Flexible compressed storage of genomic information beyond file formats: Our experience with CARGO

Łukasz Roguski† and **Paolo Ribeca**‡

†Algorithm Development
Centro Nacional de Análisis Genómico, Barcelona, Spain

‡Integrative Biology
The Pirbright Institute, Woking, UK

Geneva, 20.10.2015

cnag
centre nacional d'anàlisi genòmica
centro nacional de análisis genómico

THE Pirbright INSTITUTE

On the SAM format
and its shortcomings

```
reads100.fastq.000000000        99      chrV    215947  254     100M    =       216365  518
GCCCACCATGATGTGTCTTTCGAGTTAAATAATGCAGATGCTGATAAGCTTCTTGGAATTGATGAGAACGCCAACGTCAATCTGAAGATACAAAG
GTGAA    HIIHIIHIHIHHHGHIJGGGHHIJFGGKHJHKGFMHIKJBGGJEBIJLIHCGILINNIHJCHGI?I@IFOKIIB?DM?
EDRJCCDMPRENCRGAFMSIIM    RG:Z:0  NH:i:1  NM:i:1  XT:A:U  md:Z:96A3
```

- The SAM format is ubiquitous in HTS alignment and storage.
- However:
  - Not rationally designed (output format for BWA/1000-genomes project)
  - Not a universal format for alignment
  - The “standard” is a continuously changing PDF, with no versioning
  - Complex (many fields, optional fields actually conveying almost all the relevant information)
  - Inconsistent:
    - One of the fields is a multi-bit flag encoded as an integer for compactness
    - An arbitrary number of additional very verbose custom fields is allowed
  - Semantics completely hidden (and hence, correctness very hard to check).

This utility explains SAM flags in plain English.
It also allows switching easily from a read to its mate.

Flag: 123 Explain

Switch to mate

Explanation:

- [x] read paired
- [x] read mapped in proper pair
- [ ] read unmapped
- [x] mate unmapped
- [x] read reverse strand
- [x] mate reverse strand
- [x] first in pair
- [ ] second in pair
- [ ] not primary alignment
- [ ] read fails platform/vendor quality checks
- [ ] read is PCR or optical duplicate
- [ ] supplementary alignment

Summary:

read paired
read mapped in proper pair
mate unmapped
read reverse strand
mate reverse strand
first in pair

About semantics
in bioinformatics

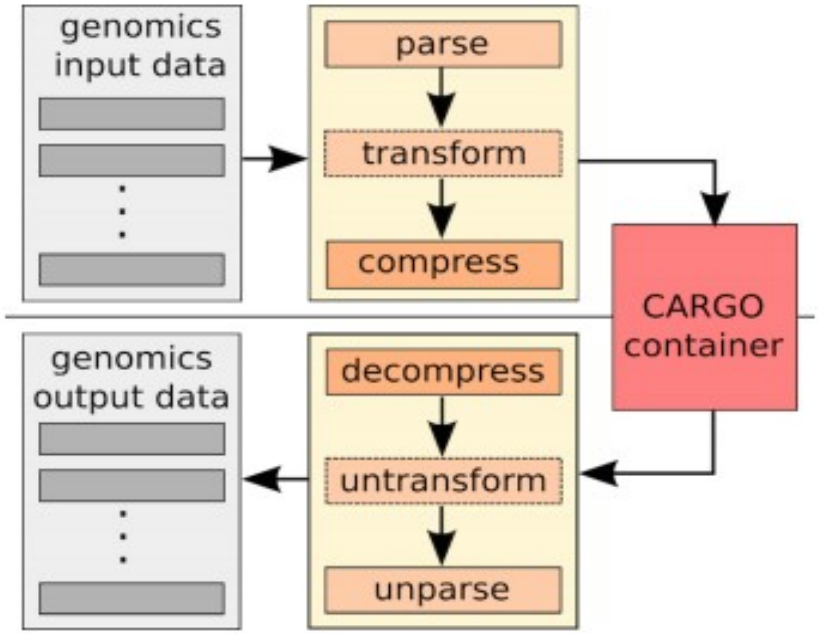
- **Relevant content is almost never represented explicitly.**
  - There are implicit graphs in every GTF/GFF file:

```
1       ensembl gene    1735    16308   .       +       .       gene_id "ENSGALG00000009771"; gene_version "4"; gene_source "ensembl"; gene_biotype
"protein_coding";
1       ensembl transcript      1735    16308   .       +       .       gene_id "ENSGALG00000009771"; gene_version "4"; transcript_id "ENSGALT000000158
transcript_version "4"; gene_source "ensembl"; gene_biotype "protein_coding"; transcript_source "ensembl"; transcript_biotype "protein_coding";
1       ensembl exon    1735    2449    .       +       .       gene_id "ENSGALG00000009771"; gene_version "4"; transcript_id "ENSGALT00000015891";
transcript_version "4"; exon_number "1"; gene_source "ensembl"; gene_biotype "protein_coding"; transcript_source "ensembl"; transcript_biotype
"protein_coding"; exon_id "ENSGALE00000301221"; exon_version "1";
1       ensembl CDS     2379    2449    .       +       0       gene_id "ENSGALG00000009771"; gene_version "4"; transcript_id "ENSGALT00000015891";
transcript_version "4"; exon_number "1"; gene_source "ensembl"; gene_biotype "protein_coding"; transcript_source "ensembl"; transcript_biotype
"protein_coding"; protein_id "ENSGALP00000015874"; protein_version "4";
```

- **In general:**
  - Data manipulation is format-centric. Formats are rigid.
  - A different format for each analysis intermediate?
    Additional data is usually stored as hacked-out “additional fields”.

Our solution: CARGO

and its main concepts



- **Each genomic dataset modeled as:**
  - A header containing meta-information
  - A collection of structured records (with possibly variant type)
- **Record type defined by the user in a rich domain-specific meta-language**

  allowing easy and rapid prototyping of type-specific compressor/decompressor applications.
  Same spirit as that of data-bases (the semantics is in the schema!).

  Record fields can be annotated (for instance, specifying a different compression method for each field).
- **All data stored in containers with configurable size, from MB to PB in size**

  that can contain an arbitrary number of datasets of different formats.

Creating a format-specific compressor
with CARGO

```
OptionalValue = [
  intValue = int^32;
  charValue = char;
  stringValue = string;
] :
  .intValue.Pack = Bzip2L4,
  .charValue.Pack = GzipL2,
  .stringValue.Pack = Bzip2L4;

OptionalField = {
  tag = string;
  value = OptionalValue;
} :
  .tag.Pack = PPMdL4;

MdOperationValue = [
  intValue = int^32;
  stringValue = string;
] :
  .intValue.Pack = Bzip2L4,
  .stringValue.Pack = LZMAL1;

MdOperation = {
  operation =
    enum [
      "None",
      "Match",
      "Substitution",
      "Insert",
      "Delete",
      "SoftClip",
      "HardClip"
    ];
  value = MdOperationValue;
} :
  .operation.Pack = Bzip2L4;
```

```
SamRecord = {
  qname = string;
  flag = uint^16;
  rname = string;
  pos = uint^32;
  mapq = uint^8;
  cigar = string;
  next = string;
  pnext = int^32;
  tlen = int^32;
  seq = string;
  qua = string;
  opt = OptionalField array;
  md = MdOperation array;
} :
  .qname.Pack = PPMdL4,
  .flag.Pack = PPMdL4,
  .rname.Pack = GzipL2,
  .pos.Pack = LZMAL1,
  .mapq.Pack = PPMdL1,
  .cigar.Pack = GzipL1,
  .next.Pack = GzipL2,
  .pnext.Pack = LZMAL1,
  .tlen.Pack = PPMdL4,
  .seq.Pack = LZMAL1,
  .qua.Pack = PPMdL1;

@record SamRecord
```

(1) Define record type in CARGO meta-language

(2) Translate the definition

(generates user files, application template files, Makefile and internal record specifications)

(3) Write record parser (and optionally record transformations, key generation)

by filling the gaps in the C++ template files generated during previous step

(4) Compile CARGO application template files

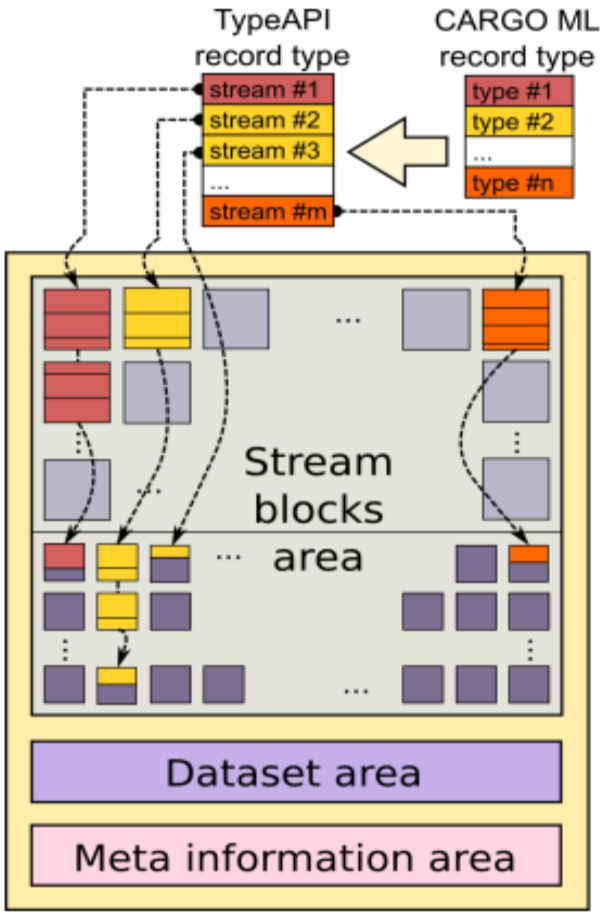
by using the Makefile script generated at step (2).

Result: command-line binary able to write to/read from a CARGO container

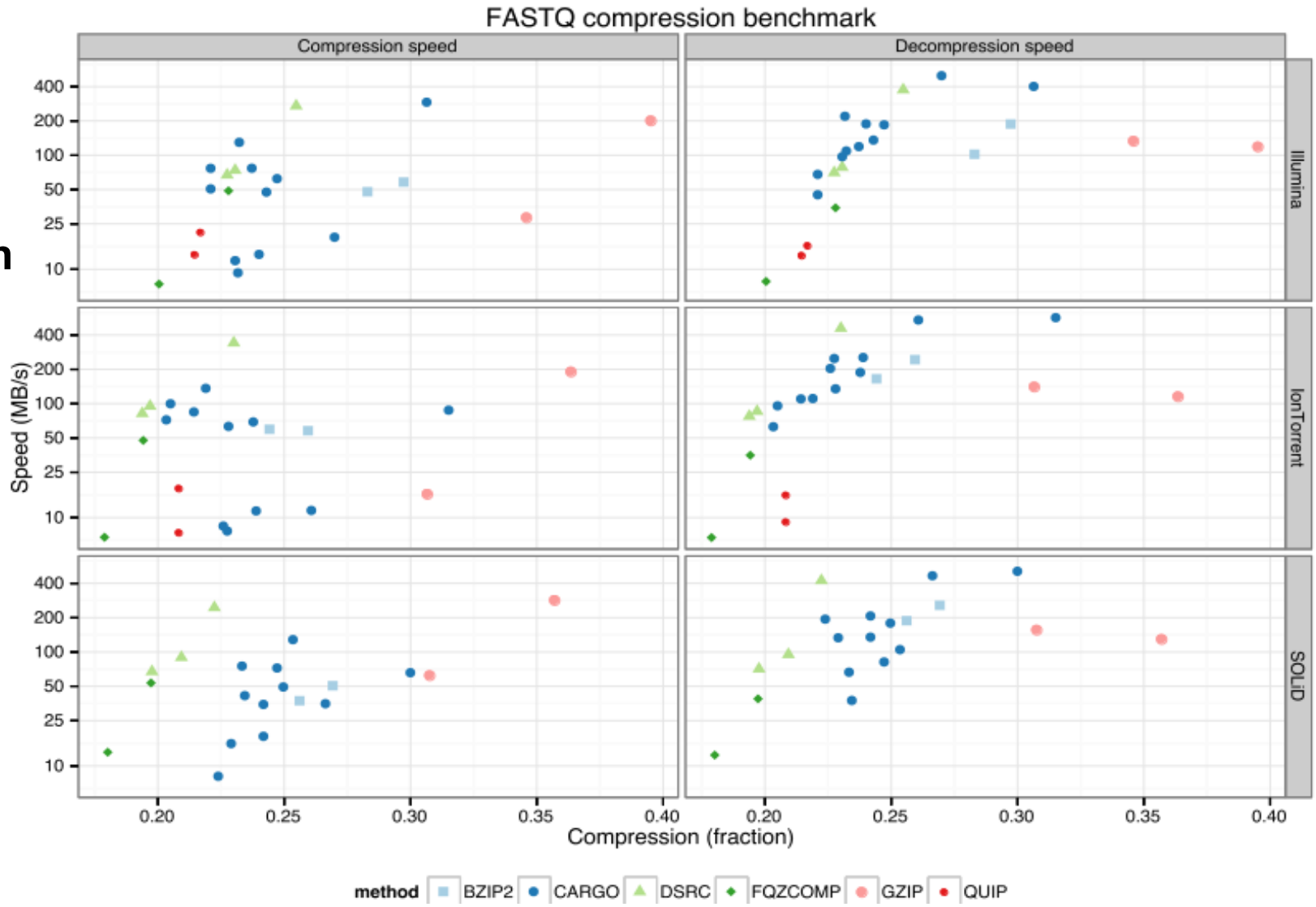
a compressed form of the specified format. **Done only once per new format!**

Data streams and containers
in CARGO

- **Record fields are decomposed in one or more separate *stream(s)***
- **Each field is stored as a set of compressed *stream blocks* within a *container***
- **Each container has separate *meta-information*, *dataset* and *stream-blocks* areas**
  - As stream blocks are stored inside container blocks, multiple datasets can be concurrently interleaved in one container
  - The container is configurable, adaptable and resizable
  - Multiple setups are possible (streaming, random access, max ratio/performance, ...)



FASTQ Compression benchmarks on single archives



SAM compression benchmarks

on single archives

HG01880, 82 GB, unsorted, different compression scenarios

Scenario		Method ID & name	Compression			Speed (MB/s)		Lines of code	
			Size (MB)	Fraction	Fold	Compr.	Decompr.	Meta	C++
Lossy quality binning	1	CARGO-Ref-Q8-Max	3,852	4.7%	21.25	147.1	224.6	67	~2.2K
	2	CARGO-Ref-Q8	5,233	6.4%	15.64	79.0	122.3	67	~2.2K
	3	SCRAMBLE-CRAM-Q8 [8][9]	6,077	7.4%	13.47	110.4	117.9		
	4	CARGO-Std-Q8	6,468	7.9%	12.66	37.6	119.3	28	86
	5	CARGO-Ext-Q8	6,894	8.4%	11.87	47.6	129.9	42	157
	6	DEEZ-Q8 [4]	7,767	9.5%	10.54	25.6	27.9		
Lossless	7	CARGO-Ref	9,623	11.8%	8.51	74.8	115.1	67	~2.2K
	8	DEEZ-Samcomp	10,120	12.4%	8.09	21.8	22.4		
	9	DEEZ-Normal	10,596	12.9%	7.73	25.1	27.1		
	10	SCRAMBLE-CRAM	10,698	13.1%	7.65	100.9	115.6		
	11	SAMTOOLS-CRAM [1][9]	10,712	13.1%	7.64	24.2	13.2		
	12	CARGO-Std	10,869	13.3%	7.53	36.4	111.1	28	58
	13	CARGO-Ext	11,284	13.8%	7.26	46.1	115.9	42	129
	14	BZIP2-Best	14,271	17.4%	5.74	52.2	111.3		
	15	BZIP2-Fast	15,091	18.5%	5.42	63.4	121.7		
	16	GZIP-Best	16,540	20.2%	4.95	95.6	113.3		
	17	SCRAMBLE-BAM [8][9]	18,418	22.5%	4.44	136.4	105.1		
	18	SAMTOOLS-BAM [1][9]	18,420	22.5%	4.44	70.4	86.2		

SAM compression benchmarks on a collection of volumes

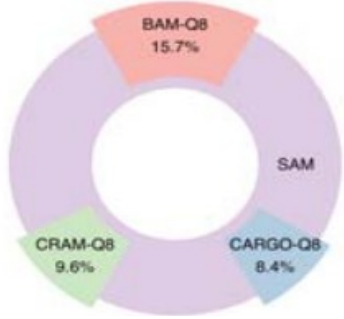
Collection of volumes, 17 TB, sorted, lossy 8-bin quality compression

A

SAM fields			CARGO streams		
Name	**Uncompressed size (GB)**	**Fraction**	**Name**	**Compressed size (GB)**	**Fraction of SAM**
QNAME	695.9	4.1%	QNAME	170.3	0.996%
FLAG	101.7	0.6%	FLAG	10.8	0.063%
RNAME	67.8	0.4%	RNAME	0.3	0.002%
POS	325.3	1.9%	POS	2.7	0.016%
MAPQ	76.2	0.5%	MAPQ	4.9	0.029%
NEXT	170.6	1.0%	NEXT	3.1	0.018%
PNEXT	321.5	1.9%	PNEXT	50.1	0.293%
TLEN	134.5	0.8%	TLEN	12.8	0.075%
QUA	3,803.4	22.3%	QUA	785.5	4.600%
SEQ	3,803.4	22.3%	SEQ*	11.1	0.065%
CIGAR	40.1	0.2%	CIGAR*	2.2	0.013%
OPT	6,719.9	39.3%	OPT*	231.0	1.351%
			MD*	151.2	0.885%
Whitespace	833.8	4.9%			
Total	17,094.1	100.0%	**Total**	1,435.8	8.4%

B

Method ID & name		Compression		
		Size (GB)	**Fraction of SAM**	**Fold vs. SAM**
	SAM	17,094	100.00%	1.00
17	SCRAMBLE-BAM-Q8	2,697	15.77%	6.34
3	SCRAMBLE-CRAM-Q8	1,633	9.56%	10.47
2	CARGO-Q8	1,436	8.40%	11.91



WARNING: this slide is very much Illumina-specific

A problem of data & software scaling

Throughput constantly improves much more than computing power

- **The first Solexa Genome Analyzer @ CRG, 2008: 3 Gb / run**
  and at the beginning (2010), the CNAG used to have 12 GA IIx: 30 Gb / run each.
- **Current Illumina HiSeq 2000 machines produce > 500 Gb / run**
  after some relatively minor hardware upgrade, which gave a 3x boost.

  The CNAG has 10 of them.
- **The CNAG has a peak sequencing capacity of > 700 Gb / day...**
  while the one originally planned in 2010 was 100 Gb / day.
- **...but still "only" 1000 processors in its cluster**
  which is the amount originally intended to process 100 Gb / day.
- **And it's happening again!**
  New 3x upgrade under way. **Will get Illumina HiSeq 4000 machines soon!**

Putting things in perspective
and getting some outlook

- **Yield is essentially an Illumina-specific issue.**
  - Short reads require high coverages (30x? 100x? 300X?)
  - (Sufficiently) Long reads require lower coverages: remember Sanger and 454! (7x?)
- **FASTQ/SAM-like formats can be used only:**
  - When base-calling is well-established • When there are few indels.
- **Orthogonal realignment formats for long reads already exist**

  à la Myers, storing intervals in the alignment matrix (linear time reconstructions) rather than differences
- **More complex format already used by long-reads vendors:**
  - PacBio (“extended SAM”) • Oxford sNanopore (HDF5, base calling information).
- **Even if quality scores were sufficient, semantics, intermediates and HPC always a problem.**

A word of caution about benchmarks in genomics

Genomics is *not* engineering!

- **(Individual) genome assembly is *not* a well-established procedure**
  - Many (repetitive) parts of the genome are missing from the reference
- **Variant calling is *not* a well-established procedure**
  - SNPs are OK, large scale variants are not (re-sequencing with short reads!)
  - ICGC paper in press showing that:
    - The same software calls very different sets of variants depending on the parameters used
    - Some classes of variants (for instance in repeats) are there but hard(er) to call
    - Benchmarks very difficult because we don't know the truth (i.e. all the variants present).
- **Premature optimization is the root of all evil!**
  - Too early to talk about “quality de-noising”? Keeping all the information might be a safer bet.


Thank you for your attention!