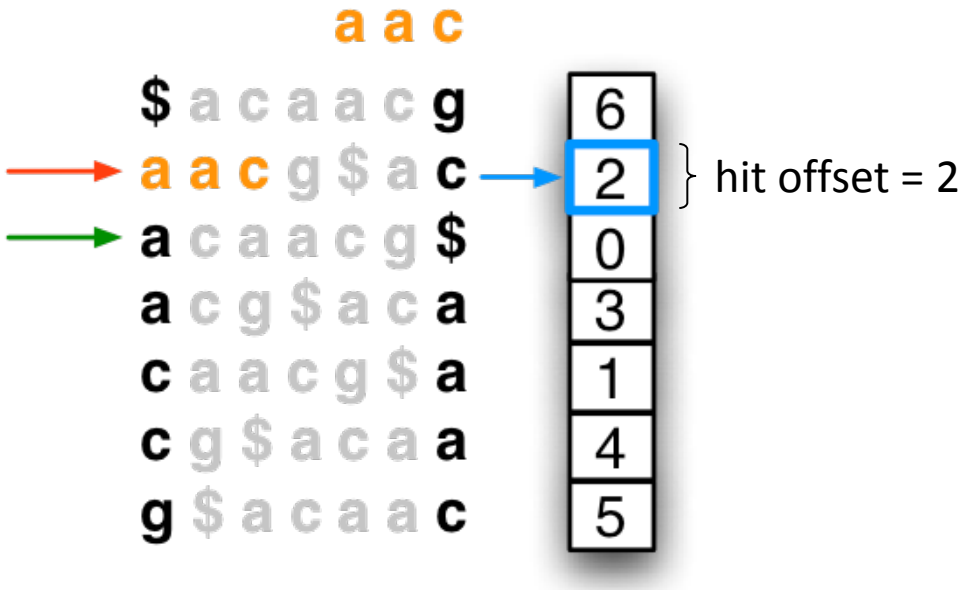
- Naïve solution 1: Use “walk-left” to walk back to the beginning of the text; number of steps = offset of hit



- Linear in length of text in general – too slow

Rows to Reference Positions

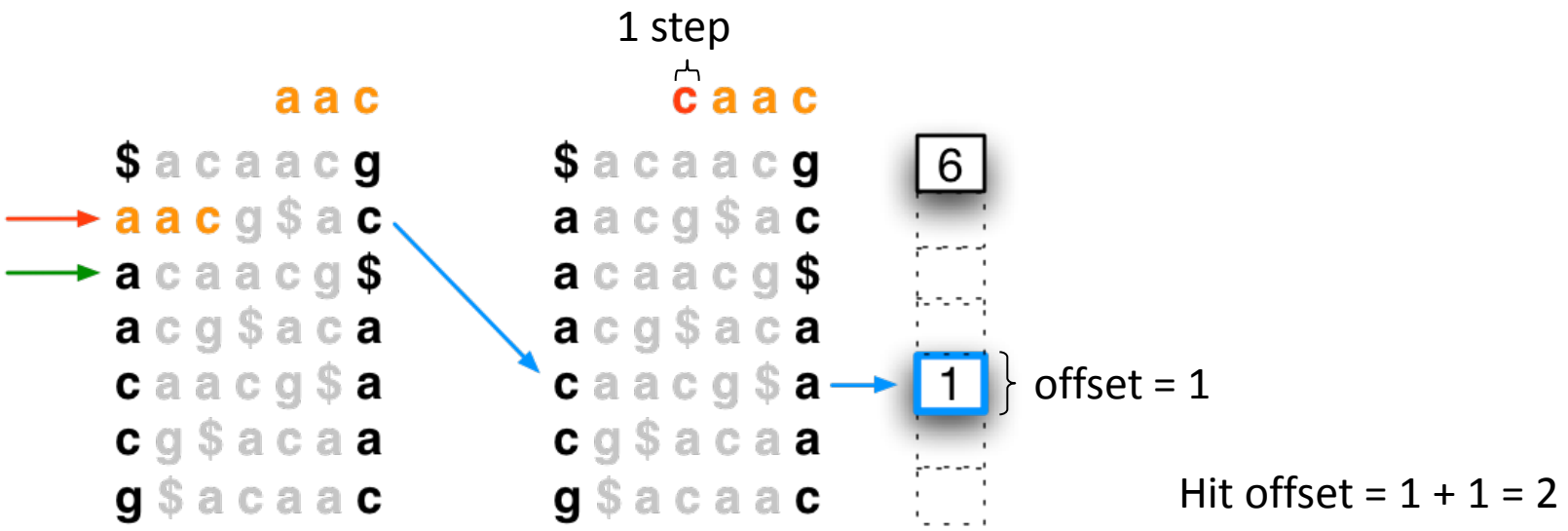
- Naïve solution 2: Keep whole suffix array in memory. Finding reference position is a lookup in the array.



- Suffix array is ~12 gigabytes for human – too big

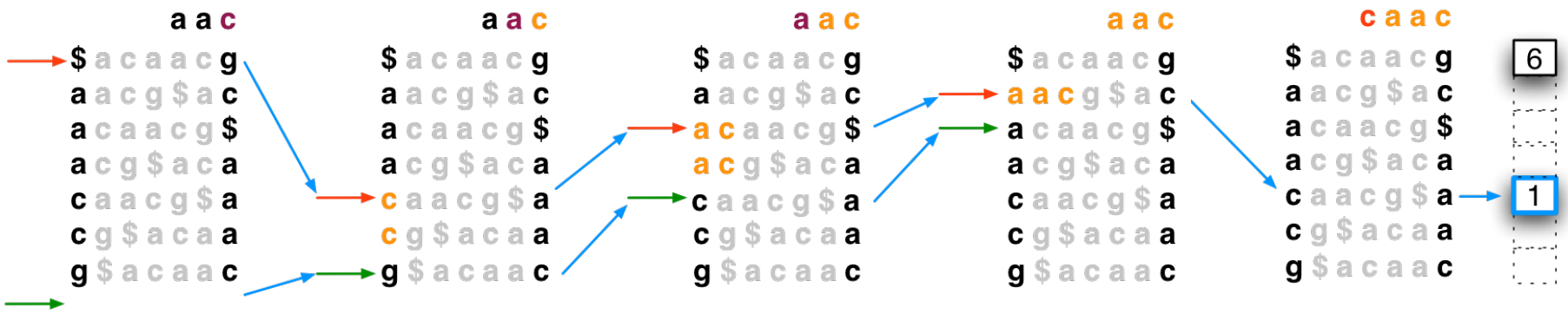
Rows to Reference Positions

- Hybrid solution: Store *sample* of suffix array; "walk left" to next sampled ("marked") row to the left
  - Due to Ferragina and Manzini



- Bowtie marks every 32nd row by default (configurable)

Put It All Together



- Algorithm concludes: “aac” occurs at offset 2 in “acaacg”

FM Index is Small

- Entire FM Index on DNA reference consists of:
  - BWT (same size as T)
  - Checkpoints (~15% size of T)
  - SA sample (~50% size of T)

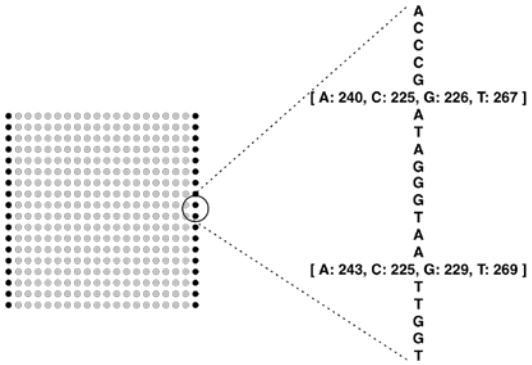
Assuming 2-bit-per-base encoding and no compression, as in Bowtie

Assuming a 16-byte checkpoint every 448 characters, as in Bowtie

Assuming Bowtie defaults for suffix-array sampling rate, etc

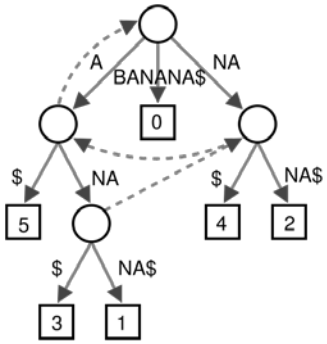
- Total: ~1.65x the size of T

Bowtie Index



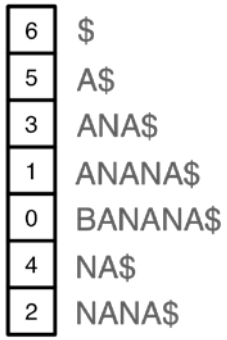
~1.65x

Suffix Tree



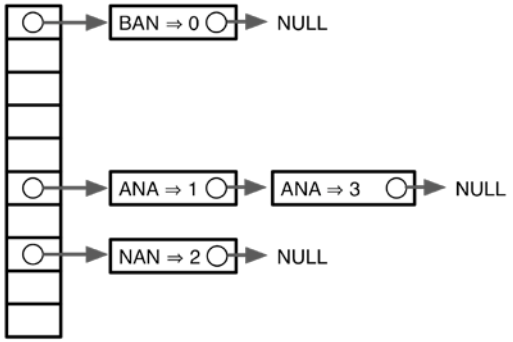
>45x

Suffix Array



>15x

k-mer Hash Tables



>15x

FM Index in Bioinformatics

- Oligomer counting
  - Healy J *et al*: Annotating large genomes with exact word matches. *Genome Res* 2003, 13(10):2306-2315.
- Whole-genome alignment
  - Lippert RA: Space-efficient whole genome comparisons with Burrows-Wheeler transforms. *J Comp Bio* 2005, 12(4):407-415.
- Smith-Waterman alignment to large reference
  - Lam TW *et al*: Compressed indexing and local alignment of DNA. *Bioinformatics* 2008, 24(6):791-797.

Short Read Alignment

- FM Index finds exact sequence matches quickly in small memory, but short read alignment demands more:
  - Allowances for mismatches
  - Consideration of quality values
- Lam *et al* try index-assisted Smith-Waterman
  - Slower than BLAST
- We tried index-assisted "seed-and-extend"
  - Competitive with other aligners, but not much faster
- Bowtie's solution: *backtracking quality-aware search*

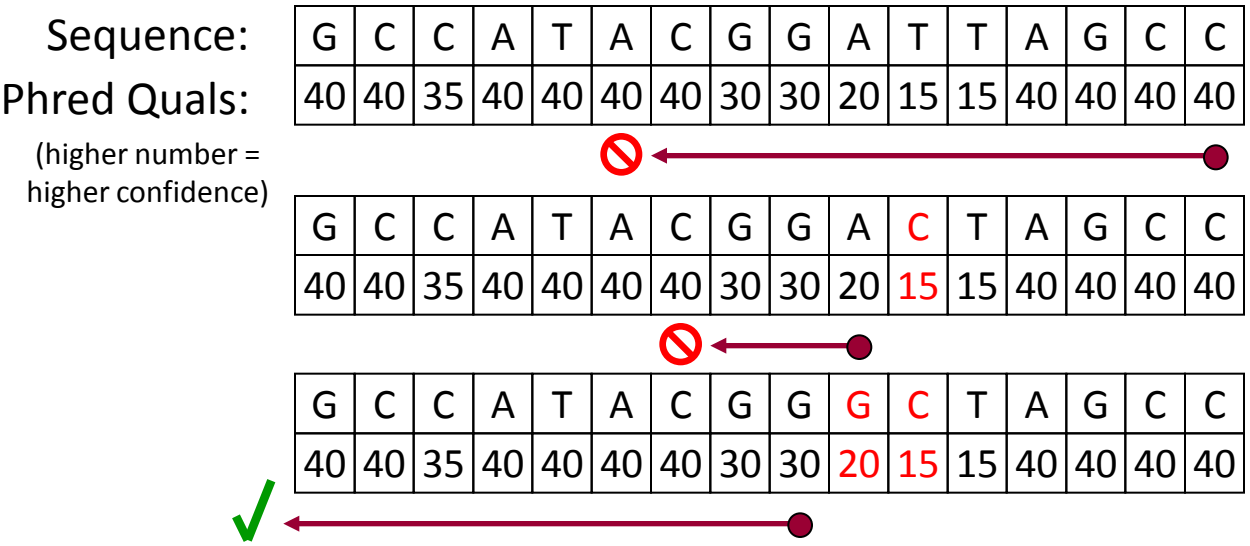
Bowtie Backtracking

- Reality is more complicated than EXACTMATCH
  - we need to account for mismatches.

- If it reaches an empty range, **backtrack** and sub a different base.
  - greedy, randomized search, first valid alignment found is reported (unless --best)

Qualities

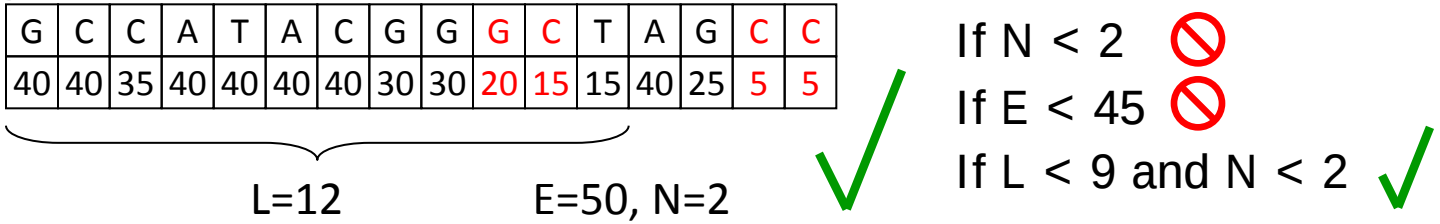
- When backtracking is necessary, Bowtie will backtrack to *leftmost just-visited position* with minimal quality



- Greedy, depth-first, not optimal, but simple

Qualities

- Bowtie supports a Maq*-like alignment policy
  - ≤ N mismatches allowed in first L bases on left end
  - Sum of mismatch qualities may not exceed E
  - N, L and E configured with **`-n`**, **`-l`**, **`-e`**
  - E.g.:



  - Maq-like is Bowtie's default mode (N=2, L=28, E=70)

* Li H, Ruan J, Durbin R: Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res* 2008.

Implementation

- Free, Open Source under Artistic License
- C++
- Uses SeqAn* library (http://www.seqan.de)
- Uses POSIX threads to exploit parallelism
- **`bowtie-build`** is the indexer
- **`bowtie`** is the aligner
- **`bowtie-convert`** converts Bowtie's alignment output format to Maq's **`.map`** format
  - Users may leverage tools in the Maq suite, e.g., **`maq assemble`**, **`maq cns2snp`**
  - Uses code from Maq

* Doring A, Weese D, Rausch T, Reinert K: SeqAn an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics* 2008, 9:11.

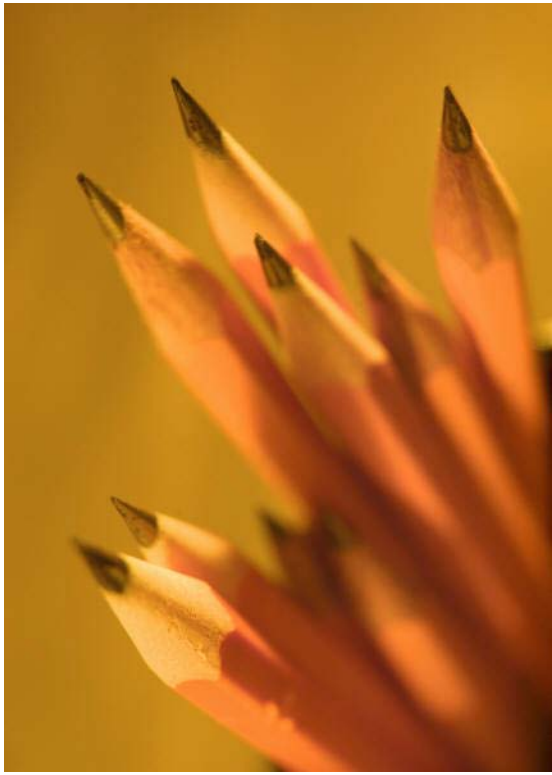
Things to watch out for

- Up to 2 mismatches in first 28 bases reported by default (Maq, Bowtie, BWA). Maq additionally reports 3-mismatches.
- Ties… If >=1 exact matches exist for a read, bowtie will report one of them. If the best match is inexact, bowtie will not always return the highest quality alignment (unless run with --best).
- Bowtie, BWA, and SOAP2 all randomly place repeat reads
- Although in theory bwa works with arbitrarily long reads, its performance is degraded on long reads (probably with any BWT aligner)
- Non-ACTG bases in the genome are converted to random bases in bwa

MAQ or Bowtie?

- BWT implementations are much faster than their hash-based counterparts
  - Several times faster still at slightly reduced sensitivity levels
  - BOWTIE's reported 30-fold speed increase over hash-based MAQ with small loss in sensitivity
  - Limitations to BWT approaches: BWA is only able to find alignments within a certain 'edit distance' (100-bp reads, BWA allows 5 'edits)
  - Important to quality clip reads (-q in BWA)
  - Non-A/C/G/T bases on reads are simply treated as mismatches
- Hash based approaches are more suitable for divergent alignment
- Rule of thumb:
  - <2% divergence -> BWT (E.g. human alignments)
  - >2% divergence -> hash based approach (E.g. mouse alignments)

BWT Exercise



- Given the string: T= tctggt

(1)Compute BWT(T)

- Given BWT(W) = TGAA$ATTGGCC

(2) Align Q = CTGG

(3) Does it align?

(4) If yes, what is the start position in W?

BWT

Original string: `ctgaaactggt`
Put a $ on the end
create cyclic rotations of the string...

c	t	g	a	a	a	c	t	g	g	t	$
t	g	a	a	a	c	t	g	g	t	$	c
g	a	a	a	c	t	g	g	t	$	c	t
a	a	a	c	t	g	g	t	$	c	t	g
a	a	c	t	g	g	t	$	c	t	g	a
a	c	t	g	g	t	$	c	t	g	a	a
c	t	g	g	t	$	c	t	g	a	a	a
t	g	g	t	$	c	t	g	a	a	a	c
g	g	t	$	c	t	g	a	a	a	c	t
g	t	$	c	t	g	a	a	a	c	t	g
t	$	c	t	g	a	a	a	c	t	g	g
$	c	t	g	a	a	a	c	t	g	g	t

BWT

Alphabetically sort the permuted strings, keeping track of which row went where
first column is the “genome dictionary”
last column is the Burrows-wheeler transformation

$	c	t	g	a	a	a	c	t	g	g	**t**	11
a	a	a	c	t	g	g	t	$	c	t	**g**	3
a	a	c	t	g	g	t	$	c	t	g	**a**	4
a	c	t	g	g	t	$	c	t	g	a	**a**	5
c	t	g	a	a	a	c	t	g	g	t	**$**	0
c	t	g	g	t	$	c	t	g	a	a	**a**	6
g	a	a	a	c	t	g	g	t	$	c	**t**	2
g	g	t	$	c	t	g	a	a	a	c	**t**	8
g	t	$	c	t	g	a	a	a	c	t	**g**	9
t	$	c	t	g	a	a	a	c	t	g	**g**	10
t	g	a	a	a	c	t	g	g	t	$	**c**	1
t	g	g	t	$	c	t	g	a	a	a	**c**	7

BWT: Exactmatch algorithm (Ferragina and Manzini): Looking for **ctgg**

Look up ctgg: start at the end with g, lookup in genome dictionary
top(g) = 6
bot(g) = 8

0	**$**	c	t	g	a	a	a	c	t	g	g	**t**
1	**a**	a	a	c	t	g	g	t	$	c	t	**g**
2	**a**	a	c	t	g	g	t	$	c	t	g	**a**
3	**a**	c	t	g	g	t	$	c	t	g	a	**a**
4	**c**	t	g	a	a	a	c	t	g	g	t	**$**
5	**c**	t	g	g	t	$	c	t	g	a	a	**a**
6	**g**	a	a	a	c	t	g	g	t	$	c	**t**
7	**g**	g	t	$	c	t	g	a	a	a	c	**t**
8	**g**	t	$	c	t	g	a	a	a	c	t	**g**
9	**t**	$	c	t	g	a	a	a	c	t	g	**g**
10	**t**	g	a	a	a	c	t	g	g	t	$	**c**
11	**t**	g	g	t	$	c	t	g	a	a	a	**c**

BWT: Exactmatch algorithm (Ferragina and Manzini): Looking for **ctgg**

Does gg exist, and what are top(gg) and bot(gg)?
Yes, gg exists.
top(gg) = top(g) + #g before g-block in bwt = 6 + 1 = 7
bot(gg) = top(gg) + # of gg in genome – 1 = 7 + 1 – 1 = 7

0	$	c	t	g	a	a	a	c	t	g	g	t
1	a	a	a	c	t	g	g	t	$	c	t	g
2	a	a	c	t	g	g	t	$	c	t	g	a
3	a	c	t	g	g	t	$	c	t	g	a	a
4	c	t	g	a	a	a	c	t	g	g	t	$
5	c	t	g	g	t	$	c	t	g	a	a	a
6	g	a	a	a	c	t	g	g	t	$	c	t
7	g	g	t	$	c	t	g	a	a	a	c	t
8	g	t	$	c	t	g	a	a	a	c	t	g
9	t	$	c	t	g	a	a	a	c	t	g	g
10	t	g	a	a	a	c	t	g	g	t	$	c
11	t	g	g	t	$	c	t	g	a	a	a	c

BWT: Exactmatch algorithm (Ferragina and Manzini): Looking for **ctgg**

Does tgg exist, and what are top(tgg) and bot(tgg)?
Yes, tgg exists.
top(tgg) = top(t) + #t before gg-block in bwt = 9 + 2 = 11
bot(gg) = top(tgg) + # of tgg in genome – 1 = 11 + 1 – 1 = 11

0	$	c	t	g	a	a	a	c	t	g	g	t
1	a	a	a	c	t	g	g	t	$	c	t	g
2	a	a	c	t	g	g	t	$	c	t	g	a
3	a	c	t	g	g	t	$	c	t	g	a	a
4	c	t	g	a	a	a	c	t	g	g	t	$
5	c	t	g	g	t	$	c	t	g	a	a	a
6	g	a	a	a	c	t	g	g	t	$	c	t
7	g	g	t	$	c	t	g	a	a	a	c	t
8	g	t	$	c	t	g	a	a	a	c	t	g
9	t	$	c	t	g	a	a	a	c	t	g	g
10	t	g	a	a	a	c	t	g	g	t	$	c
11	t	g	g	t	$	c	t	g	a	a	a	c

BWT: Exactmatch algorithm (Ferragina and Manzini): Looking for **ctgg**

Does ctgg exist, and what are top(ctgg) and bot(ctgg)?
Yes, ctgg exists.
top(ctgg) = top(c) + #c before tgg-block in bwt = 4 + 1 = 5
bot(ctgg) = top(ctgg) + # of ctgg in genome – 1 = 5 + 1 – 1 = 5

0	$	c	t	g	a	a	a	c	t	g	g	t
1	a	a	a	c	t	g	g	t	$	c	t	g
2	a	a	c	t	g	g	t	$	c	t	g	a
3	a	c	t	g	g	t	$	c	t	g	a	a
4	c	t	g	a	a	a	c	t	g	g	t	$
5	c	t	g	g	t	$	c	t	g	a	a	a
6	g	a	a	a	c	t	g	g	t	$	c	t
7	g	g	t	$	c	t	g	a	a	a	c	t
8	g	t	$	c	t	g	a	a	a	c	t	g
9	t	$	c	t	g	a	a	a	c	t	g	g
10	t	g	a	a	a	c	t	g	g	t	$	c
11	t	g	g	t	$	c	t	g	a	a	a	c

BWT: Exactmatch algorithm (Ferragina and Manzini): Looking for **ctgg**

Found ctgg at position 5, which is position 6 in original string ctgaaa**ctgg**t

0	**$**	c	t	g	a	a	a	c	t	g	g	**t**	11
1	**a**	a	a	c	t	g	g	t	$	c	t	**g**	3
2	**a**	a	c	t	g	g	t	$	c	t	g	**a**	4
3	**a**	c	t	g	g	t	$	c	t	g	a	**a**	5
4	**c**	t	g	a	a	a	c	t	g	g	t	**$**	0
5	**c**	t	g	g	t	$	c	t	g	a	a	**a**	6
6	**g**	a	a	a	c	t	g	g	t	$	c	**t**	2
7	**g**	g	t	$	c	t	g	a	a	a	c	**t**	8
8	**g**	t	$	c	t	g	a	a	a	c	t	**g**	9
9	**t**	$	c	t	g	a	a	a	c	t	g	**g**	10
10	**t**	g	a	a	a	c	t	g	g	t	$	**c**	1
11	**t**	g	g	t	$	c	t	g	a	a	a	**c**	7

Dynamic Programming

- **Global Alignment (Needleman-Wunsch)**

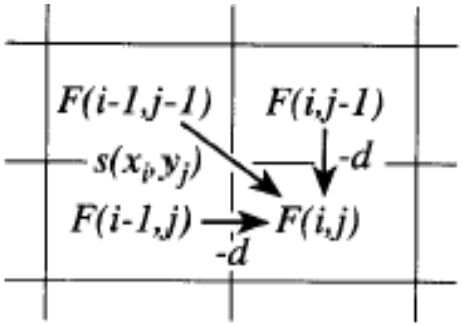
  a) General goal is to obtain optimal global alignment between two sequences, allowing gaps.

  b) We construct a matrix *F* indexed by *i* and j, one index for each sequence, where the value F(*i,*j) is the score of the best alignment between the initial segment *x1…i* of *x* up to *xi* and the initial segment *y1…j* of *y* up to *y*j. … We begin by initializing F(0,0) = 0.
  We then proceed to fill the matrix from top left to bottom right. If F(*i-1, j*-1), F(*i-1,*j) and F(*i,j*-1) are known, it is possible to calculate F(*i,*j).

Dynamic Programming

F(*i*,j) = max { F(*i-1, j*-1) + s(*xi , yj*);F(*i-1*,j) – d;F(*i, j*-1) – d. }

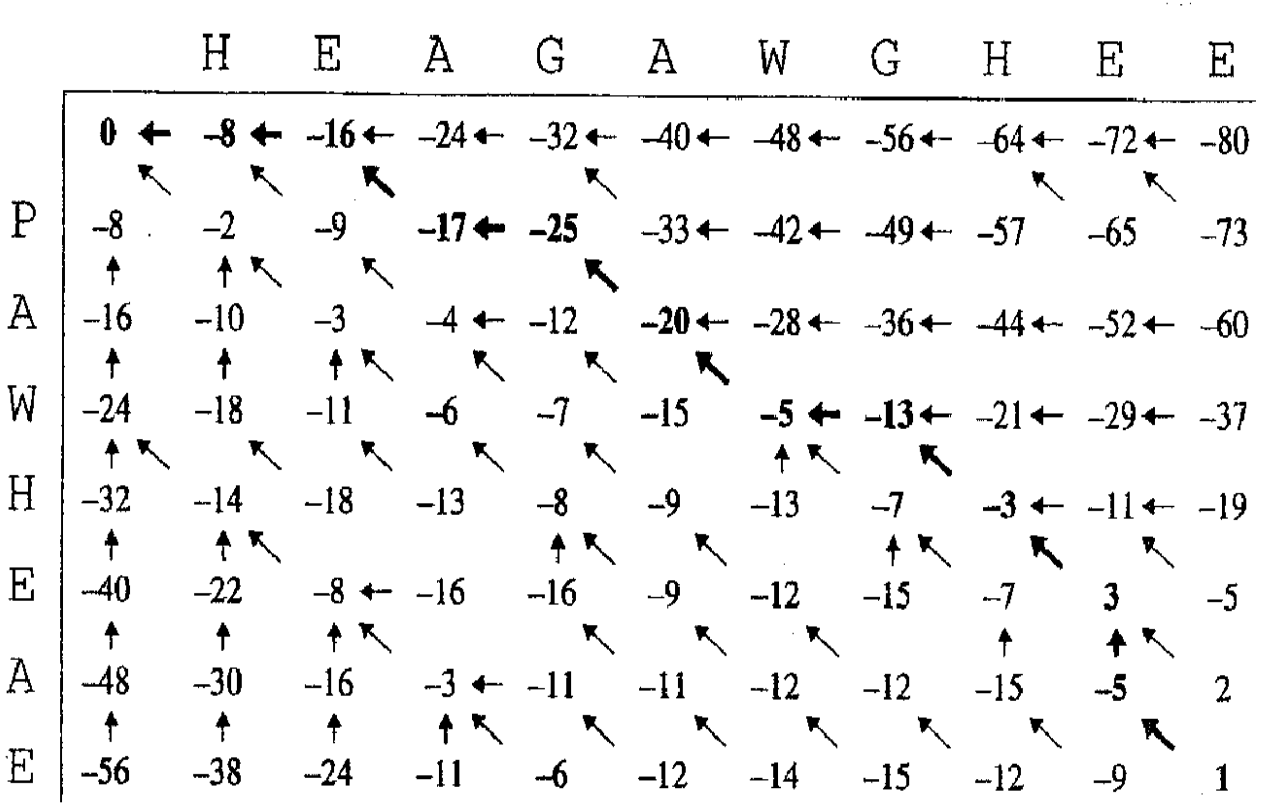
where s(*a*,b) is the likelihood score that residues *a* and *b* occur as an aligned pair, and *d* is the gap penalty.



- Once you construct the matrix, you trace back the path that leads to F(*n*,m), which is by definition the **best score for an alignment** of *x1…n* to *y1…*m.

Dynamic Programming

- Global Dynamic programming matrix



The Sequence Alignment/Map (SAM) format

- Generic alignment format
- Supports short and long reads
- Supports different sequencing platforms
- Flexible in style, compact in size, computationally efficient to access

SAM Mandatory Flelds

- 1. QNAME: Query name of the read or the read pair
- 2. FLAG: Bitwise flag (pairing, strand, mate strand, etc.)
- 3. RNAME: Reference sequence name
- 4. POS: 1-Based leftmost position of clipped alignment
- 5. MAPQ: Mapping quality (Phred-scaled)
- 6. CIGAR: Extended CIGAR string (operations: MIDNSHP)
- 7. MRNM: Mate reference name (‘=’ if same as RNAME)
- 8. MPOS: 1-based leftmost mate position
- 9. ISIZE: Inferred insert size
- 10. SEQQuery: Sequence on the same strand as the reference
- 11. QUAL: Query quality (ASCII-33=Phred base quality)

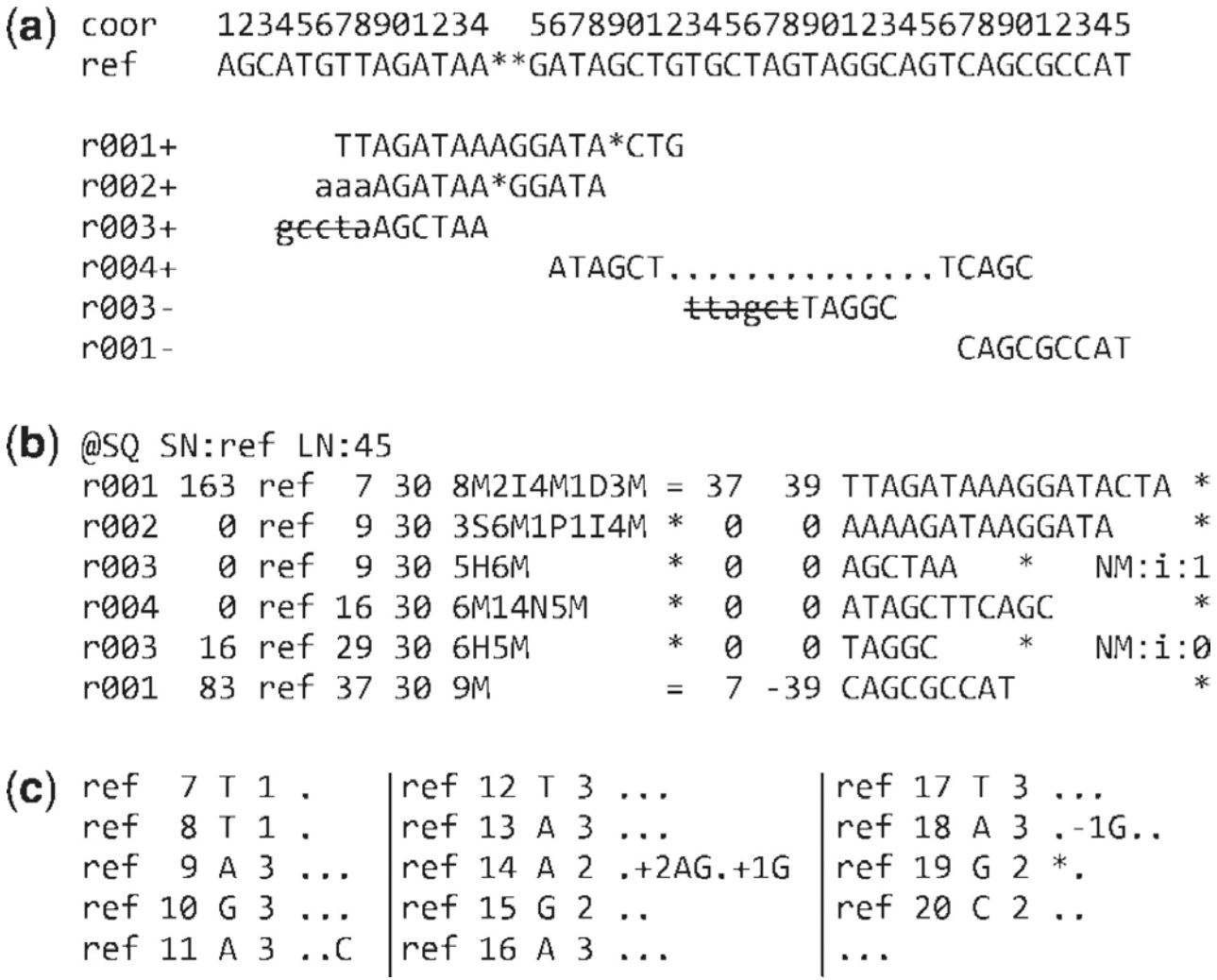
CIGAR operators

- M: match/mismatch
- I: insertion
- D: deletion
- S: softclip
- H: hardclip
- P: padding
- N: skip

```
 Ref: GCATTCAGATGCAGTACGC
Read:   ccTCAG--GCAGTAgtg

POS CIGAR
5   2S4M2D6M3S
```

Example of extended CIGAR and the pileup output.



Li H et al. Bioinformatics 2009;25:2078-2079

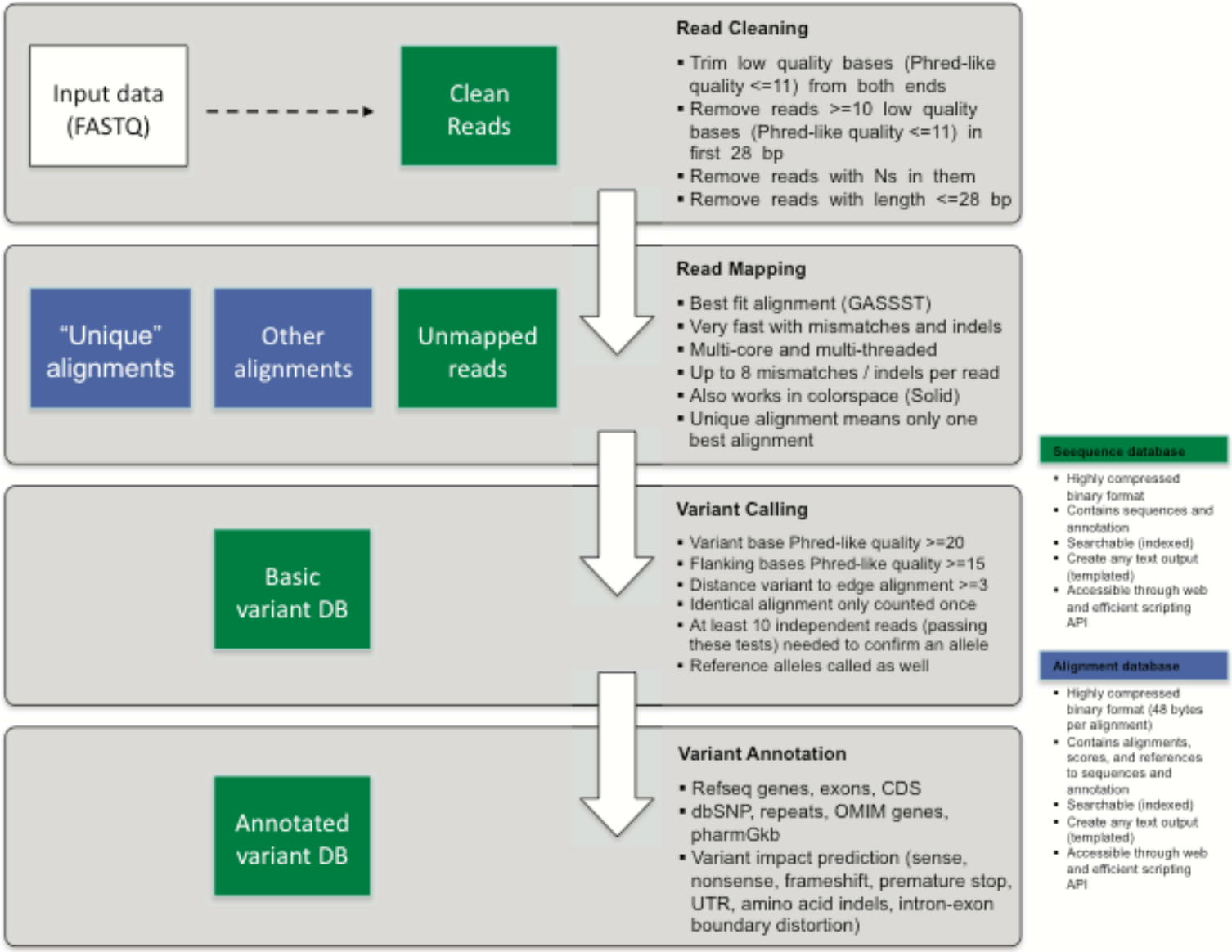
© 2009 The Author(s)

Bioinformatics

NGS Data Analysis and Computation I

- ☑ Overview
- ☑ Cloud Approaches
- ☑ Quality Control
- ☑ NGS Data Formats
- ☑ Alignment
- ☑ Variant Calling

Analysis Pipeline



Crossbow is a scalable software pipeline for whole genome resequencing analysis. It combines Bowtie, an ultrafast and memory efficient short read aligner, and SoapSNP, and an accurate genotyper. These tools are combined

Crossbow is a scalable software pipeline for whole genome resequencing analysis. It combines Bowtie, an ultrafast and memory efficient short read aligner, and SoapSNP, and an accurate genotyper. These tools are combined in an automatic, parallel pipeline that runs in the cloud (Elastic MapReduce in this case) on a local Hadoop cluster, or on a single computer, exploiting multiple computers and CPUs wherever possible. The pipeline can analyze over 35x coverage of a human genome in one day on a 10-node local cluster, or in 3 hours for about $85 using a 40-node, 320-core cluster rented from Amazon Web Services.

- Set the memory cap on the sort task to be inversely proportional to `--cpus`, to avoid memory footprint blowup on computers with more processors.
- Fixed a final issue that affected how Crossbow handles quality value conversion.
- Fixed issue whereby `bzip2`'ed data would be handled incorrectly by the preprocessor.
- Fixed counter in Preprocess step that would erroneously refer to unpaired reads as paired. Also "Read data fetched to EC2" has been changed to "Read data fetched".
- In EMR mode, updated where user credentials are found; Amazon changed their path sometime around 8/30/2010.
- In EMR mode, updated the manner in which the bootstrap action is specified; the old way was disabled by Amazon sometime around 8/30/2010.
- Fixed issue whereby `ReduceWrap.pl` would crash in cases with a large number of bins (>10 million) .
- NOTE: The Short Read Archive (SRA) seems to be in the midst of a reorganization that includes files that were previously gzipped being replaced with versions zipped with bzip2. The files will sometimes disappear for a while. If you are having problems with an example where input reads come from the SRA, try renaming files in the manifest file as appropriate. If that doesn't work, please contact us.

Version 1.0.7 - August 27, 2010

- Fixed issue whereby the order of the arguments to `bowtie` would result in a crash when `POSIXLY_CORRECT` was set.
- Fixed `--keep-all` option, which was causing a crash.
- Fixed a lingering quality bug whereby qualities were converted immediately to phred33 but phred64 or solexa64 flags would be spuriously passed to Bowtie.

Version 1.0.6 - August 26, 2010

- Single-computer mode now copies the output that it writes to the console to a file `cb.local.(pid).out`. Please include the contents of this file when reporting issues.

Tophat: RNA-Seq splice junction mapper
Cufflinks: Isoform assembly, quantitation
SoapSNP: Accurate SNP/consensus calling

Reference jars

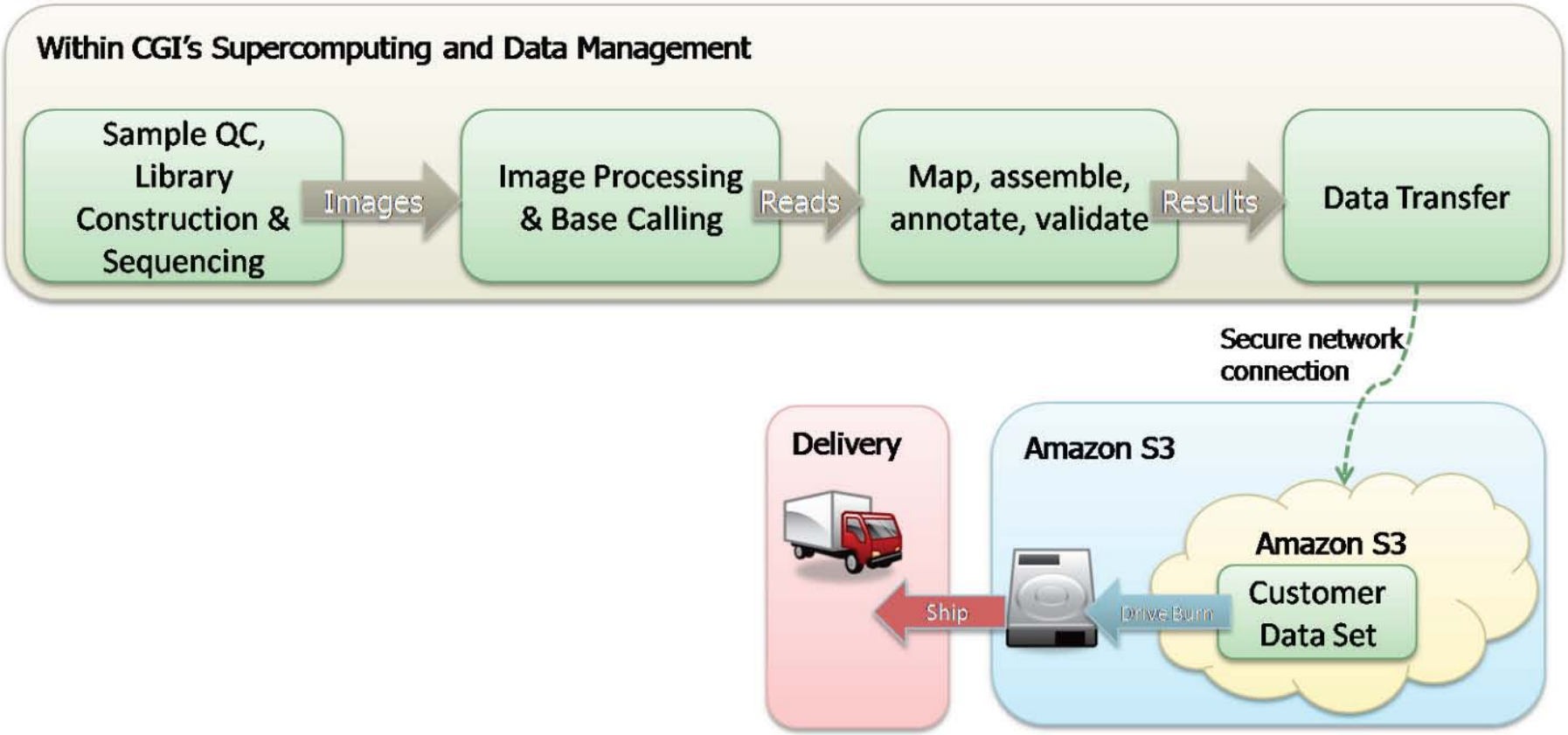
H. sapiens: hg18/dbSNP 130
s3n://crossbow-refs/hg18.jar
M. musculus: mm9/dbSNP 128
s3n://crossbow-refs/mm9.jar
E. coli: O157:H7, NCBI (no SNPs)
s3n://crossbow-refs/e_coli.jar

Related publications

Langmead B, Schatz M, Lin J, Pop M, Salzberg SL. **Searching for SNPs with cloud computing**. *Genome Biology* **10**:R134.

Schatz M, Langmead B, Salzberg SL. **Cloud computing and the DNA data race**. *Nature Biotechnology* 2010

Complete Genomics



http://media.completegenomics.com/documents/DNA-to-Data_v1-1_FINAL.pdf

More next time…

Your Assignment

- Find an NGS data set e.g. (http://www.personalgenomes.org/)
- Use Genome Quest to analyze the data
- Find 10 mutations with known side-effects
- Write a research report detailing each aspect of the analysis

Next week's help

- Map the reads from the iDEA MCF-7 Low Pass DNA Seq dataset against the Feb 2009 GRCh37/hg19 human reference genome.
- Based on the previous mapping step, run the variant calling workflow with the following changes to the default parameters and report the total number of variant alleles found and the number of variant alleles analyzed:
  - Min variant base quality = 33; Min flanking base quality = 28
  - Min variant base quality = 35; Min flanking base quality = 30
  - Min variant base quality = 30; Min flanking base quality = 25
  - Min variant base quality = 33; Min flanking base quality = 28; Minimum Number of Reads = 1; Min Number of Forward Reads = 0; Min Number of Reverse Reads = 0
- Annotate the variants from the database generated in 2a with all available tracks and report the number of variants with the following annotations:
  - Any annotation
  - Inside the boundaries of a known human gene (Refseq Genes)
  - Inside an exon of a known human gene (Refseq Exons)
  - Inside the protein coding portion of a known human gene (Refseq Proteins)
  - Overlapping with a variant in dbSNP131 (dbSNP 131)
  - Overlapping with a region in OMIM (Omim Genes)
  - Overlapping with a known diagnostic variant (NGS diagnostic panel)
  - Novel
  - Novel + exon
  - Novel + protein
- Repeat the mapping and variant calling steps using the March 2006 NCBI36/hg18 human reference genome and the variant calling parameters in 2a. Annotate the resulting variant calls with all available tracks. What type of information can be linked to these variants that is not currently readily available for variants identified from sequence reads mapped to the Feb 2009 GRCh37/hg19 human reference genome?

BMI 714 Wall

THANK YOU