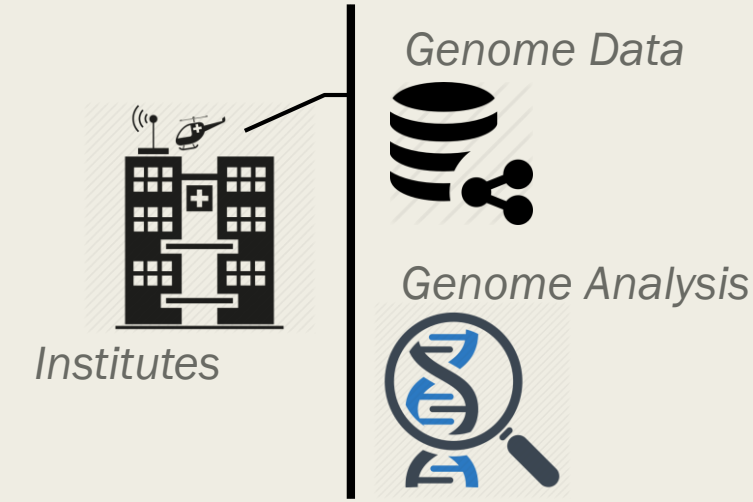
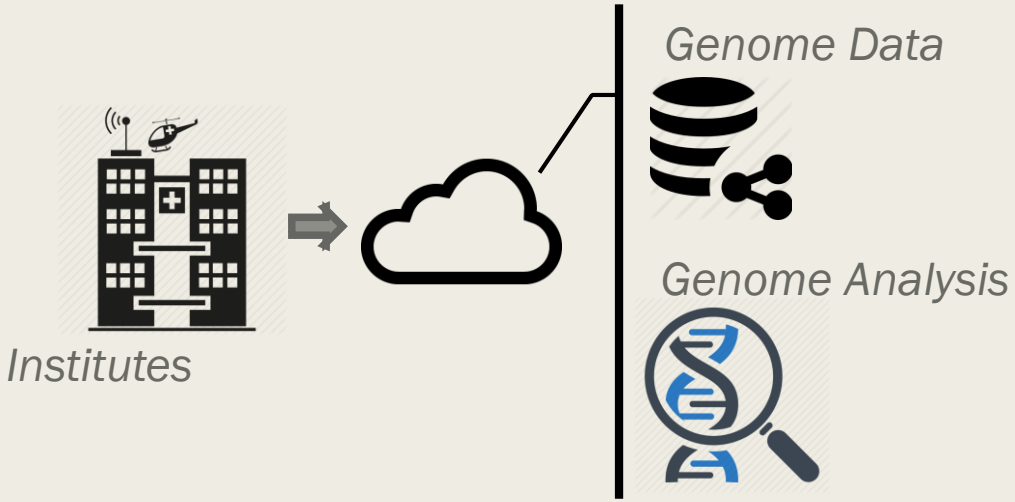
PRESAGE: PRIVACY-PRESERVING GENETIC TESTING VIA INTEL SGX

Motivation

- It is becoming a big challenge to efficiently store and process the huge amount of genomic data for individual biomedical research institutions.



Cloud Computing emerges as an ideal platform for providing elastic computation and storage resources for genomic data analysis



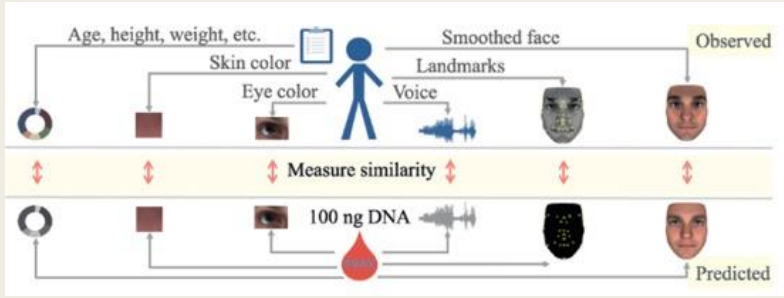
Motivation

- Individual genomic information tends to reveal sensitive personal information. Thus privacy concerns have posed challenges to outsource genomic data in an untrusted cloud environment

•***Lin et. al. 2004 Science***: 75 or more SNPs (Single-nucleotide polymorphism) will be sufficient to identify a single person.

•***Gymrek et al. 2013 Science***: surnames can be recovered from personal genomes, linking Utah Residents with Northern and Western European Ancestry (CEU) and public genetic genealogy databases (Ysearch & SMGF).

•***Lipper et.al. 2017 PNAS***: Prediction of human physical traits and demographic information.



Homer et al. 2008 PLOS Genetics: aggregated genome data (i.e., allele frequencies) can also be used for re-identifying an individual in a case group with a certain disease.

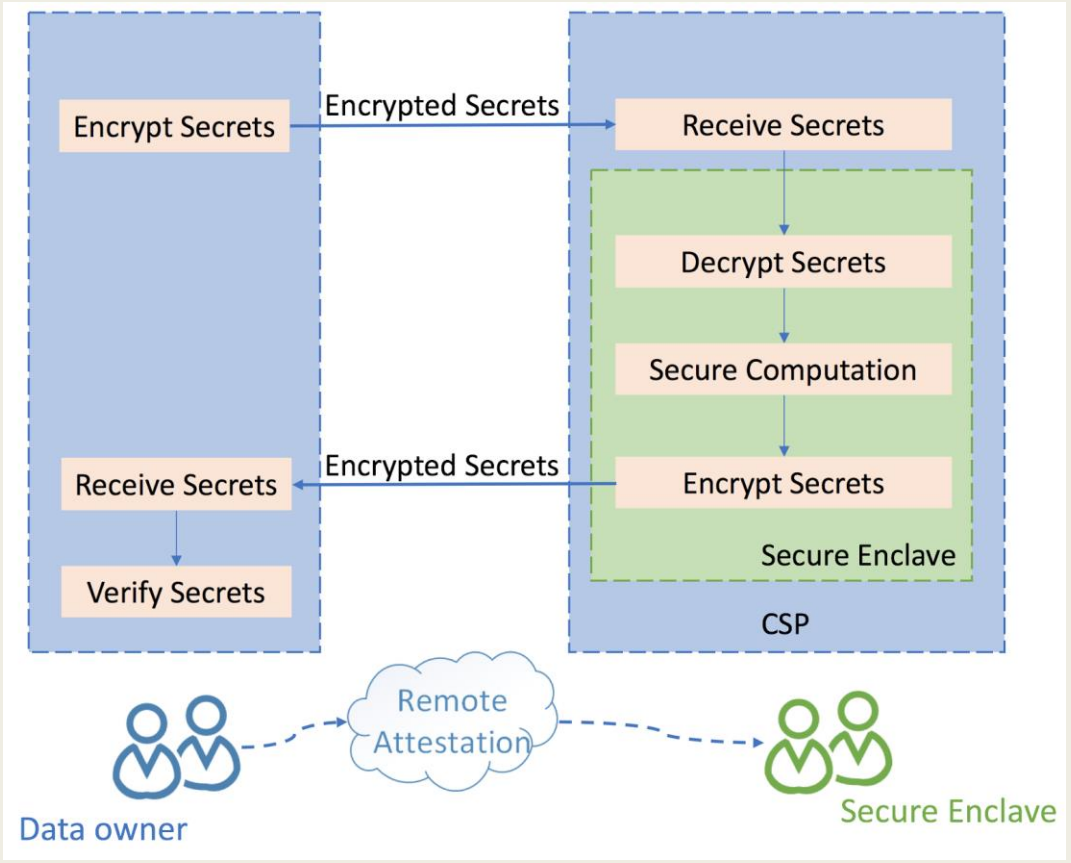
Our Solution

- We present one of the first implementations of SGX based secure genetic testing framework to facilitate efficiently outsourced storage and computation.

The secure outsource storage is achieved through data sealing scheme within SGX framework, which is immune to replay attack.

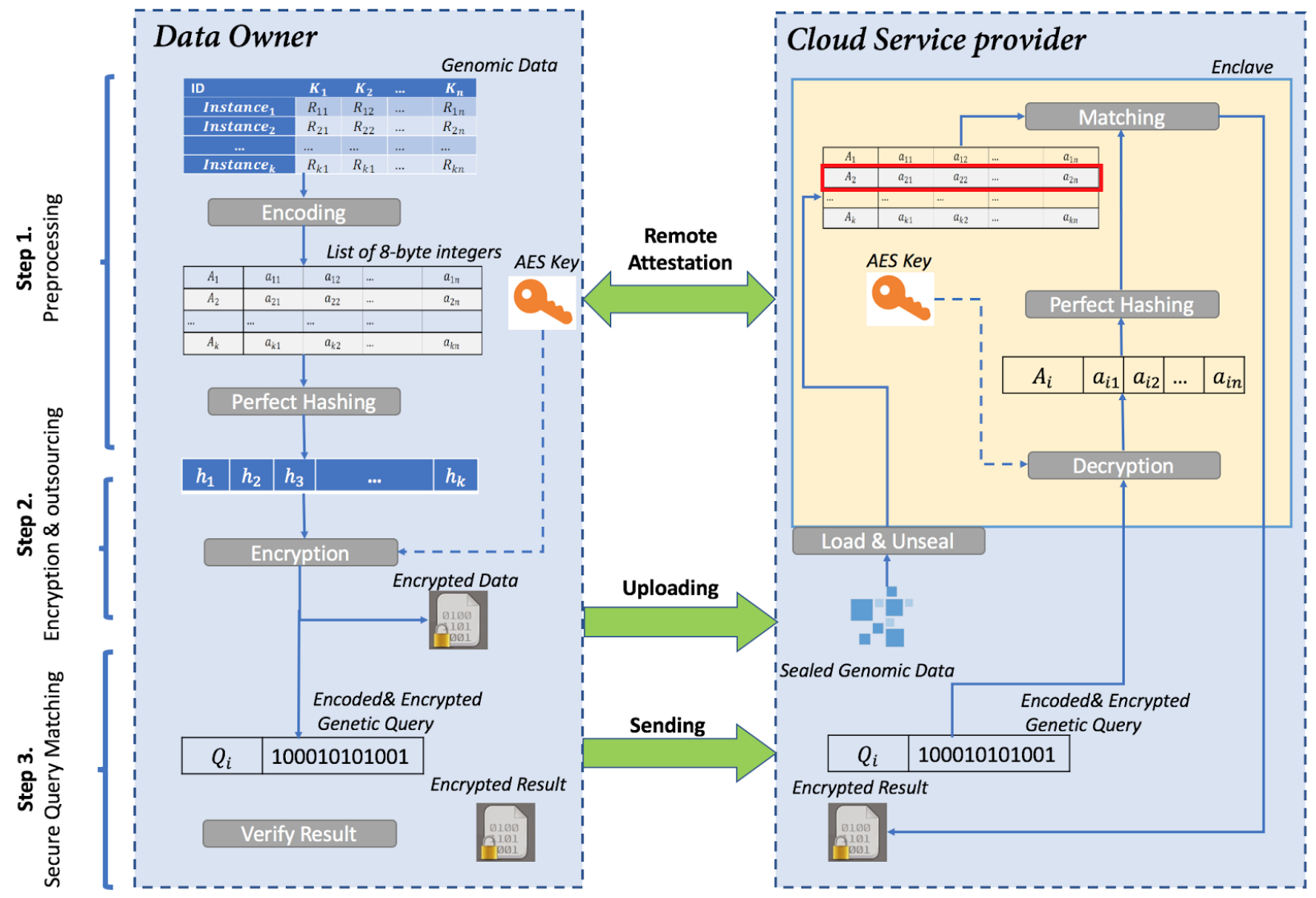
We have taken into account the oblivious access protection by using 4KB page-wise data access model.

To improve the performance, we adopt a perfect hashing scheme to achieve O(1) complexity data access within each 4KB page.



Intel SGX Based Secure Genetic Testing Cloud

Our Solution



Workflows of the proposed PRESAGE framework, presented in three consecutive steps:

1. Preprocessing
2. Encryption and data outsourcing.
3. Secure Genetic Query Matching.

Experimental Studies

- ***Dataset***: The dataset is presented in VCF format. And sizes of VCF datasets used in our experiments vary from **10,000** to **200,000** records.
- ***Experiment Environment:*** All of the experiments except the iDASH competition results are conducted on a Windows 10 SGX-enabled machine with **i7 6820HK CPU** and **48 GB** memory. Both data owner and CSP were simulated on the aforementioned SGX machine. The iDASH competition results were evaluated on the Linux server with an Xeon Processor **E3-1275 v5** and **64 GB** memory

Results

Comparison of querying performance among PRESAGE, HME-based method and plaintext implementation

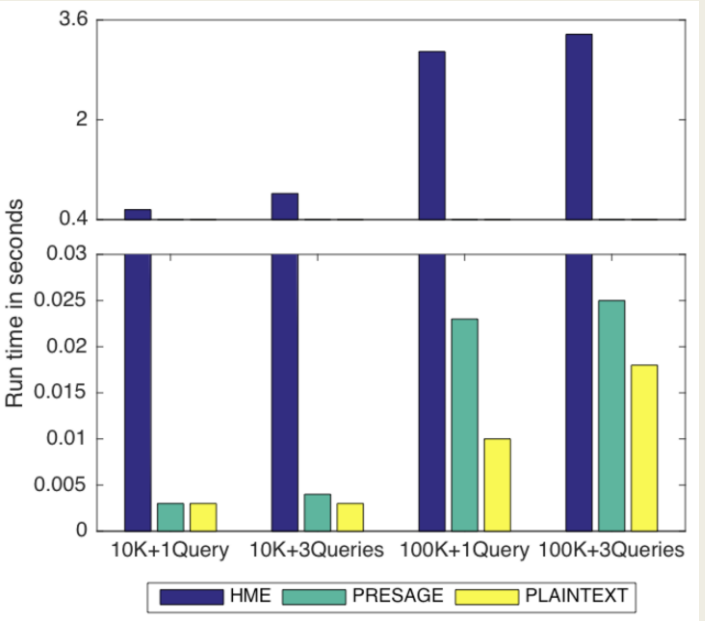


Table 1. The breakdown run time (in seconds) of the proposed PRESAGE framework

Record size	Plaintext	Encoded data	Sealed data	Enclave memory usage	
				1 query	3 queries
10,000	0.55 MB	0.09 MB	0.12 MB	3.006 MB	3.006 MB
50,000	2.59 MB	0.45 MB	0.59 MB	3.010 MB	3.010 MB
100,000	5.26 MB	0.90 MB	1.15 MB	3.010 MB	3.010 MB
200,000	10.5 MB	1.75 MB	2.31 MB	3.010 MB	3.010 MB

Table 2. The data size and enclave memory consumption (in MB) for different datasets.

Record size	Attestation	SNPs coding	Hash generation	Enclave creation	Data sealing	Number of queries	
						1	3
10,000	0.121s	0.016s	1.130s	0.169s	0.094s	0.003s	0.003s
50,000	0.126s	0.080s	6.371s	0.173s	0.517s	0.012s	0.013s
100,000	0.124s	0.164s	13.473s	0.179s	0.980s	0.023s	0.025s
200,000	0.120s	0.309s	28.677s	0.171s	2.045s	0.043s	0.048s

Conclusion

- We proposed a secure outsourcing framework, which can defend malicious attack. To improve the efficiency, an minimal perfect hashing scheme has been incorporated

- Our experiment results demonstrated the efficiency of the proposed PRESAGE framework. For a VCF file with 200K records, the PRESAGE securely processes a query within 0.05 seconds, which includes file loading, unsealing and query matching. Compared with state-of-the-art HME solution, PRESAGE framework shows at least 120x performance gain.

Acknowledgements

- Feng Chen,
- Chenghong Wang,
- Wenrui Dai,
- Xiaoqian Jiang,
- Noman Mohammed,
- Md Momin Al Aziz,
- Md Nazmus Sadat,
- Kristin Lauter,
- Shuang Wang

Secure GWAS via Intel SGX

CAN KOCKAN

Motivation / Goal

- Enable secure whole genome variant search among multiple individuals from multiple institutions
- Institution A has VCF files from x_A individuals labeled case/control, Institution B has VCF files from x_B individuals labeled case/control, …
- Institutes don't want to share data, want to do GWAS on untrusted cloud

Requirements

- Secure (Everything kept encrypted outside the SGX Enclave)
- Fast
- Accurate
- Scalable

Challenges

- Data is too large (iDASH dataset ~30GB, real-life much bigger)
- SGX Enclave max size 128 MB
- Linux allows 4GB paging – paging is extremely slow, typically many orders of magnitude slower than RAM
- More data longer transfer, more data slower encryption/decryption

Solution Outline

- Keep hash table inside the SGX enclave (server)
- Filter and compress VCF files (Client(s))
- Construct Enclave (Server)
- Perform Remote Attestation (Both Parties Exchange Messages)
- Receive Data, Update Hash Table (Server)
- Calculate Top-K (K=10 for iDASH) SNPs wrt X^2 test (Server)

Allele Counting for X^2 Test

	GG	GT	TT	Total
Cases	r_0	r_1	r_2	R
Controls	s_0	s_1	s_2	S
Total	n_0	n_1	n_2	N

Observed allele counts

	G	T	Total
Cases	$2r_0+r_1$	r_1+2r_2	$2R$
Controls	$2s_0+s_1$	s_1+2s_2	$2S$
Total	$2n_0+n_1$	n_1+2n_2	$2N$

Expected allele counts

G	T
$2R(2n_0+n_1)/(2N)$	$2R(n_1+2n_2)/(2N)$
$2S(2n_0+n_1)/(2N)$	$2S(n_1+2n_2)/(2N)$

Chi-square test for independence of rows and columns (null hypothesis):

$$\sum \frac{(\text{Obs} - \text{Exp})^2}{\text{Exp}} \sim \chi^2 \text{ with 1 df}$$

VCF (Variant Call Format)

```
##real id in 1000genome project: HG03518
#CHROM  POS     ID      REF     ALT     QUAL    FILTER  TYPE
1       13380   rs571093408     C       G       100     PASS    heterozygous
1       15211   rs78601809      T       G       100     PASS    heterozygous
1       15820   rs2691315       G       T       100     PASS    heterozygous
1       18849   rs533090414     C       G       100     PASS    heterozygous
1       30923   rs806731        G       T       100     PASS    heterozygous
1       49298   rs200943160     T       C       100     PASS    heterozygous
1       52238   rs2691277       T       G       100     PASS    heterozygous
1       55164   rs3091274       C       A       100     PASS    heterozygous
1       62777   rs528401309     A       T       100     PASS    heterozygous
1       69897   rs200676709     T       C       100     PASS    heterozygous
1       82343   rs563238524     T       C       100     PASS    heterozygous
1       83084   rs181193408     T       A       100     PASS    heterozygous
1       86331   rs115209712     A       G       100     PASS    heterozygous
```

Filtering & Variable Length Encoding

```
[203760]      2182 -> 100010000110      12
[203761]     11414 -> 10110010010110    14
[203762]      2463 -> 100110011111      12
[203763]      2083 -> 100000100011      12
[203764]      1169 -> 10010010001       11
[203765]     20377 -> 100111110011001   15
[203766]      7717 -> 1111000100101     13
[203767]     15460 -> 11110001100100    14
[203768]      5258 -> 1010010001010     13
[203769]      8842 -> 10001010001010    14
[203770]      2846 -> 101100011110      12
```

Filtering VCF and Compressed Representation

- Only SNP "ID" and "TYPE" columns essential
- "QUAL" and "FILTER" can be removed during preprocessing
- "CHROM", "POS", "REF", "ALT" can all be found via "ID" from dbSNP

- Trim the "rs" in front of "ID", represent as integer
- Sort by "TYPE", so that we don't have to keep heterozygous/homozygous
- Keep a single integer to determine when "TYPE" changes

Variable Length Encoding

- Sort "ID"s, grouped by "TYPE"
- Keep only differences using the minimum number of bits needed
- Keep another small stream for bit-lengths, encoded by a Huffman Tree

- Example VCF Filtering/Compression: Actual VCF Size: 15,428,390 Bytes
- Heterozygous-Stream: 944,166 bits Homozygous-Stream: 428,921 bits Total Main Stream Size: 1,373,087 bits
- 5-bits/len Auxiliary Stream Size: 1,631,105 bits Huffman Encoded Auxiliary Stream Size: 1,070,458 bits
- Main Stream + Huffman Auxiliary Stream: 305,444 Bytes

Preliminary Results

- 1000 case / 1000 control. ~300K-350K SNPs per VCF, ~5.5M unique SNPs
- SGX Enclave creation: 0.193381 seconds
- Remote Attestation: 0.002464 seconds
- Main Application: 49.990706 seconds
- SGX Enclave destruction: 0.034888 seconds

Acknowledgements

- Can Kockan (Indiana University)
- Natnatee Dokmai (Indiana University)
- Oguzhan Kulekci (Istanbul Technical University)
- Steve Myers (Indiana University)

- The Cancer Genome Collaboratory
- Indiana University Precision Health Initiative

Precision genomics under scalability, privacy and security constraints

S. Cenk Sahinalp

Indiana University Bloomington

Vancouver Prostate Centre

Genome storage and communication: the need

- Research: massive genome projects (e.g. PCAWG) require to exchange 10000s of genomes.
- Need to cover >200 cancer types, many subtypes.
- Within PCAWG TCGA to sequence 11K patients covering 33 cancer types. ICGC to cover 50 cancer types >15PB data at The Cancer Genome Collaboratory
- Clinic: $100s/genome (e.g. NovaSeq) enable sequencing to be a standard tool for pathology
- PCAWG estimates that 250M+ individuals will be sequenced by 2030

Current needs

- Typical technology: Illumina NovaSeq 100-400 bp reads, 250-500GB uncompressed data for high coverage human genome, high redundancy
- 40% of the human genome is repetitive (mobile genetic elements, centromeric DNA, segmental duplications, etc.)
- Upload/download: 55 hrs on a 10Mbit consumer line; 5.5 hrs on a 100Mbit high speed connection
- Technologies under development: expensive, longer reads, higher error (PacBio, Nanopore) – lower redundancy and higher error rates limit compression

File formats

- Raw read data: FASTQ/FASTA – dominant fields: (read name, sequence, quality score)
- Mapped read data: SAM/BAM – reads reordered based on mapping locus on a reference genome (not suitable for metagenomics, organisms with no reference)
- Key decisions to be made:
- Each format can be compressed through specialized methods – should there be a standardized format for compressed genomes?
- Better file formats based on mapping loci on sequence graphs representing common variants in a pan-genomic reference?

Genome Compression: Towards an International Standard

- Collaboration with MPEG to evaluate the current state of HTS data compression towards an International Standard
- Standard Benchmark DataSet: 2+ TB sequence data:
  - 7 FASTQ samples and 8 SAM samples, covering 6 species, 6 technologies, various use-cases (high and low coverage data, cancer cell lines, WGS, RNA-Seq, metagenomics etc.)
- 15 FASTQ tools and 10 SAM tools evaluated
- Available at https://github.com/sfu-compbio/compression-benchmark

Comparison of high-throughput sequencing data compression tools
[Numanagić et al., *Nat. Meth.*, Dec 2016]

FASTA/Q compression

General purpose compressors used in genomics

- LZ77 tools (gzip, pigz)
- BWT tools (bzip2, pbzip2)
- LZMA (7z)
- Context mixing (zpaq, lpaq)
- NCBI Toolkit (used at SRA for storing samples)

- General compressors do not take into account redundancies specific to FASTQ format (FASTQ files are treated as ordinary text files)

Compressor (on 53.8 GB human g. 6.5x coverage)	**Size (total)**	**Size (field by field)**	**Size (sequence)**
pigz	18.5 GB	16.1 GB	5.9 GB
pbzip2	14.8 GB	14.1 GB	5.4 GB
NCBI SRA	~ 14.2 GB		

Specialized FASTA/Q compressors

- Goals:
  - Read name tokenization
  - Separate sequence and quality score modeling

- Examples:
  - DSRC and DSRC2 [1] (Huffman coding)
  - fastqz, fqzcomp [2] and Slimfastq [3] (context mixing with arithmetic coding)
  - FQC [4] and LFQC [5] (LZMA, paq and ppmd as compression engine)

Compressor (on 53.8 GB)	**Size (total)**	**Size (sequence)**
DSRC2	13.2 GB	5.2 GB
Slimfastq	11.0 GB	4.4 GB
FQC	11.4 GB	N/A

[1] Roguski S, Deorowicz S. **DSRC 2--Industry-oriented compression of FASTQ files.** Bioinformatics, 2014
[2] Bonfield JK, Mahoney MV. **Compression of FASTQ and SAM Format Sequencing Data.** PLoS ONE, 2013
[3] Ezra J. https://github.com/Infinidat/slimfastq
[4] Dutta A, Haque MM, Bose T, Reddy CV, Mande SS. **FQC: A novel approach for efficient compression, archival, and dissemination of FASTQ datasets.** J Bioinform Comput Biol., 2015
[5] Nicolae M, Pathak S, Rajasekaran S. **LFQC: a lossless compression algorithm for FASTQ files.** Bioinformatics, 2015

FASTA/Q compressors based on read reordering

- Goals:
  - Reorder reads to improve locality of reference

Compressor (on 53.8 GB)	Size (total)	Size (sequence)
SCALCE	10.8 GB	3.0 GB
ORCOM	N/A	1.7 GB
Mince	N/A	6.0 GB
LW-FQZip	N/A	

- Examples:
  - SCALCE [1] (uses locally consistent parsing for read reordering/clustering)
  - ORCOM [2] (uses lexicographically smallest k-mers for clustering)
  - Mince [3] (similar to ORCOM)
  - LW-FQZip [4] (uses implicit mapping to a reference)

[1] Hach F, Numanagić I, Alkan C, Sahinalp SC. **SCALCE: boosting sequence compression algorithms using locally consistent encoding.** Bioinformatics, 2012
[2] Grabowski S, Deorowicz S, Roguski L. **Disk-based compression of data from genome sequencing.** Bioinformatics, 2014
[3] Patro R, Kingsford C. **Data-dependent Bucketing Improves Reference-free Compression of Sequencing Reads.** Bioinformatics, 2015
[4] Zhang Y, Li L, Yang Y, Yang X, He S, Zhu Z. **Light-weight reference-based compression of FASTQ data.** BMC Bioinformatics, 2015

FASTA/Q compressors based on read assembly

- Goals:
  - Assemble the underlying genome and map reads to the assembly

- Examples:
  - Quip [1] (Bloom filters, assembles clusters of 1 million reads)
  - Leon [2] (probabilistic de Bruijn graph)
  - k-Path [3] (probabilistic de Bruijn graph)

Compressor (on 53.8 GB)	**Size (total)**	**Size (sequence)**
Quip	11.3 GB	4.5 GB
Leon	13.6 GB	4.7 GB
k-Path	N/A	2.0 GB

[1] Jones DC, Ruzzo WL, Peng X, Katze MG. **Compression of next-generation sequencing reads aided by highly efficient de novo assembly**. Nucleic Acids Res. 2012
[2] Benoit G, Lemaitre C, Lavenier D, Drezen E, Dayris T, Uricaru R, Rizk G.. **Reference-free compression of high throughput sequencing data with a probabilistic de Bruijn graph.** BMC Bioinformatics, 2105.
[3] Kingsford C, Patro K. **Reference-based compression of short-read sequences using path encoding**. Bioinformatics, 2015

Compression results on raw (FASTA/Q) read data

Sample	SRR554369		SRR327342		MH0001.081026		SRR1284073		SRR870667		ERR174310		ERR174324	
Organism	*P.aeruginosa*		*S.cerevisiae*		*H.sapiens Gut*		*E.coli*		*T.cacao*		*H.sapiens*		*H.sapiens*	
Technology	Illumina GAIIx		Illumina GAII		Illumina GA		PacBio		Illumina GAIIx		HiSeq		HiSeq	
Coverage	105x		Unknown		Unknown		5x		65x		25x		335x	
Original	**550** *165*		**3,881** *947*		**1,880** *512*		**1,309** *649*		**22,944** *7,463*		**53,869** *20,966*		**2,717,029** *1,059,387*	
pigz	**158** *48*	1.00 1.00	**1,020** *277*	1.00 1.00	**501** *149*	1.00 1.00	**547** *188*	1.00 1.00	**6,943** *2,108*	1.00 1.00	**18,597** *5,982*	1.00 1.00	**305,690** *104,927*	1.00 1.00
pbzip2	**125** *44*	1.19 6.12	**831** *251*	1.41 6.80	**390** *139*	1.33 6.10	**463** *176*	0.74 6.99	**5,577** *1,879*	1.07 3.59	**14,887** *5,473*	0.80 3.08	**242,834** *95,969*	0.21 1.23
DSRC2	**105** *41*	0.21 2.15	**668** *257*	0.26 3.22	**312** *128*	0.25 2.06	N/A		**4,761** *1,865*	0.23 1.45	**13,214** *5,239*	0.20 1.25	N/A	
Fqzcomp	**89** *37*	0.35 N/A	**559** *203*	0.37 7.39	**280** *120*	0.41 N/A	N/A		**4,028** *1,556*	0.34 N/A	**11,320** *4,623*	0.31 3.38	N/A	
Fastqz	N/A		N/A		N/A		N/A		N/A		**10,955** N/A	3.45 N/A	N/A	
Slimfastq	**94** *30*	0.54 11.62	**507** *149*	0.48 9.93	**266** *104*	0.54 10.94	N/A		**4,280** *1,416*	0.52 5.82	**11,045** *4,426*	0.47 4.89	**178,092** *77,629*	0.49 5.94
FQC	**76** N/A	1.04 12.16	**494** N/A	1.23 13.42	**268** N/A	1.51 18.66	**413** N/A	0.98 12.12	**3,912** N/A	1.20 6.34	**11,409** N/A	1.22 5.87	N/A	
LFQC	**69** *17*	9.24 159.86	**490** *129*	8.67 146.15	**266** *103*	10.44 162.94	**407** *156*	18.03 386.25	**2,412** N/A	8.53 N/A	N/A		N/A	
SCALCE	**76** *17*	0.38 4.59	**487** *68*	0.29 3.87	**297** *71*	0.40 5.83	**421** *161*	0.67 9.78	**3,699** *998*	0.35 2.88	**10,827** *3,017*	0.30 2.36	**161,067** *28,452*	0.57 1.94
LW-FQZip	**117** *45*	1.10 5.60	**790** *320*	0.60 5.25	N/A		N/A		**5,038** *1,735*	2.16 2.56	N/A		N/A	
Quip	**89** *37*	0.39 7.58	**537** *181*	0.48 8.51	**272** *114*	0.49 11.00	**420** *159*	0.36 10.59	**3,914** *1,462*	0.50 5.74	**11,312** *4,556*	0.46 5.39	**184,051** *79,771*	0.38 4.64
Leon	**87** *19*	3.61 16.95	**544** *89*	2.66 17.02	**291** *87*	3.66 14.22	**479** *170*	2.81 34.31	**4,518** *1,360*	3.93 10.49	**13,623** *4,739*	3.55 9.90	**220,397** *83,539*	1.13 4.66
KIC	**95** *32*	5.75 6.89	**613** *188*	7.40 7.88	**307** *122*	5.32 7.38	**451** *168*	9.40 9.37	**4,498** *1,594*	6.74 3.60	**13,006** *4,915*	6.19 3.30	N/A	
Orcom	*11*	0.25 0.77	*36*	0.22 0.43	*51*	0.43 0.90	N/A		*825*	0.49 0.72	*1,798*	0.33 0.43	*6,921*	0.12 0.23
BEETL	*23*	4.09 37.52	*117*	3.14 32.22	*114*	2.46 29.83	N/A		*1,200*	3.97 20.68	*3,912*	4.39 21.76	N/A	
k-Path	*14*	1.01 15.25	*45*	0.81 9.49	*62*	6.47 71.69	N/A		*660*	1.35 9.05	*2,088*	1.62 8.55	N/A	

SAM/BAM compression

General purpose compressors for SAM files

- LZ77 tools (gzip, pigz)
- BWT tools (bzip2, pbzip2)
- Current standard: LZ77-based BAM (Samtools [1], Sambamba [2], Picard [3])

- None of those methods treat differently separate SAM columns.
- Clearly, simple stream separation without any additional post-processing increases significantly the overall compression rate

[1] Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, 1000 Genome Project Data Processing Subgroup. **The Sequence Alignment/Map format and SAMtools.** Bioinformatics, 2009
[2] Tarasov A, Vilella AJ, Cuppen E, Nijman IJ, Prins P. **Sambamba: fast processing of NGS alignment formats.** Bioinformatics, 2015
[3] Broad Institute. http://broadinstitute.github.io/picard/

Compressor (on human cancer g. sample 427 GB)	**Size**	**Size (separate streams)**
pigz	119 GB	103 GB
pbzip2	100 GB	94 GB
Samtools	131 GB	102 GB

Specialized SAM tools

- Separate fields into different compression streams
- Use reference to store sequence information, if possible

Compressor (on human cancer sample; 427 GB)	**Size**
Cramtools	95 GB
Scramble	82 GB
Scramble (without reference)	86 GB
Quip (without reference)	98 GB
sam_comp * does not support all SAM fields	42 GB*

- Primarily reference based:
  - CRAM format (Scramble [1], Cramtools [2])
- Assembly and reference based:
  - Quip [3]
- Statistical modeling and arithmetic encoding:
  - sam_comp [4]

[1] Bonfield JK.**The Scramble conversion tool.** Bioinformatics, 2014
[2] Hsi-Yang Fritz M, Leinonen R, Cochrane G, Birney E. **Efficient storage of high throughput DNA sequencing data using reference-based compression.** Genome Res., 2011
[3] Jones DC, Ruzzo WL, Peng X, Katze MG. **Compression of next-generation sequencing reads aided by highly efficient de novo assembly**. Nucleic Acids Res. 2012
[4] Bonfield JK, Mahoney MV. **Compression of FASTQ and SAM Format Sequencing Data.** PLoS ONE, 2013

Local assembly based SAM tools

- Avoid redundant storing of SNPs and other small SVs
- Find the underlying genome via local assembly, and encode SNPs and small SVs only once

- Examples:
  - DeeZ [1]
  - CBC [2]

Compressor (on human cancer sample; 427 GB)	**Size**
DeeZ	78 GB

Sequence only without assembly	**Sequence only with assembly**
4,169 MB	4,120 MB

[1] Hach F, Numanagić I, Sahinalp SC. **DeeZ: reference-based compression by local assembly.** Nat. Methods, 2014
[2] Ochoa I, Hernaez M, Weissman T. **Aligned genomic data compression via improved modeling.** Journal of bioinformatics and computational biology, 2014

DeeZ: DeeNA Zeep

Motivation

- BAM (the most common format for storage and communication) misses some opportunities in SAM format, particularly common SNV loci in reads

- Alternative SAM/BAM compression tools, based on arithmetic coding (AC) and other data modeling methods, like Quip and Samcomp, provide superior compression, but not random-access capability

- DeeZ locally assembles reads and represents each SNV once, on the contig.

DeeZ: Quality scores

- Quality scores account for majority of the space in almost any format
  - minor improvement in quality score compression is more beneficial than improvement in other areas

DeeZ on human cancer sample **427 GB**	**Size**	**Gain**
Sequence only without local assembly (5% of compressed file)	4,169 MB	
Sequence only with local assembly (5% of compressed file)	4,120 MB	49 MB
Quality scores only with order-1 AC model (42% of compressed file)	33,516 MB	
Quality scores only with sam_comp model (41% of compressed file)	31,010 MB	2,506 MB

Compression results on mapped (SAM/BAM) read data

Sample	DH10B		9827.2.49		sample-2-1		K562.LID8465		HCC1954		NA12878.S1	
Organism	*E.coli*		*H.sapiens*		*H.sapiens*		*H.sapiens*		*H.sapiens*		*H.sapiens*	
Technology	MiSeq		HiSeq		IonTorrent		RNASeq		Cancer Cell		HiSeq	
Coverage	490x		2x		0.7x		7x		35x		60x	
Original	5,579		21,059		5,924		75,915		427,028		589,083	
pigz	1,336	0.77	6,021	1.55	1,378	1.48	12,785	1.06	119,839	1.40	113,462	0.13
		0.63		0.82		0.49		0.70		0.91		0.60
pbzip2	1,074	1.65	5,243	1.93	1,127	4.04	10,251	3.57	100,280	1.62	89,598	0.46
		3.16		3.39		3.72		2.46		3.23		0.59
Samtools	1,407	1.00	6,499	1.00	1,469	1.00	13,757	1.00	131,566	1.00	121,710	1.00
		1.00		1.00		1.00		1.00		1.00		1.00
Picard	1,425	1.42	6,517	1.04	1,474	1.82	13,818	1.48	132,861	1.18	N/A	
		2.76		1.52		2.10		2.44		1.91		
Sambamba	1,407	1.05	6,499	0.93	1,469	1.12	13,757	1.05	131,566	1.39	121,710	0.13
		1.08		1.13		0.97		0.97		1.12		0.53
Cramtools	1,066	0.93	3,778	1.42	1,170	2.12	10,344	1.70	95,442	1.28	N/A	
		1.71		1.67		4.93		2.00		1.50		
Scramble	863	0.23	3,297	0.29	1,030	0.62	9,261	0.38	82,041	0.27	66,632	0.10
		0.76		0.66		1.58		0.67		0.71		0.50
Scramble without reference	899	0.29	4,236	1.18	1,113	0.45	9,839	0.43	86,914	0.37	72,407	0.10
		0.74		0.63		1.06		0.78		0.79		0.47
Scramble with bzip2	851	0.76	3,262	0.62	998	1.50	8,611	1.27	80,094	0.60	N/A	
		0.89		0.66		1.72		0.81		0.82		
DeeZ	823	0.56	3,221	0.78	1,028	1.81	8,120	0.92	78,473	0.91	62,966	0.26
		3.90		2.46		5.51		3.35		2.94		1.00
DeeZ with bzip2	730	0.91	2,734	1.23	918	3.49	7,266	2.01	74,509	1.66	53,497	0.41
		10.11		5.60		9.86		7.91		6.39		1.90
TSC	1,105	2.21	7,939	0.80	1,193	2.55	20,864	3.17	164,627	0.50	N/A	
		9.05		2.24		6.75		6.27		2.65		
Quip	1,103	0.67	4,419	0.94	1,230	0.96	11,186	1.19	98,303	0.83	97,165	0.44
		10.69		7.81		3.37		8.27		9.05		2.18
Quip with reference	803	0.67	N/A		N/A		8,743	1.17	N/A		64,493	0.43
		10.06						8.20				2.20
sam_comp	700	0.68	2,649	0.76	891	1.20	7,023	0.71	42,522	0.62	53,263	0.37
		3.36		2.95		6.54		3.56		3.25		2.00

Optimal Compressed Representation of High Throughput Sequence Data via Light Assembly

Cenk Sahinalp

Based on joint work with

Kaiyuan Zhu, Tony Ginart, Joseph Hui, Ibrahim Numanagić,

Thomas Courtade, David Tse

Current FASTQ Compression Schemes

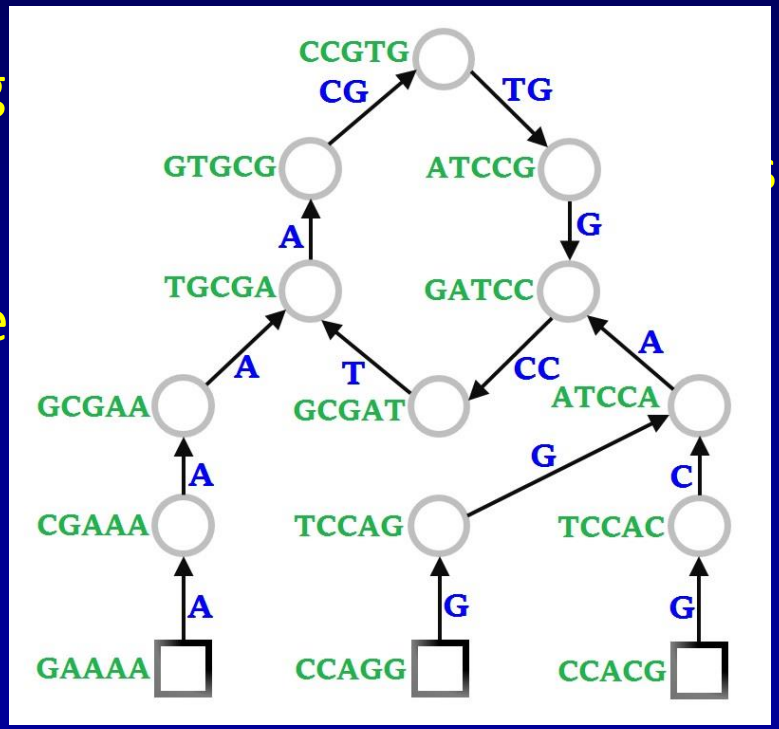
- General purpose compressors (FASTQ files are treated as ordinary text files)
  - gzip (parallel gzip--pigz), bzip2 (parallel bzip2--pbzip2)
- Alignment e underlying reference g
  - use de-n raph
    - Quip,
  - use an us ng
    - LW-FC
- Reordering reordering of the reads ly boost the compression rates while avoiding information loss.
  - SCALCE, Orcom, Mince

Compressor (on 53.8 GB human with 6.5x coverage)	Size (total)	Size (field by field)	Size (sequence)
pigz	18.5 GB	16.1 GB	5.9 GB (2.251)
pbzip2	14.8 GB	14.1 GB	5.4 GB (2.060)
SRA	~ 14.2 GB		

Assembltrie: Our New Compressed Representation

- Combine the advantages of reordering and alignment based compressors.
- The input reads are organized into a *forest* of compact trie-like data structures called **read forest**.
  - Each node v represents a read (a string
  - Each directed edge (u, v) represents
    covered by its parent, v
  - May contain a single cycle acting as the

not



An example trie-like structure

Combinatorial Optimization Formulation

- Among all possible read forests, our objective is to find the one contains minimum number of symbols, i.e.

$$T^* = \arg\min_{T} \sum_{\tau \in T} \sum_{v \in V_\tau} w[v, \pi(v)]$$

τ: a trie in the forest T, with the corresponding vertex set V_τ;
$w[v,u]$: the length of the shortest suffix of v that can not be covered by a suffix of u;
$\pi(v)$: the parent node of v in T, $w(v,X)$, start at v

- The greedy algorithm to build the desired read forest
  - Pick for each read u an already processed read v with minimum $w[v,u]$, set its parent $\pi(u)$ to v
  - Identify each already processed read v with $w[v,u] < w(u,X)$ to trie with only u (can assume $\pi(u)$ = NIL)

<u>Theorem</u>: The greedy algorithm computes the optimal T^* with minimum overlap K.

Information Theoretic Upper Bound for HTS Data Compression

- Assembltrie achieves combinatorial optimality
  - For any finite collection $\mathcal{R}$ of reads to be compressed with any explicit or implicit (overlap graph) assembly based compressor, it produces the smallest number of symbols to be encoded for reads.
- Is it possible to obtain better compression performance by a fundamentally different data structure (i.e. representation of reads)?
  - NO. The minimum number of bits needed by any algorithm to describe the reads $\mathcal{R}$ is given by $H(\mathcal{R})$.
  - Assembltrie approaches this bound on error-free
- ≠ Optimal compression in practice
  - The proof does not consider read errors
  - Need to account for read, sampling etc.

$$H(\mathcal{R}) \approx NL\log(3)h_2(p) + |G| \cdot H\left(\text{Poisson}\left(N/|G|\right)\right) + LZ(G)$$

Sequencing errors | Read sampling process | Reference genome

$$h_2(p) = -p\log p\ -(1-p)\log 1-p$$

N: number of reads; *L*: read length; *G*: reference genome of length $|G|$

Compression Performance (8 Threads, in bit per base)

MPEG HTS FASTQ Dataset

Sampled S. cerevisiae

Sample	Read L. / Cov.	Assembltrie	Orcom	Mince	K-Path	SCALCE
P.aeruginosa	100 / 25	0.345	0.518	0.484	0.673	0.821
S.cerevisiae	63 / 80	0.271	0.304	0.312	0.384	0.578
H.sapiens gut	44 / NA	0.757	0.804	0.786	2.545	1.104
T.cacao	108 / 20	1.733	0.884	0.735	0.707	1.070
Sim. *T.cacao*	108 / 19	0.479	0.667	N/A	N/A	N/A
H.sapiens 1	101 / 7	0.570	0.686	0.746	0.797	1.151
H.sapiens 2	101 / 20	0.322	0.364	N/A	N/A	N/A

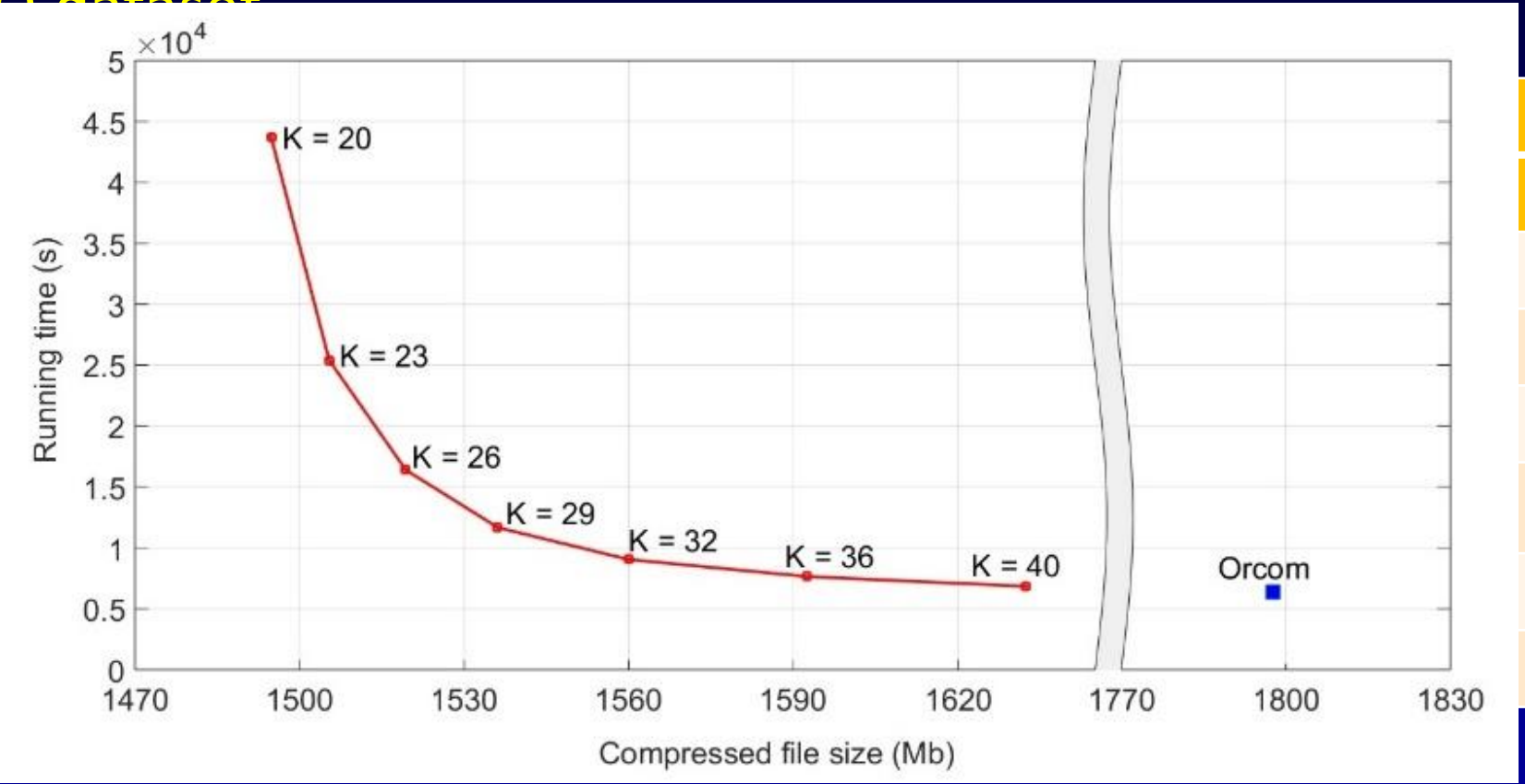
Coverage

Ref: Numanagić, I., Bonfield, J. K., Hach, F., Voges, J., Ostermann, J., Alberti, C., ... & Sahinalp, S. C. (2016). Comparison of high-throughput sequencing data compression tools. *Nature methods*.

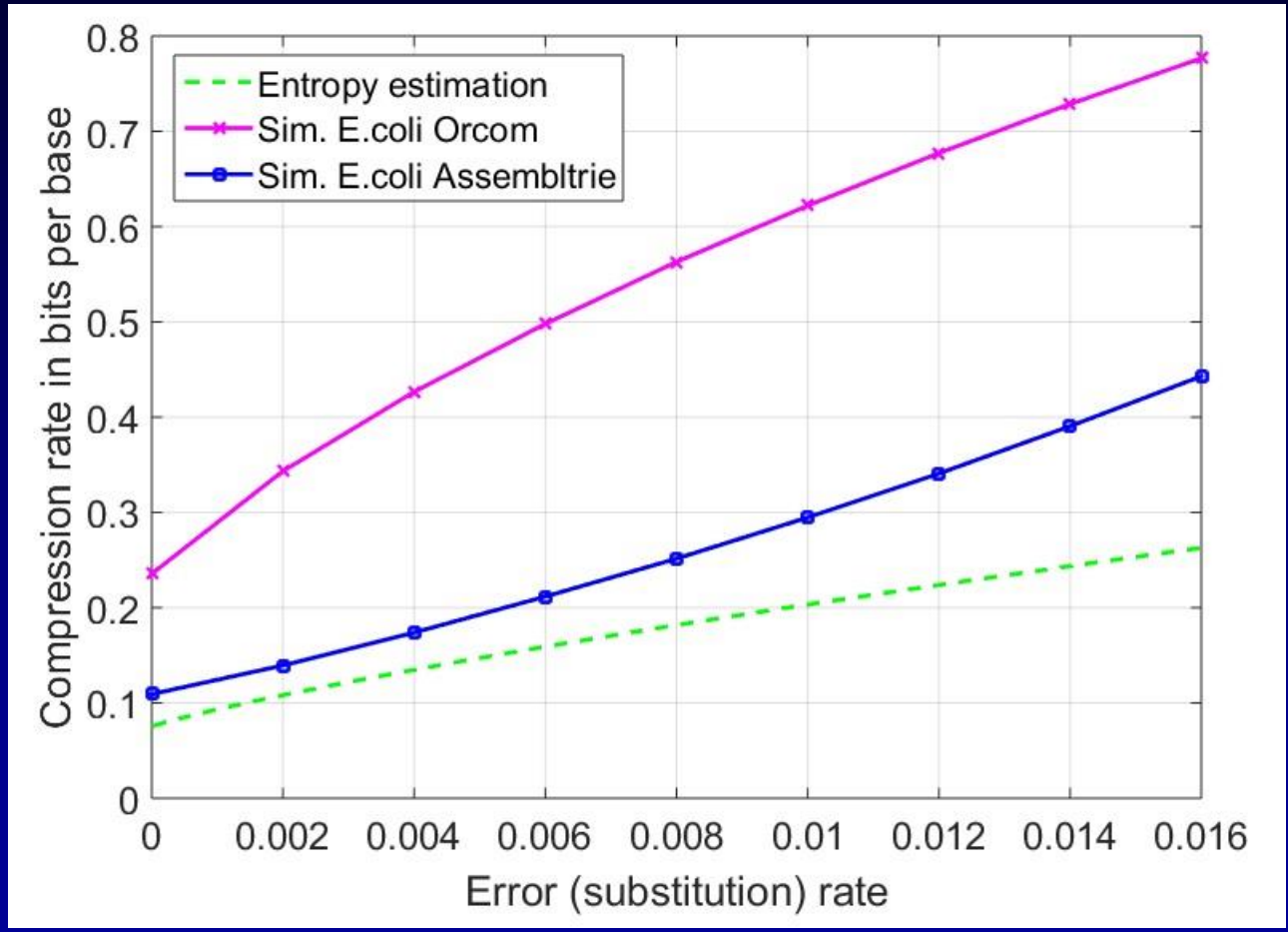
Running Time (8 Threads, in seconds)

- Default running time to generate the compression rates in MPEG FASTQ dataset

Sampl								
P.aeru								
S.cere								
H.sapi								
T.caca								
H.sapi								
H.sapi								



Assembltrie's Performance vs Information theoretic upper bound on compression



Acknowledgements

- National Science Foundation (NSF) CCF-1619081, CCF-1528132 and CCF-0939370 (Center for Science of Information)
- National Institutes of Health (NIH) GM108348
- The Cancer Genome Collaboratory
- Indiana University Precision Health Initiative