# Analyzing massive genomics datasets using Databricks

Frank Austin Nothaft, PhD
frank.nothaft@databricks.com
@fnothaft

**VISION** Accelerate innovation by unifying data science, engineering and business

---

**PRODUCT** Unified Analytics Platform powered by Apache Spark

---

**WHO WE ARE**

- Founded by the creators of Apache Spark
- Contributes **75%** of the open source code, **10x** more than any other company
- Trained **40k+** Spark users on the Databricks platform

# What is Apache Spark?

- A distributed, next-generation map-reduce system
  - Splits a dataset into chunks that can be processed in parallel
  - APIs that allow you to functionally or relationally manipulate datasets
  - The Apache Spark engine then executes these queries
- What does map-reduce look like for genomics data?

```
val kmers = sc.loadAlignments("/path/to/my/reads.sam")
    .rdd
    .flatMap(_.getSequence.sliding(21).map(k => (k, 1L)))
    .reduceByKey(_ + _)
```

# Why use Apache  for genomics?

- Lots of movement in the OSS bioinformatics community:
  - Both ADAM and Hail provide SQL-like interfaces to genomics data
  - Both ADAM and GATK4 provide rapid variant calling pipelines
  - Hail, SparkSeq, and VariantSpark provide statgen/ML methods on large variation datasets
- Why are these tools being built on Spark?
  - ...Spark solves a lot of common genomics pain points!

# Common genomics pain points

1. Analyses are slow and need to be manually parallelized, if they can be parallelized at all
2. Ad hoc analysis of large datasets is generally impractical
3. Analyses are hard to construct and share

# Common genomics pain points

1. **Analyses are slow and need to be manually parallelized, if they can be parallelized at all**
2. Ad hoc analysis of large datasets is generally impractical
3. Analyses are hard to construct and share

**Great fit for Apache Spark!**

Spark understands how to read genomics formats in parallel
Spark provides a high level interface for parallel query
And, you can use Spark to auto-parallelize tools

# Common genomics pain points

1. Analyses are slow and need to be manually parallelized, if they can be parallelized at all
2. **Ad hoc analysis of large datasets is generally impractical**
3. Analyses are hard to construct and share

**Use parallelism to drop latency:**

Easy to write SQL-like queries that are run across a cluster

Scaling out allows you to run alignment in <15min

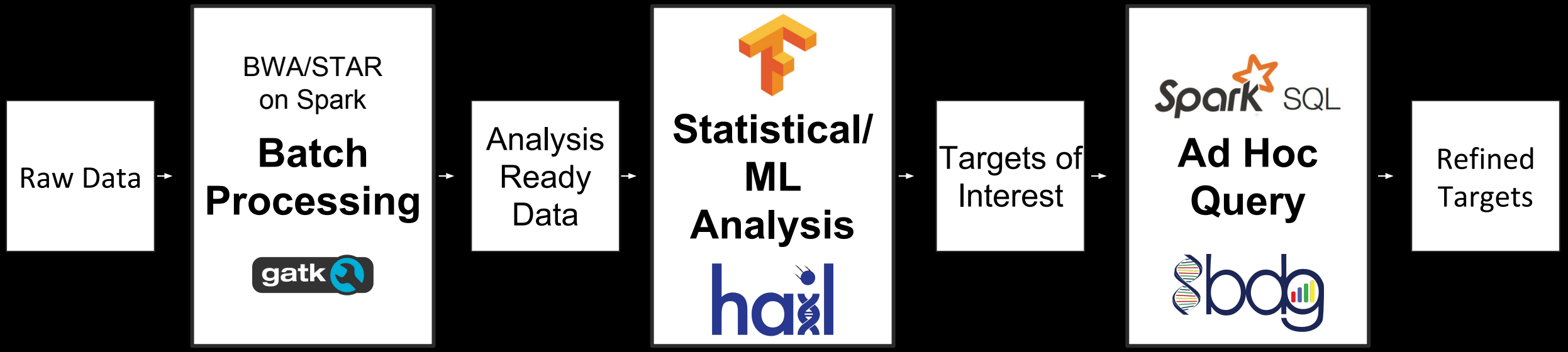
# Common genomics pain points

1. Analyses are slow and need to be manually parallelized, if they can be parallelized at all
2. Ad hoc analysis of large datasets is generally impractical
3. **Analyses are hard to construct and share**

**Spark increases the "level of abstraction":**

HPC-oriented genomics libraries do a lot of "stack smashing"
Spark is designed so you can write performant, high level code

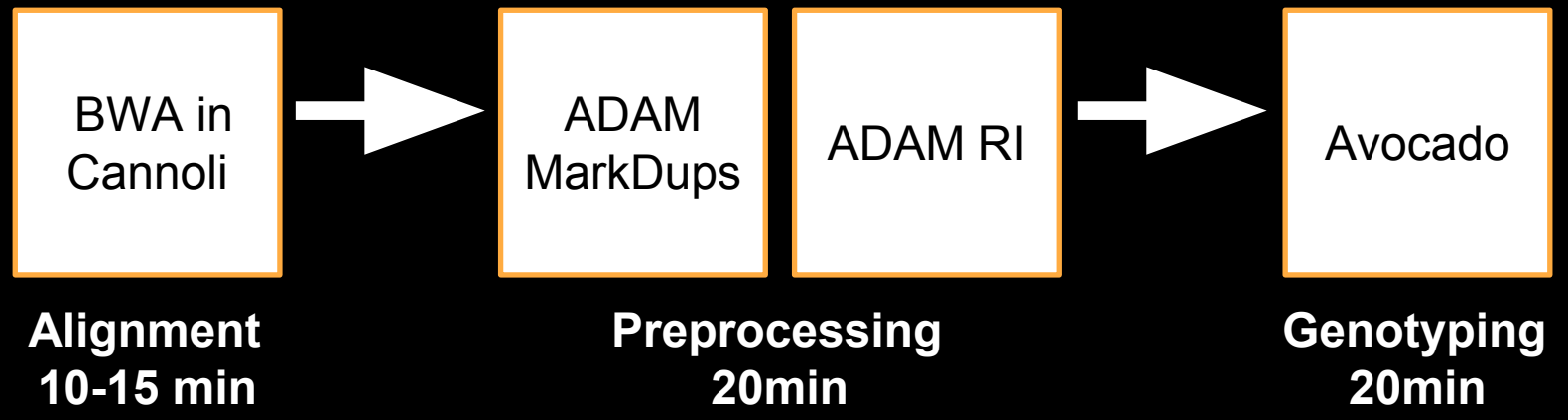
# Spark is a platform for end-to-end discovery




Easy-to-deploy, highly scalable tools, in an integrated environment allow you to answer larger questions faster!

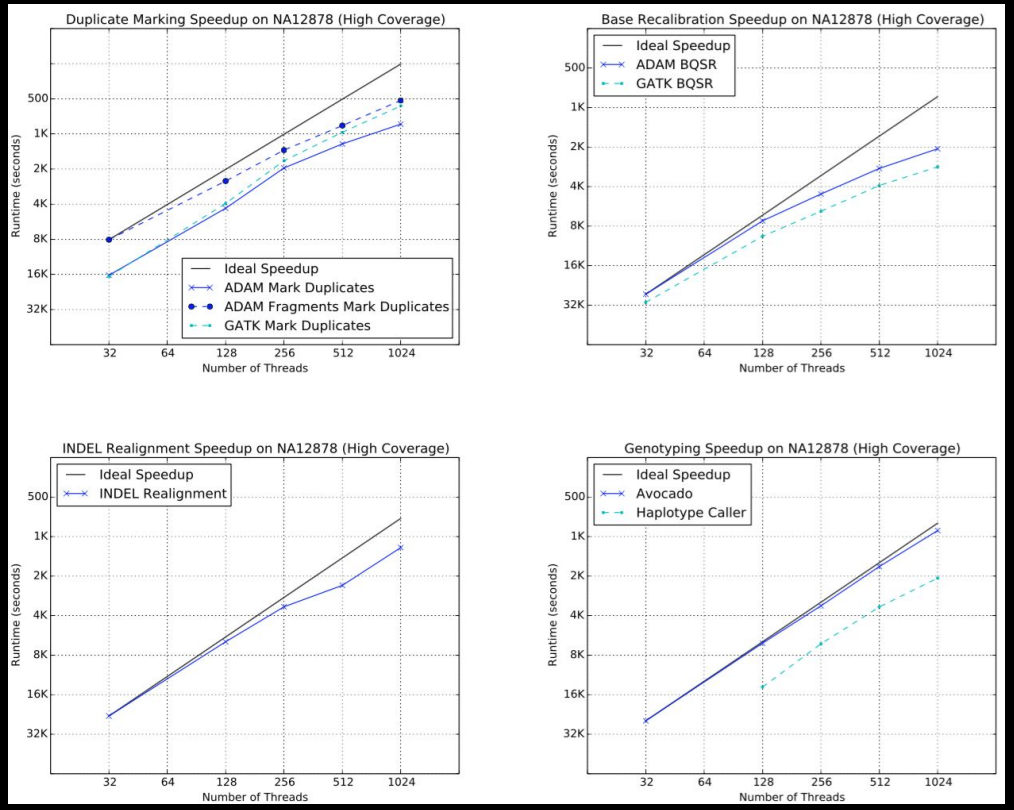
# Why use Spark?
# Variant calling

# End-to-end variant calling on Spark




- ADAM preprocessing stages are highly concordant with GATK, Avocado uses a biallelic model, built on ADAM APIs
- Achieves >99% precision/recall for SNPs, 95-97% for INDELs using 58x NA12878 WGS vs. GIAB truth set

# Variant calling shows strong scaling




- Supports joint genotyping as well: 5M variants by 800G gVCF reference models in ~6hrs

# How do I do bioinformatics on Spark?

# Bioinformatics on Spark

- Most alignment-based bioinformatics analyses map well to Spark:
  - Map genomic data into a schema
  - Use Spark SQL, ADAM, or Hail for overlap and aggregate queries
  - Run standalone tools using Pipe API
- First, we'll walk through some of the APIs, and then we'll walk through a few use cases

# Representing genomic data with a schema

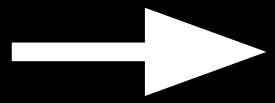
- Widely used technique across best-practice Spark genomics tools:
  - ADAM provides schemas for reads, variants/genotypes, and generic genomic features
  - Hail provides schemas for variants/genotypes and some feature formats
- We also see customers develop their own schemas:
  - Corresponding to specific sequencing methodologies (e.g., cfDNA-seq)
  - Corresponding to specific annotation databases

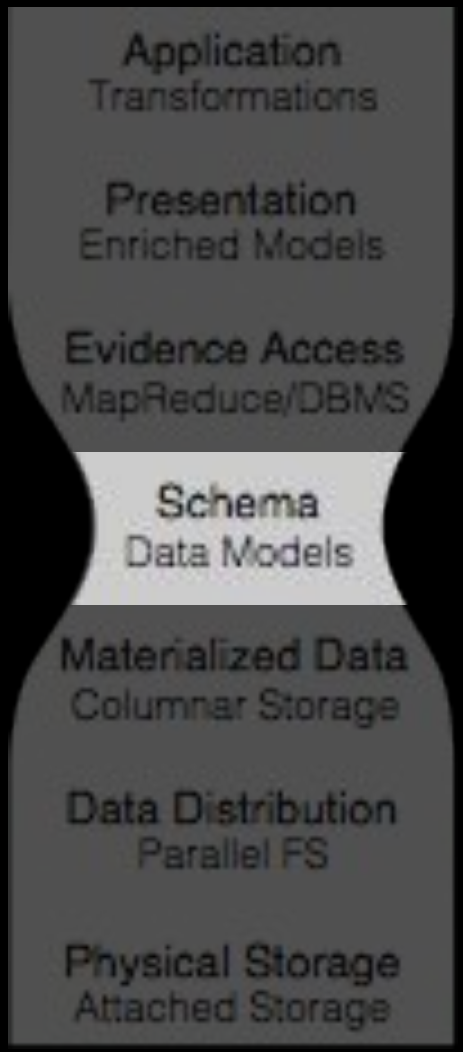
# Representing genomic data with a schema

```
record AlignmentRecord {
    union { null, string } contigName = 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 } mateContig = null;
}
```

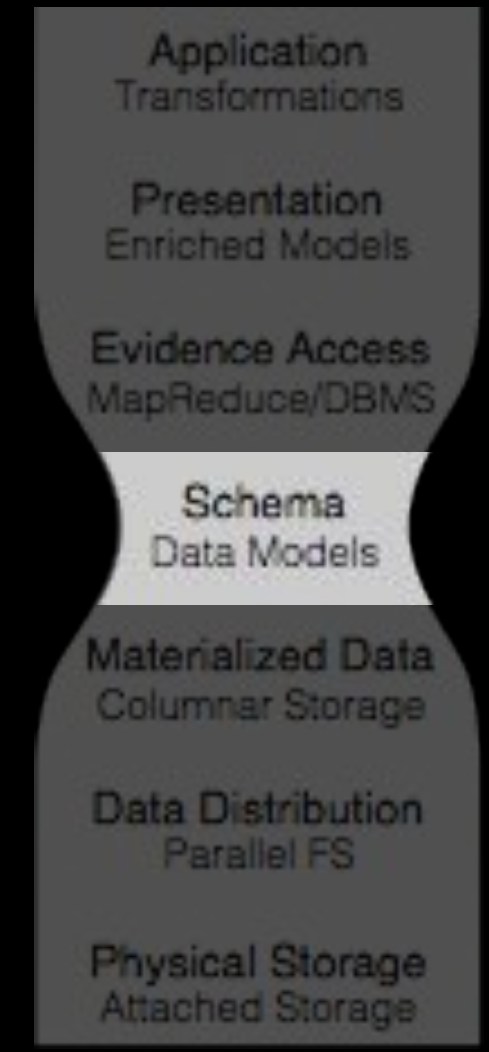
# Representing genomic data with a schema

```
record AlignmentRecord {
    union { null, string } contigName = 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 } mateContig = null;
}
```



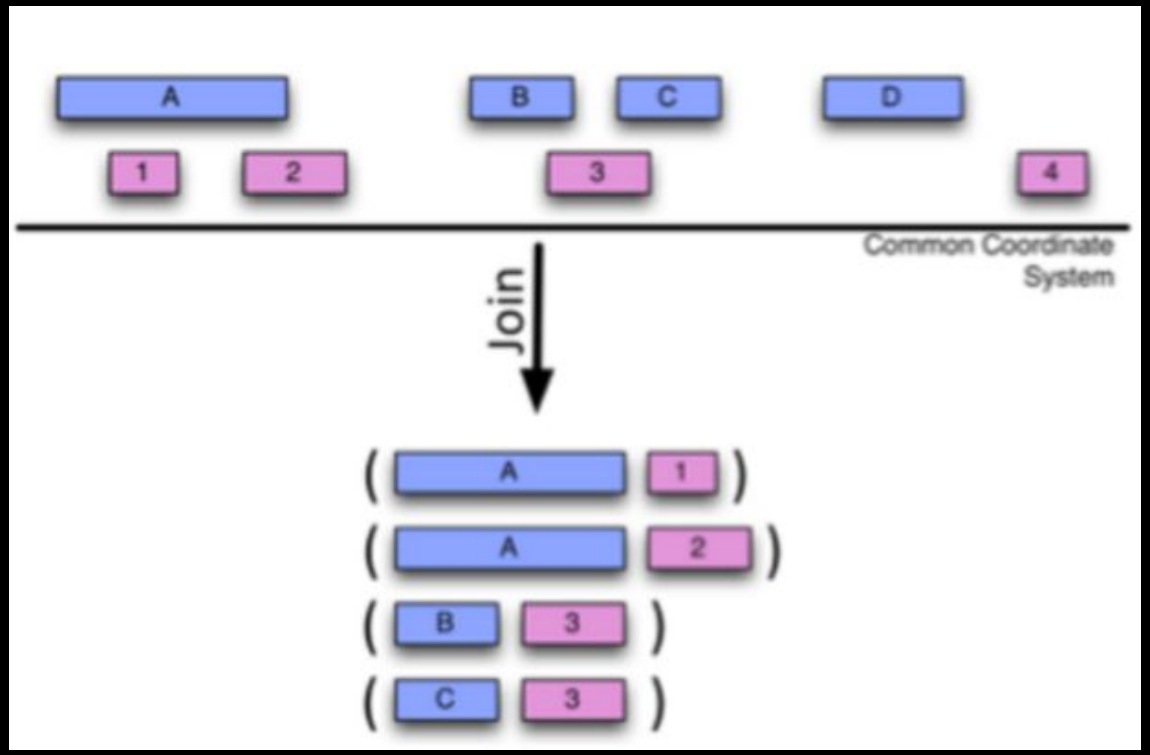


Having a stack makes it easy to accelerate genomic queries



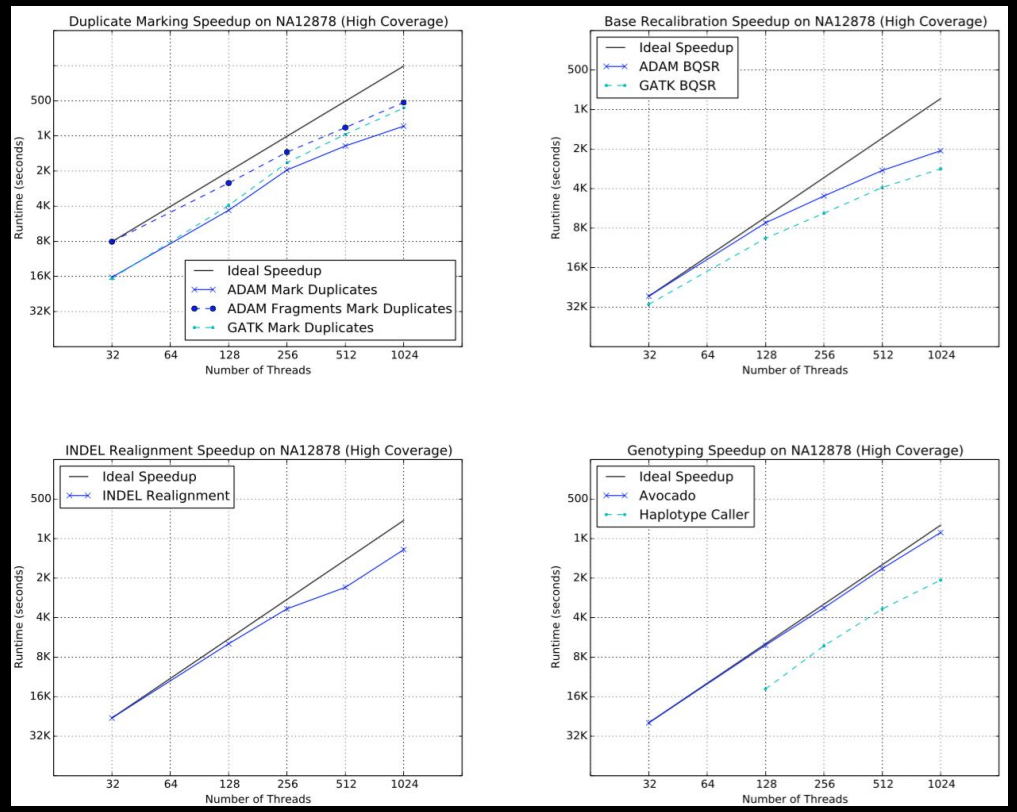
# ...while also providing higher level abstractions

- Eliminates need to use "genome walker":
  - Use region join for overlap computation
  - Use groupBy functions from Spark SQL to process features aligned at a genomic coordinate point



# Why use Spark?

# Variant calling shows strong scaling




- Supports joint genotyping as well: 5M variants by 800G gVCF reference models in ~6hrs

# Rapid case studies from the wild

- Rapid variation query:
  - Interactive query against variation + phenotype data from >100k WES → support cubed drill-down across complex G2P dataset
- Direct query against read data:
  - Customer with 10,000 WGS sequences has generated SV breakpoint calls on individual samples, use Spark + ML to generate cleaned CNV calls
- Scaling out bioinformatics workflows:
  - Can use Spark to rapidly accelerate common analyses → align in 10 min
  - Or… scale those analyses out → use Spark to joint genotype 5k samples in a single shot

# Spark is a platform for end-to-end discovery

- Spark allows you to program across large clusters of computers
  - Drop analysis time for complex tasks (e.g. variant calling from 100 hours to <1hr)
- There's a groundswell of support for Spark for bioinformatics:
  - ADAM for general-purpose genomics query
  - ADAM and GATK4 for variant calling
  - Hail, VariantSpark, SparkSeq for variant analysis

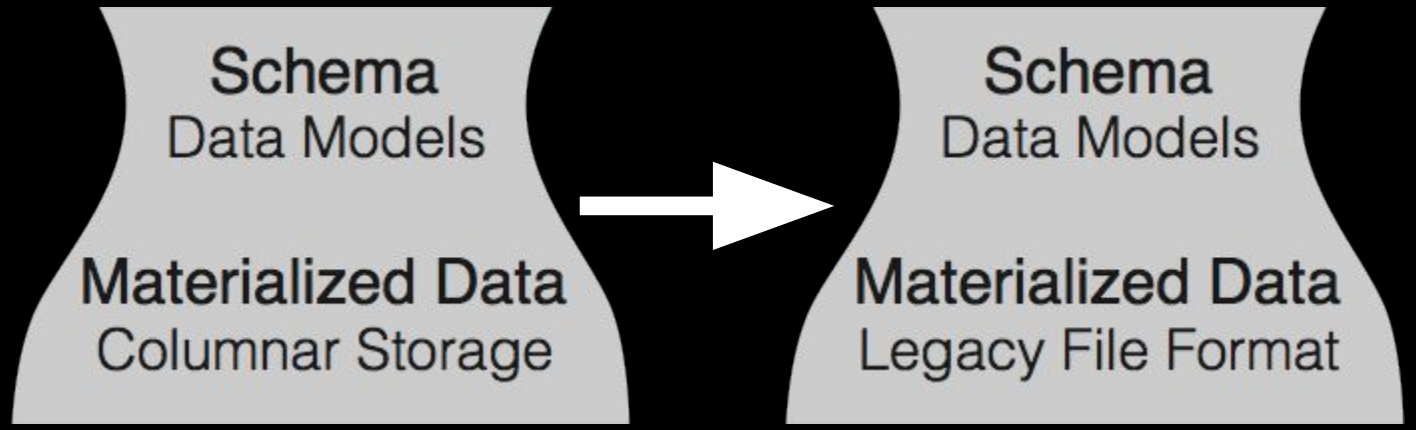
# Spark is a platform for end-to-end discovery

- Spark allows you to program across large clusters of computers
  - Drop analysis time for complex tasks (e.g. variant calling from 100 hours to <1hr)
- There's a groundswell of support for Spark for bioinformatics:
  - ADAM for general-purpose genomics query
  - ADAM and GATK4 for variant calling
  - Hail, VariantSpark, SparkSeq for variant analysis

Interested? Try Databricks Community Edition:
https://databricks.com/try-databricks

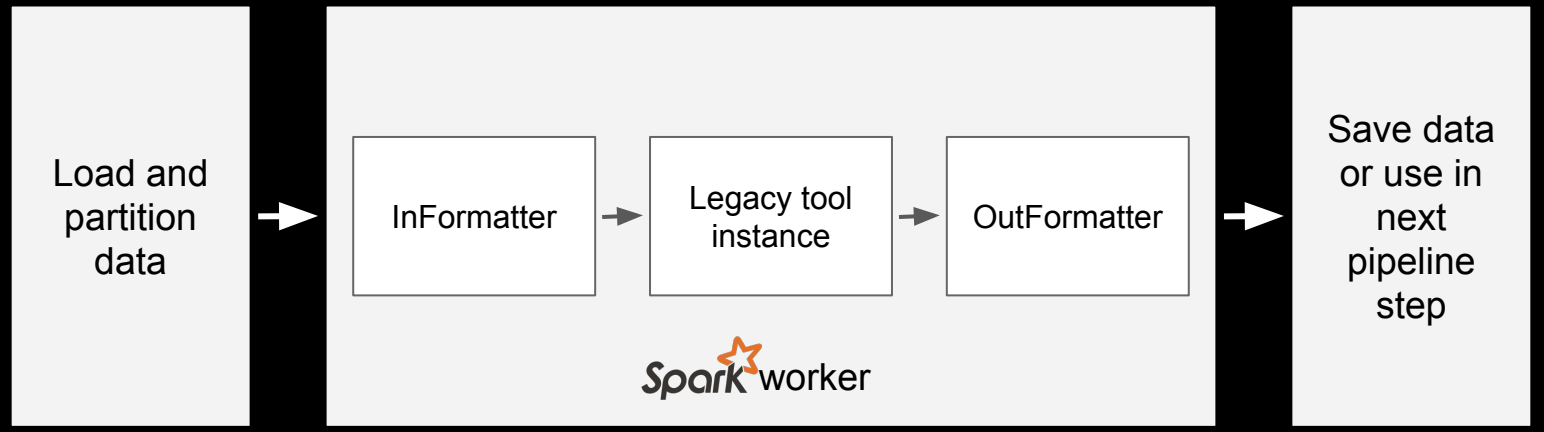
# Extra Slides

# Schemas simplify file format support




- Reads from Parquet, SAM, BAM, CRAM, FASTQ, FASTA
- Variants from Parquet, HBase, VCF, BCF
- Features from Parquet, BED, GTF, GFF, NarrowPeak, IntervalList

# Reuse tools with the Pipe API




- Support common file formats (SAM/BAM/CRAM, VCF, BED/GTF/GFF/NarrowPeak, FASTQ)
- Use pipe to drop single sample alignment time with BWA to <20 minutes