Processing Terabyte Scale Genomics Datasets with ADAM

Frank Austin Nothaft
University of California, Berkeley
@fnothaft

SPARK SUMMIT
EAST 2017

Genome Resequencing

- When we sequence a human genome, we obtain several hundred GB of raw sequence data
- With a reference genome, we can use this sequence to compute diffs between individuals
- Two problems:
  - How do we compute this diff?
  - How do we make sense of the differences?

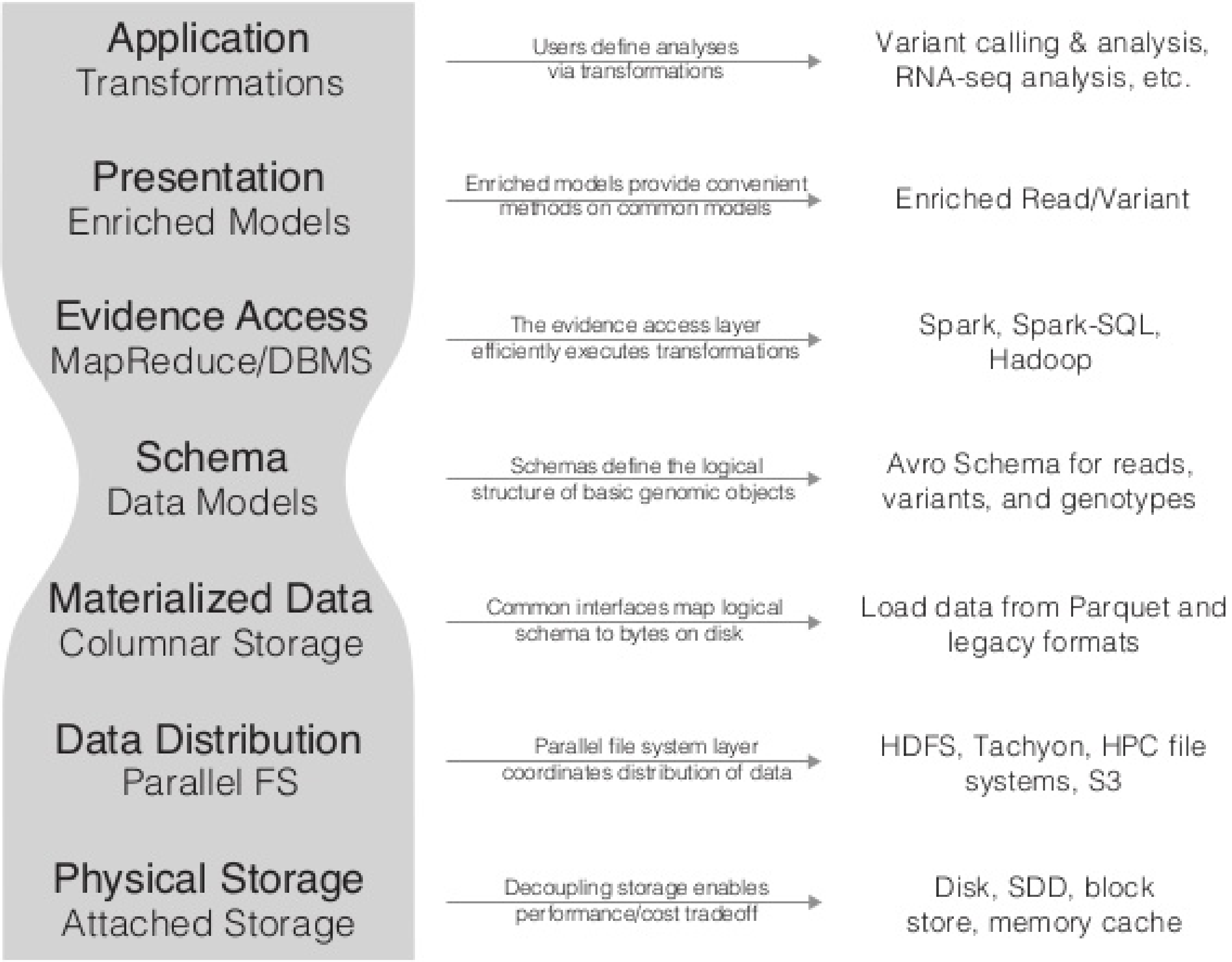
Building Scalable Genomics Tools on ADAM

- ADAM is an open source, high performance, distributed library for genomic analysis
- ADAM defines a:
  - Data schema and layout on disk
  - Programming interface for distributed processing of genomic data using Spark + Scala
- Goal is to enable both batch and exploratory analysis of all types of genomic data

Genomics is built around flattened, single node tools

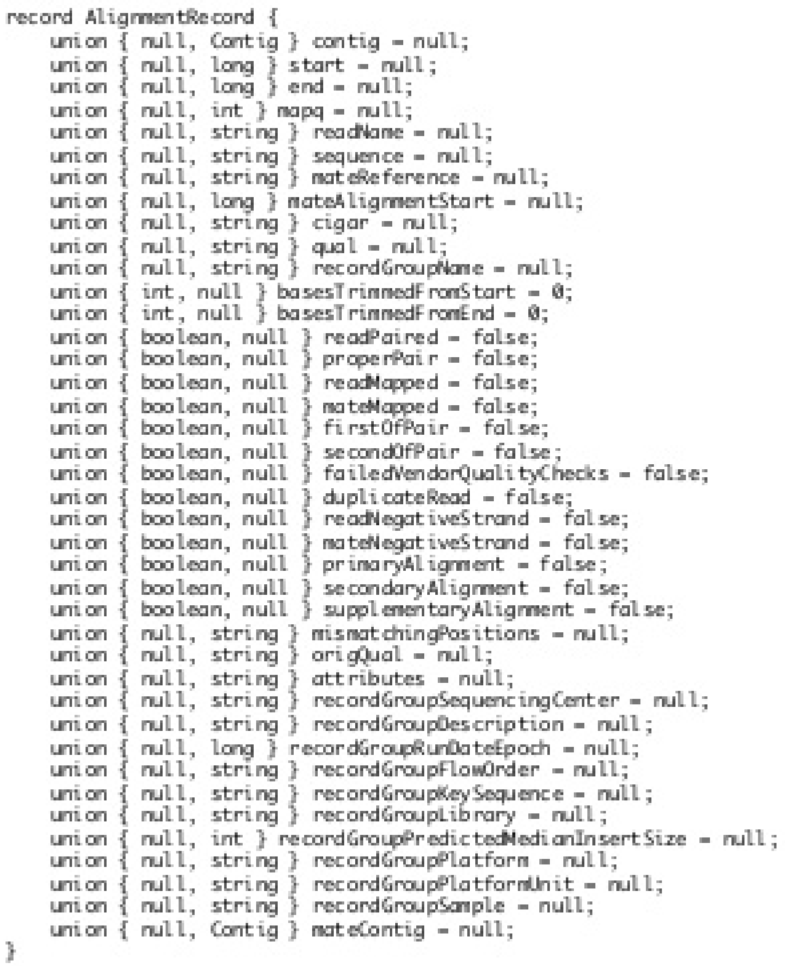
- Legacy flat-file formats:
  - Manually curated text/binary flat files
  - E.g., SAM/BAM → alignment, VCF → variants, BED/GTF/etc → features
- These formats scale poorly beyond single computer storage/compute capacity
- These legacy formats are functionally limiting and bug-prone:
  - What accesses can be optimized (read a full row)
  - What predicates can be evaluated (small number of genomic loci)
  - How we write genomic algorithms (sorted iterator over genome)
  - How we avoid technical lock-in (extend metadata)

ADAM uses a schema as a narrow waist

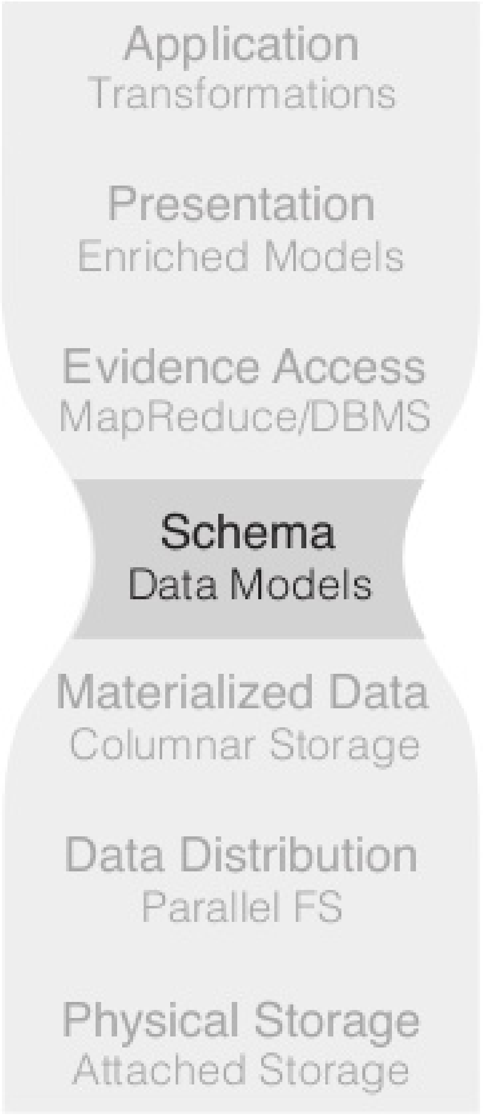


Application Transformations	Users define analyses via transformations	Variant calling & analysis, RNA-seq analysis, etc.
Presentation Enriched Models	Enriched models provide convenient methods on common models	Enriched Read/Variant
Evidence Access MapReduce/DBMS	The evidence access layer efficiently executes transformations	Spark, Spark-SQL, Hadoop
Schema Data Models	Schemas define the logical structure of basic genomic objects	Avro Schema for reads, variants, and genotypes
Materialized Data Columnar Storage	Common interfaces map logical schema to bytes on disk	Load data from Parquet and legacy formats
Data Distribution Parallel FS	Parallel file system layer coordinates distribution of data	HDFS, Tachyon, HPC file systems, S3
Physical Storage Attached Storage	Decoupling storage enables performance/cost tradeoff	Disk, SDD, block store, memory cache

ADAM uses a schema as a narrow waist

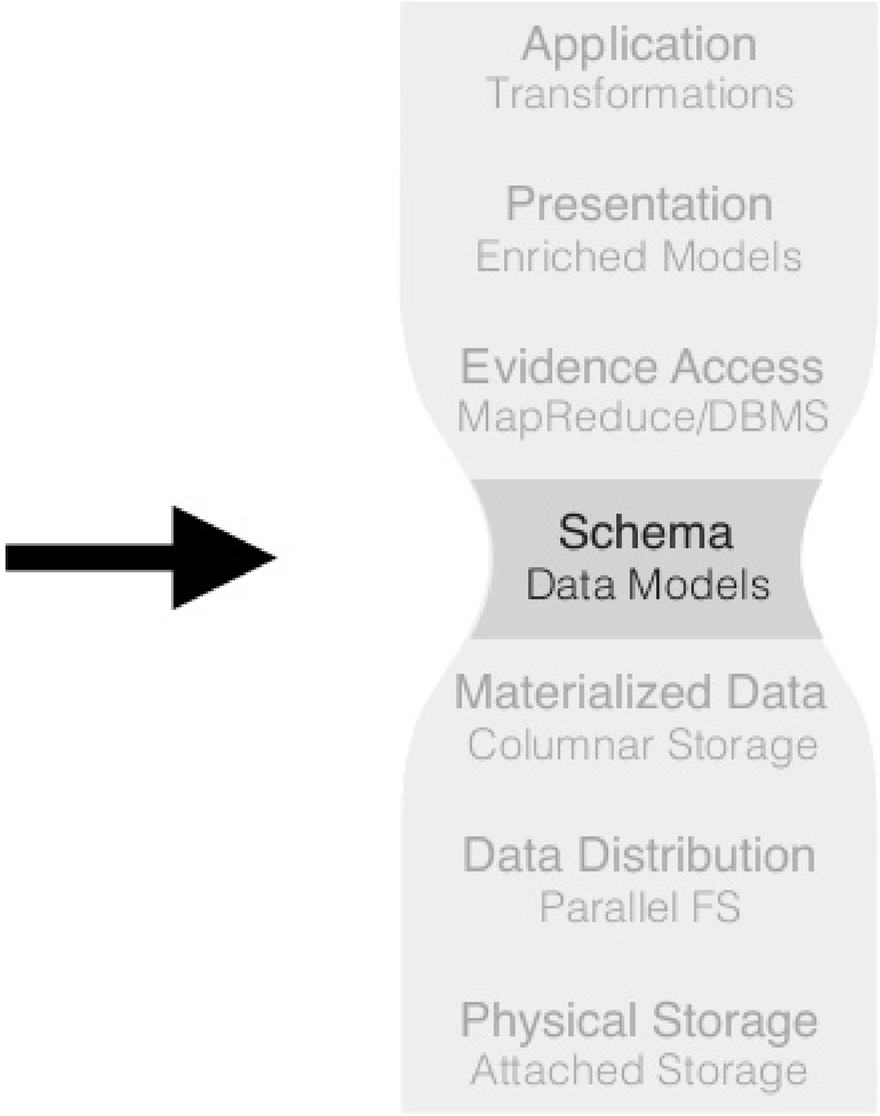


```
record AlignmentRecord {
    union { null, Contig } contig = null;
    union { null, long } start = null;
    union { null, long } end = null;
    union { null, int } mapq = null;
    union { null, string } readName = null;
    union { null, string } sequence = null;
    union { null, string } mateReference = null;
    union { null, long } mateAlignmentStart = null;
    union { null, string } cigar = null;
    union { null, string } qual = null;
    union { null, string } recordGroupName = null;
    union { int, null } basesTrimmedFromStart = 0;
    union { int, null } basesTrimmedFromEnd = 0;
    union { boolean, null } readPaired = false;
    union { boolean, null } properPair = false;
    union { boolean, null } readMapped = false;
    union { boolean, null } mateMapped = false;
    union { boolean, null } firstOfPair = false;
    union { boolean, null } secondOfPair = false;
    union { boolean, null } failedVendorQualityChecks = false;
    union { boolean, null } duplicateRead = false;
    union { boolean, null } readNegativeStrand = false;
    union { boolean, null } mateNegativeStrand = false;
    union { boolean, null } primaryAlignment = false;
    union { boolean, null } secondaryAlignment = false;
    union { boolean, null } supplementaryAlignment = false;
    union { null, string } mismatchingPositions = null;
    union { null, string } origQual = null;
    union { null, string } attributes = null;
    union { null, string } recordGroupSequencingCenter = null;
    union { null, string } recordGroupDescription = null;
    union { null, long } recordGroupRunDateEpoch = null;
    union { null, string } recordGroupFlowOrder = null;
    union { null, string } recordGroupKeySequence = null;
    union { null, string } recordGroupLibrary = null;
    union { null, int } recordGroupPredictedMedianInsertSize = null;
    union { null, string } recordGroupPlatform = null;
    union { null, string } recordGroupPlatformUnit = null;
    union { null, string } recordGroupSample = null;
    union { null, Contig } mateContig = null;
}
```

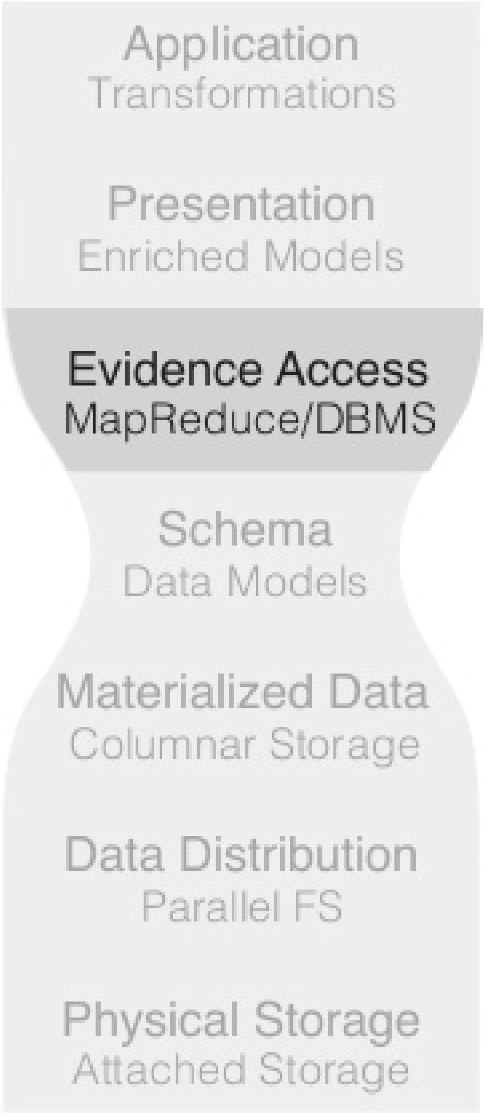


ADAM uses a schema as a narrow waist

- ADAM has schemas for:
  - Reads: SAM/BAM/CRAM, FASTQ
  - Features: BED/GTF/GFF2,3/NarrowPeak/IntervalList
  - Variants/Genotypes: (g)VCF/BCF1
  - Sequence: FASTA



Having a stack makes it easy to accelerate genomic queries



Application
Transformations

Presentation
Enriched Models

Evidence Access
MapReduce/DBMS

Schema
Data Models

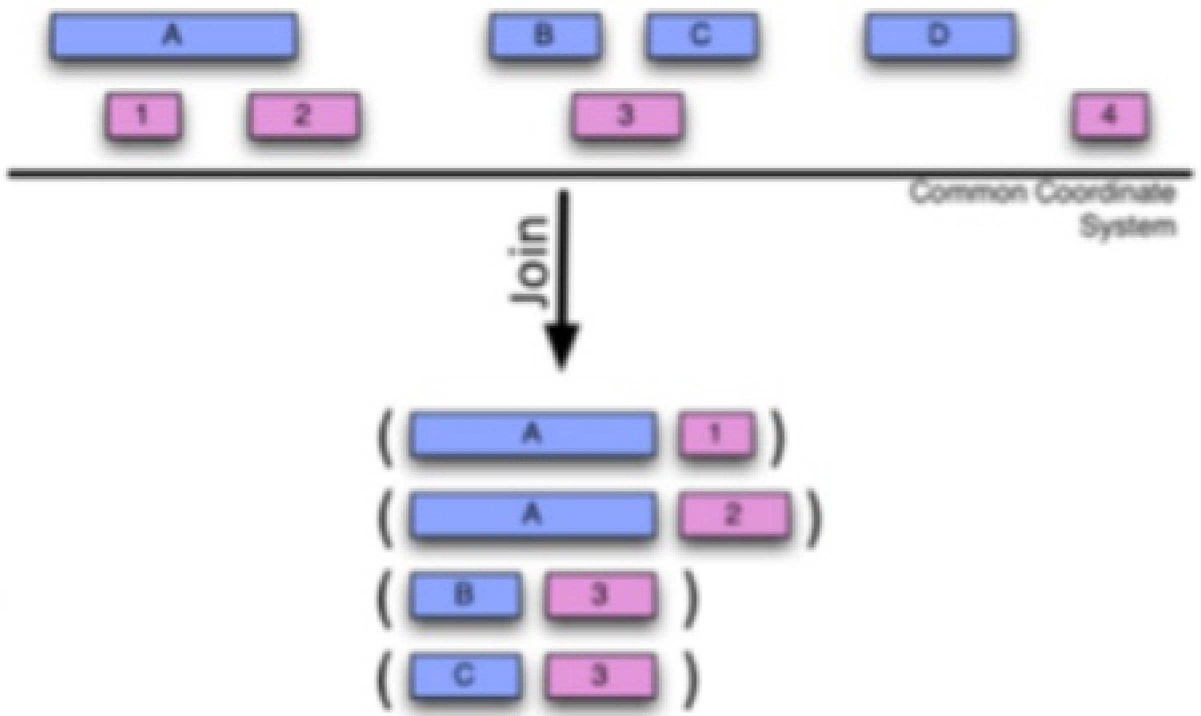
Materialized Data
Columnar Storage

Data Distribution
Parallel FS

Physical Storage
Attached Storage

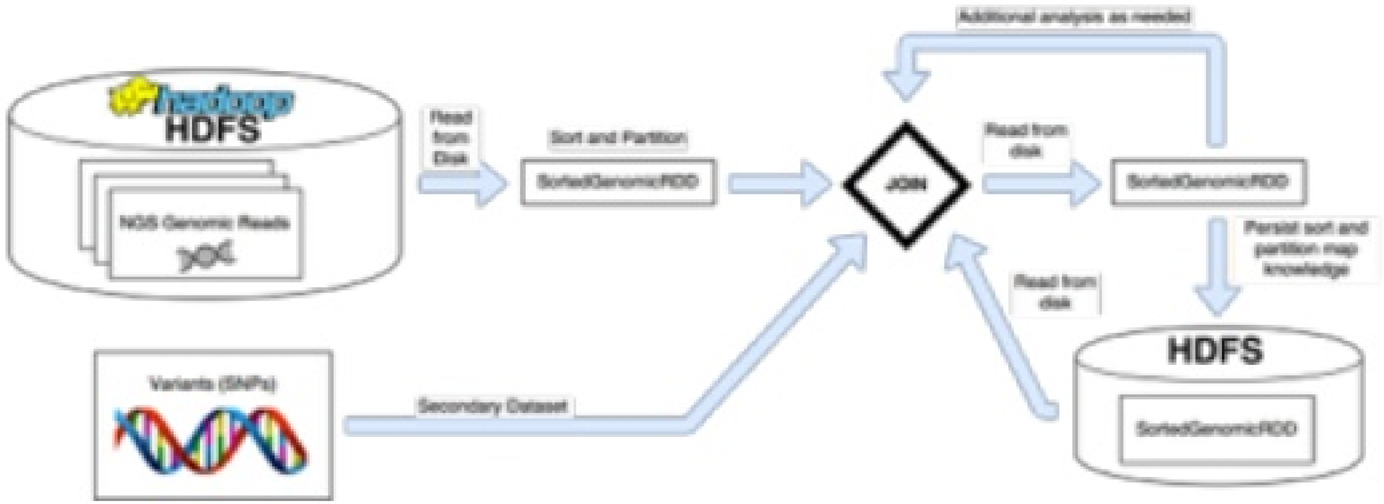
...while also providing higher level abstractions

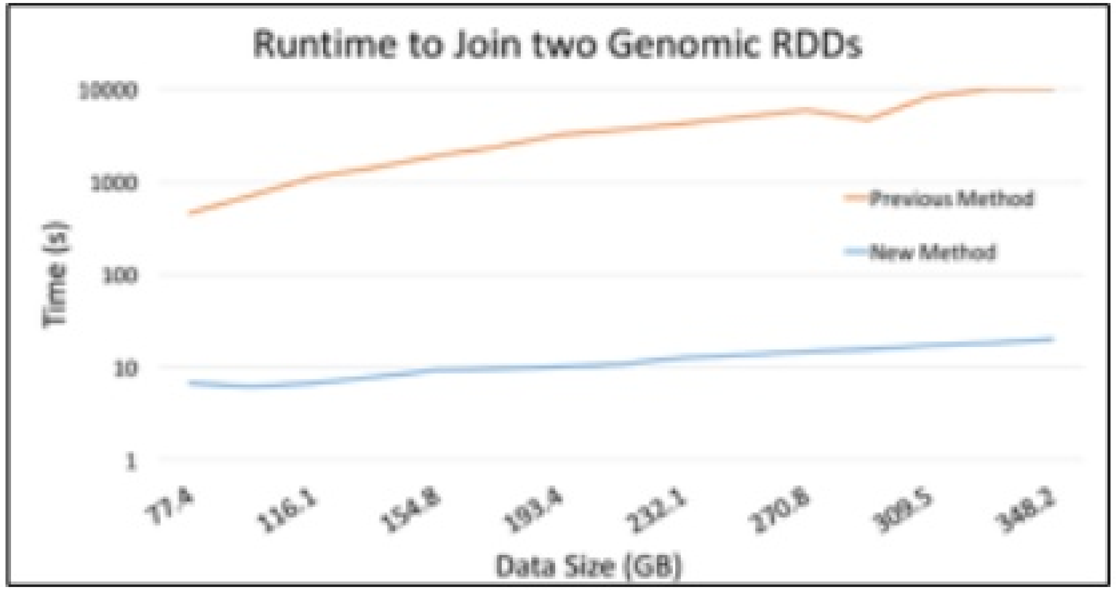
- ADAM eliminates need to use “genome walker”:
  - Use region join for overlap computation
  - Use group/reduceByKey functions from Spark to process features aligned at a genomic coordinate point
- Can reduce targeted regions across a genome via sort + fold
- Higher level primitives enable optimizations:
  - Can leverage indices/sort orders
  - Can push down join/filter queries into storage



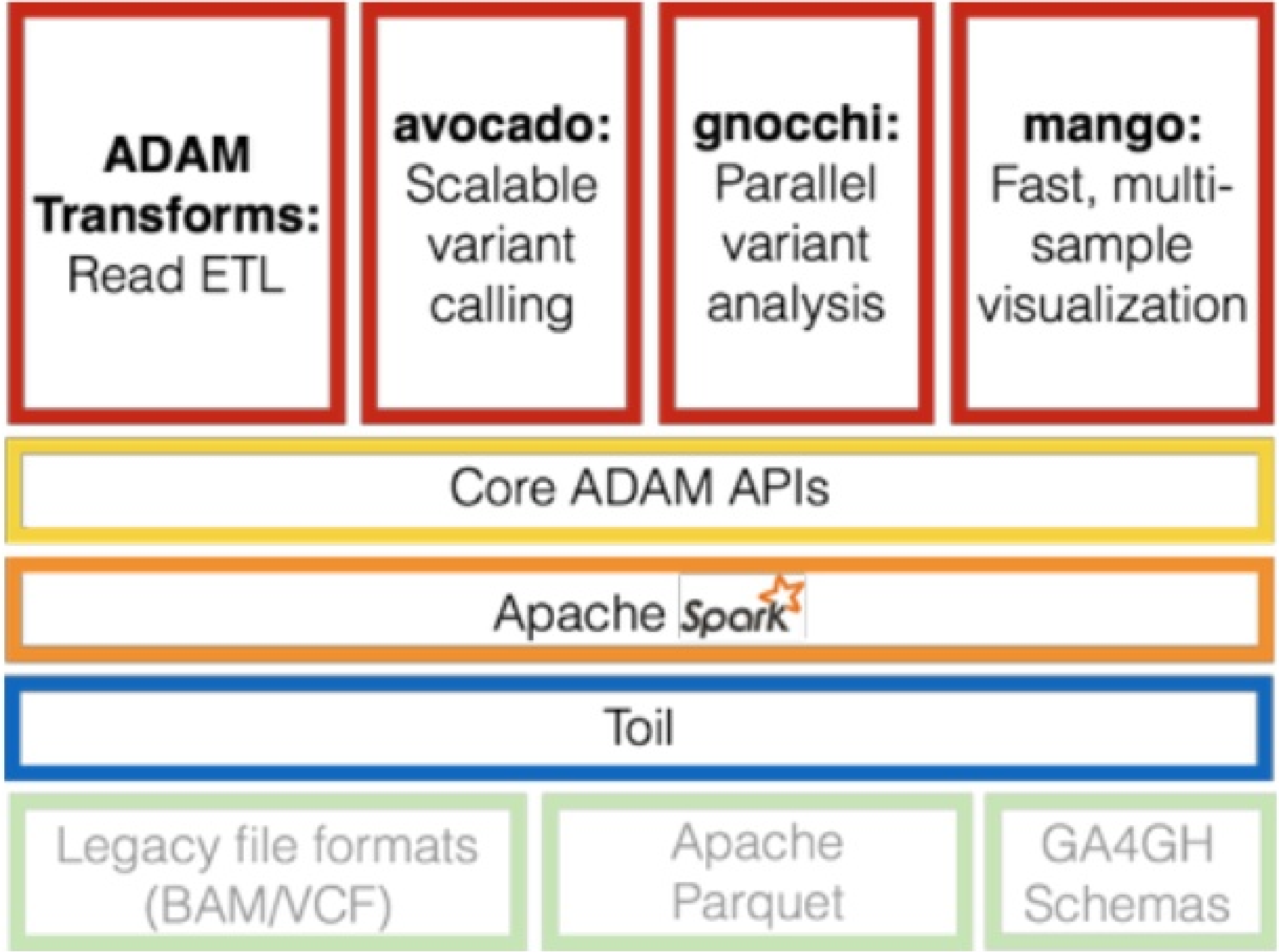
Higher level primitives enable optimizations

- Maintain sort order across runs and optimize to reduce data skew
- Leverage indices/sort orders
- Push down join/filter queries into
  storage
- Use join optimizations to develop BEDtools equivalent





Big Data Genomics Stack



ADAM Transforms: Read ETL

avocado: Scalable variant calling

gnocchi: Parallel variant analysis

mango: Fast, multi-sample visualization

Core ADAM APIs

Apache Spark

Toil

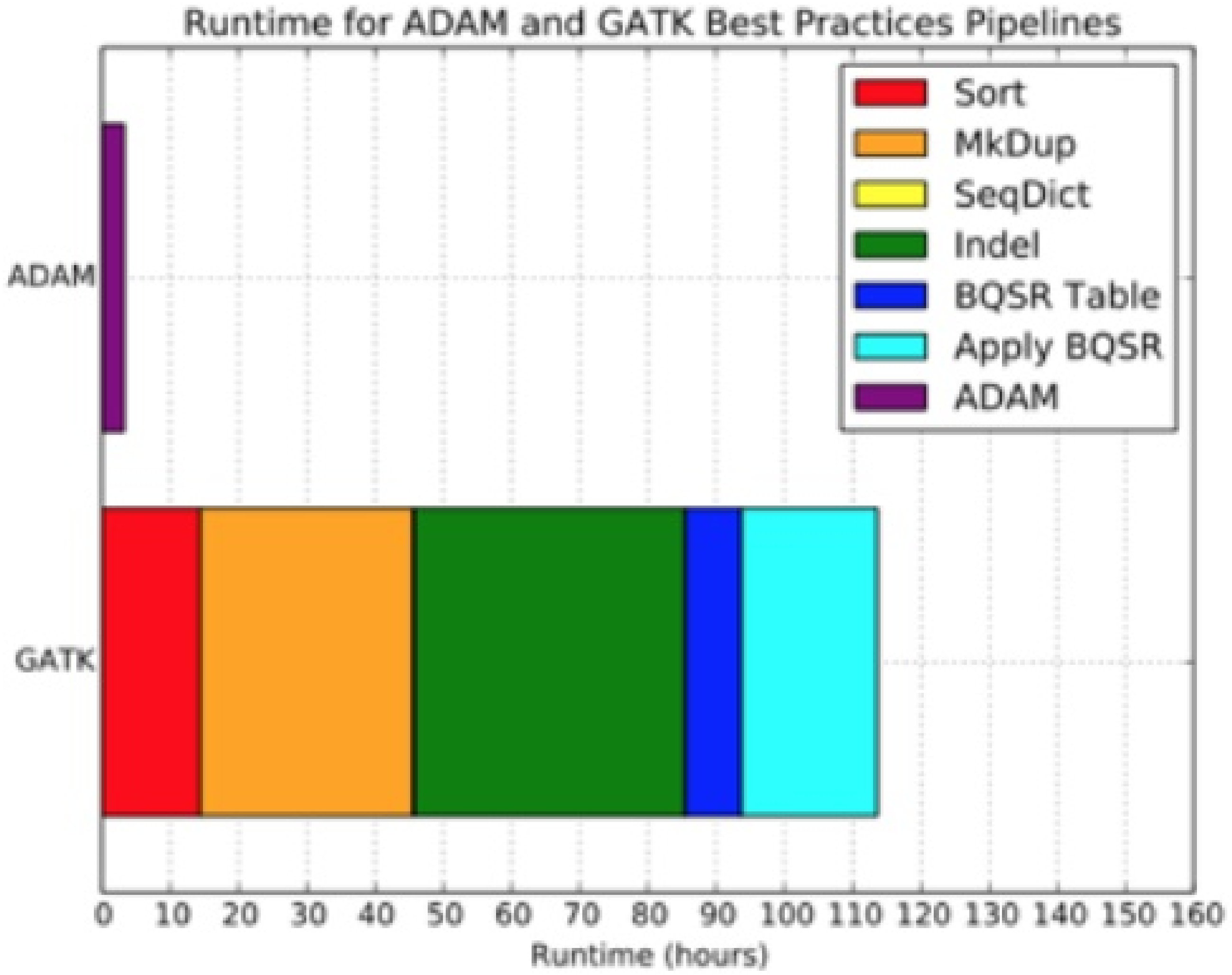
Legacy file formats (BAM/VCF)

Apache Parquet

GA4GH Schemas

Benchmarking ADAM

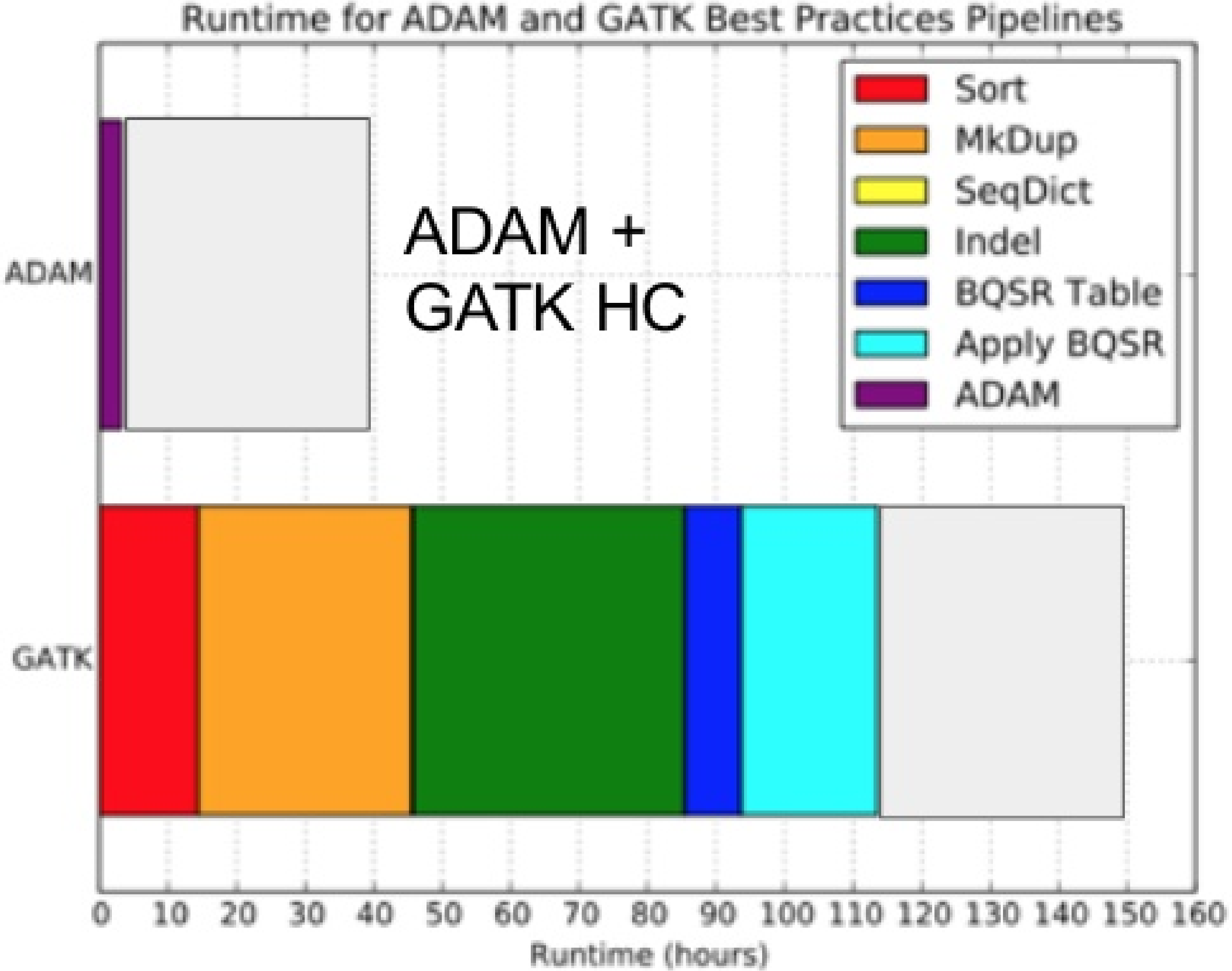
- ADAM produces statistically equivalent results to the GATK best practices pipeline
- Read preprocessing is >30x faster and 3x cheaper



SPARK SUMMIT EAST 2017

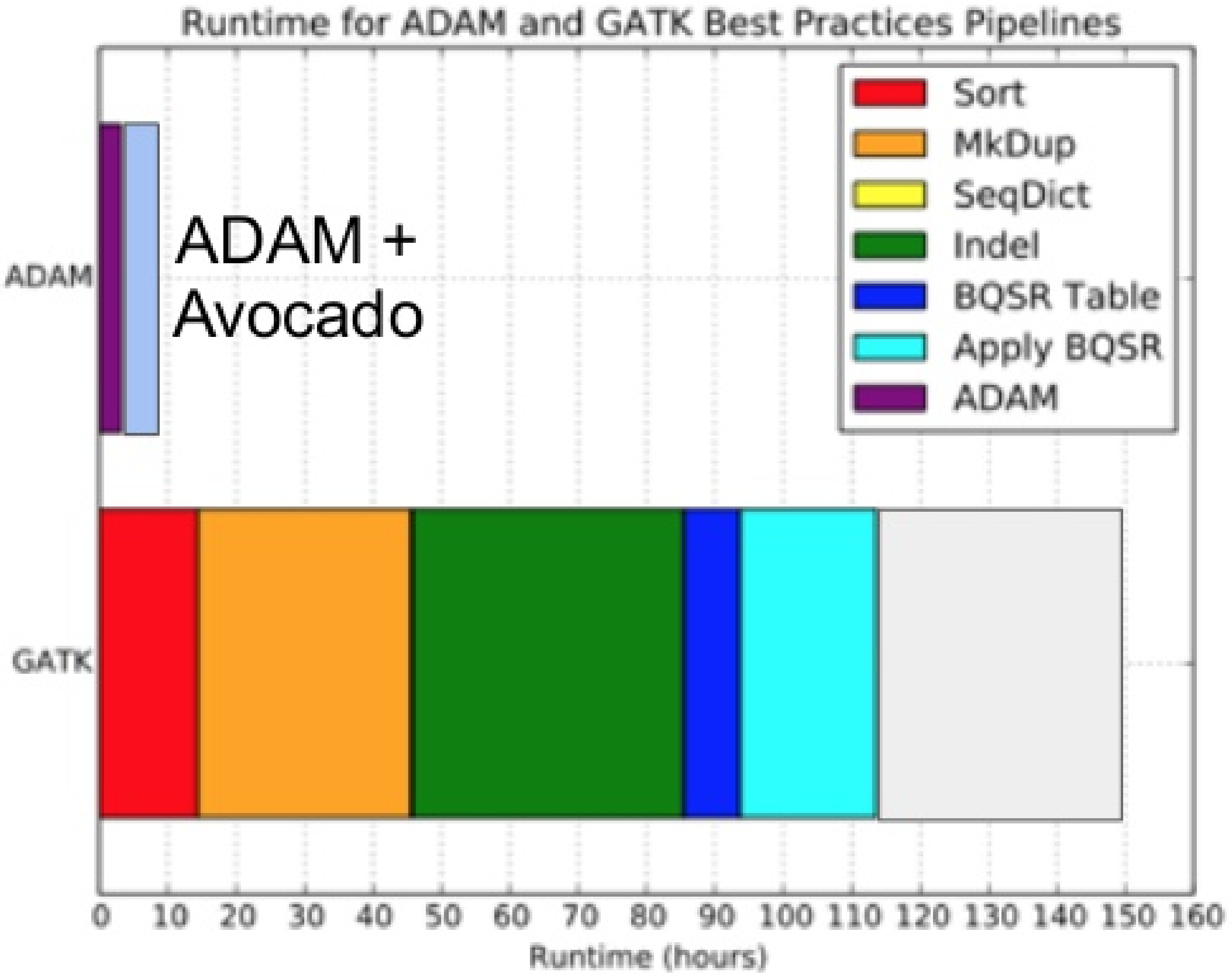
Benchmarking ADAM

- ADAM produces statistically equivalent results to the GATK best practices pipeline
- Read preprocessing is >30x faster and 3x cheaper, end-to-end pipeline is 4x faster, 3.5x cheaper

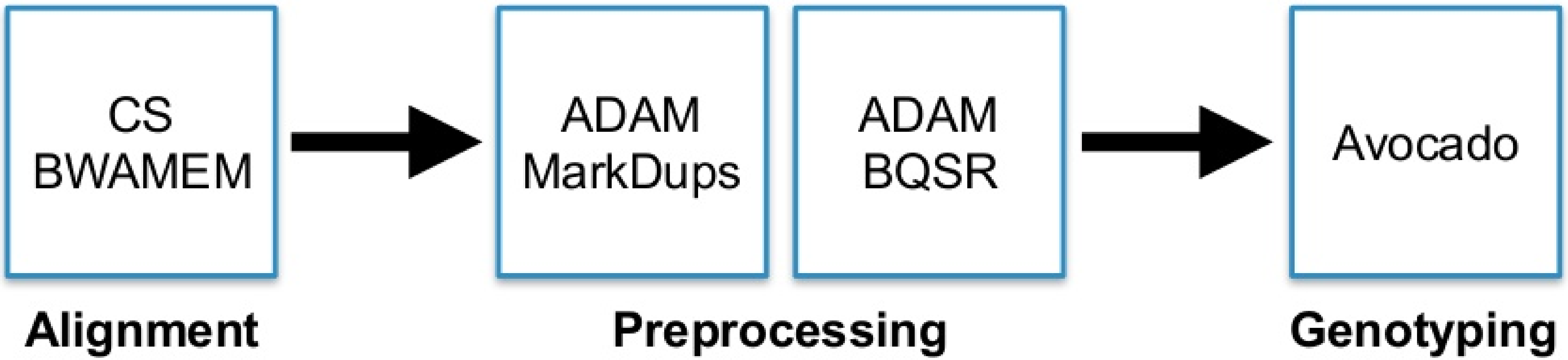


Benchmarking ADAM + Avocado

- Avocado outperforms GATK at SNP calling, slightly behind on INDELs
- Overall pipeline is >17x faster and 2x cheaper
- Avocado relies on novel, efficient INDEL canonicalization engine, drops INDEL discovery cost by 5x

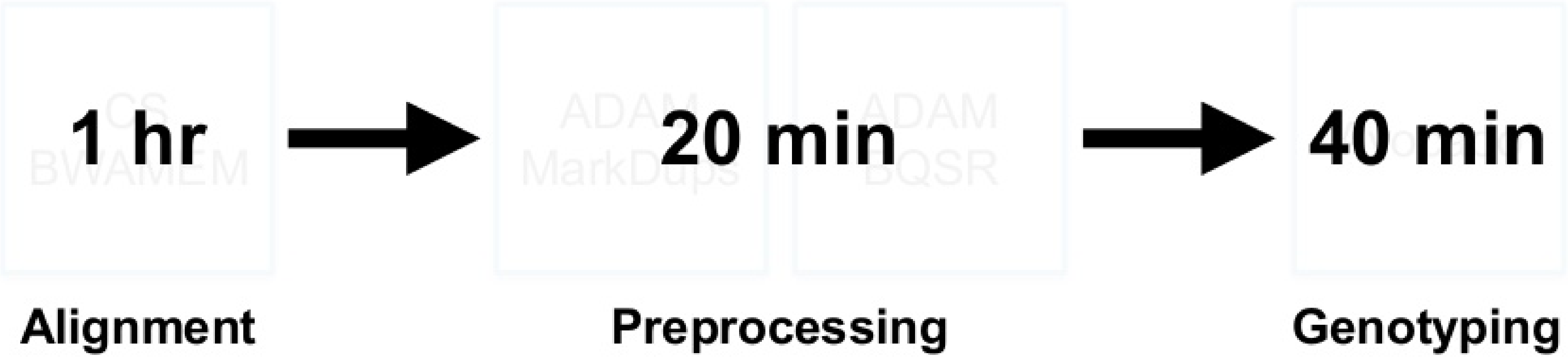


End-to-end variant analysis in Spark



- Can process a 65x whole genome in <2hrs on 1,024 cores
- CS-BWAMEM: https://github.com/ytchen0323/cloud-scale-bwamem

End-to-end variant analysis in Spark


CS BWAMEM

1 hr

Alignment

ADAM MarkDups

ADAM BQSR

20 min

Preprocessing

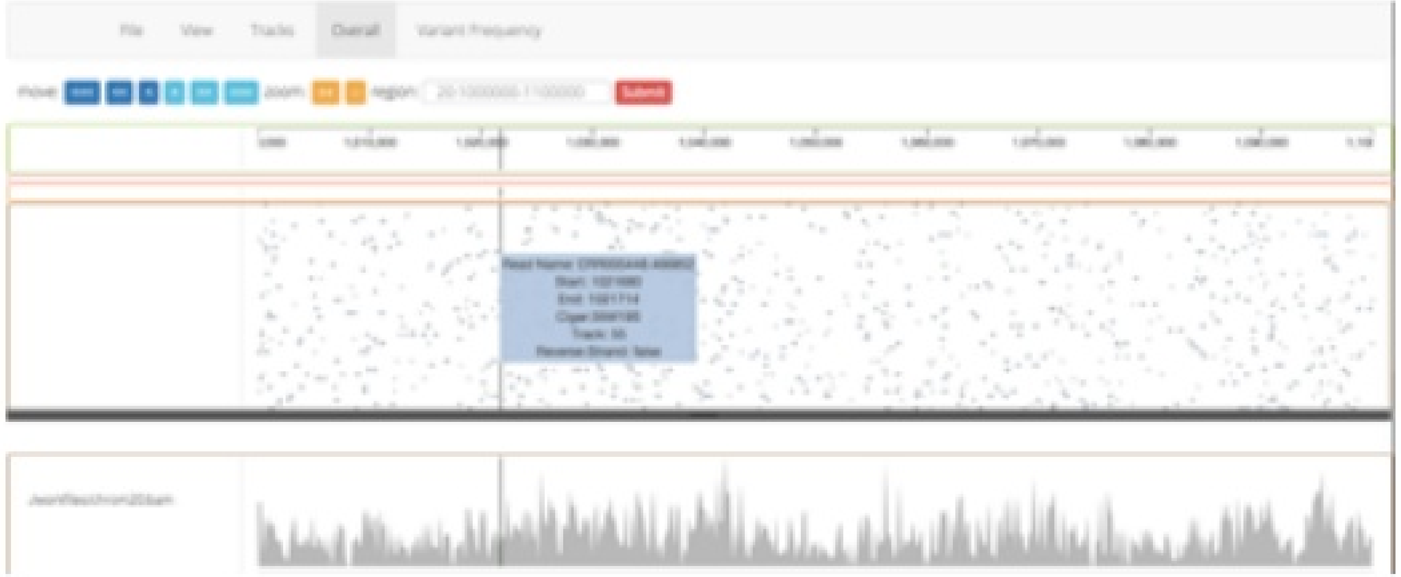
Avocado

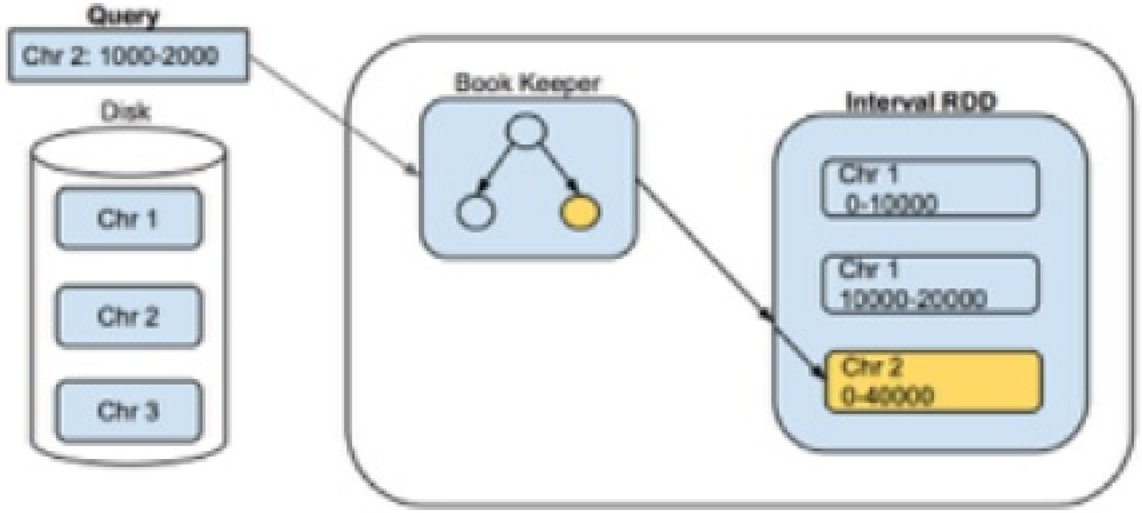
40 min

Genotyping

- Can process a 65x whole genome in <2hrs on 1,024 cores
- CS-BWAMEM: https://github.com/ytchen0323/cloud-scale-bwamem

Optimizing for genomic EDA

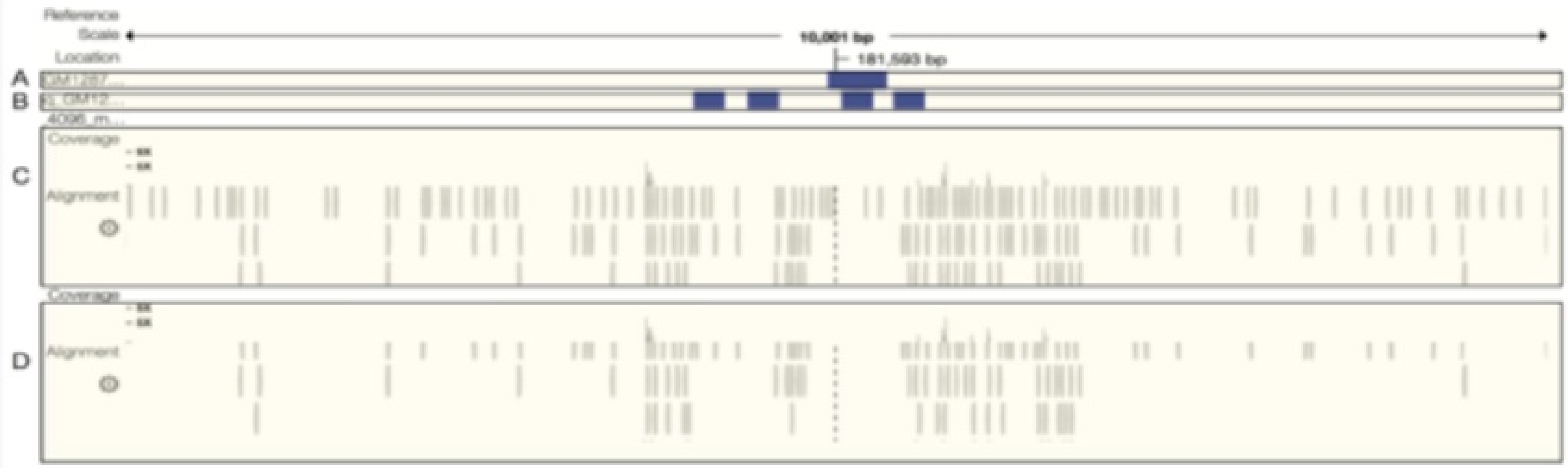




Narrow waist in stack → can swap in/out levels of stack

- For interactive queries against genomic loci, swap in RDD implementation optimized for point/range queries
- Persistent store optimizations minimize initial overhead for fetching raw data
- Memory optimizations minimize latency for genomic range queries

Benefit of stack: intersection of technologies



- Apply the model interactively to a new dataset using Mango, use join query to overlap “ground truth” data against predictions
- 10kbp query+apply latency: ~400ms

Ongoing work: variant warehousing

- Reads yield genotypes, but we're often interested in statistical aggregates across genotypes:
  - Probability of seeing a genotype in a population
  - Probability of a genotype associating with a phenotype
- Data typically is arriving (near) continuously

Demonstrating Incremental Update in Gnocchi

- Problem: want to compute associations between genotypes and phenotypes (linear/logistic regression)
- Solution: Incremental update of many small GnocchiModels
    - Train each distributed model using standard methods
    - When new data added, build locally optimized model on new data and merge resulting model with old model (do not need old data!)
- Work in progress:
    - Requires periodic recomputes over entire cohort to remain close to full recompute solution
    - Can limit the number of recomputes by being smart about haplotype blocks

Acknowledgements

- **UC Berkeley:** Matt Massie, Timothy Danford, André Schumacher, Jey Kottalam, Karen Feng, Eric Tu, Alyssa Morrow, Niranjan Kumar, Ananth Pallaseni, Michael Heuer, Justin Paschall, Taner Dagdelen, Devin Petersohn, Anthony D. Joseph, Dave Patterson
- **Mt. Sinai:** Arun Ahuja, Neal Sidhwaney, Ryan Williams, Michael Linderman, Jeff Hammerbacher, Uri Laserson
- **GenomeBridge:** Carl Yeksigian
- **Cloudera:** Uri Laserson, Tom White
- **Microsoft Research:** Ravi Pandya, Bill Bolosky
- **UC Santa Cruz:** Benedict Paten, David Haussler, Hannes Schmidt, Beau Norgeot, Audrey Musselman-Brown, John Vivian
- And many other open source contributors, especially Neil Ferguson, Andy Petrella, Xavier Tordior, Deborah Siegel, Denny Lee
- Over 60 contributors to ADAM/BDG from >12 institutions

Thank You.

Check out the code: https://github.com/bigdatagenomics

Check out a demo: https://databricks.com/blog/2016/05/24/genome-sequencing-in-a-nutshell.html

Run ADAM in Databricks CE: http://goo.gl/xK8x7s

SPARK
SUMMIT
EAST 2017