ADAM

Introduction

ADAM is a genomics analysis platform with specialized file formats built using Apache Avro, Apache Spark and Apache Parquet. Apache 2 licensed. Some quick links:

- Follow our Twitter account.
- Chat with ADAM developers in Gitter.
- Join our mailing list.
- Checkout the current build status.
- Download official releases.
- View our software artifacts on Maven Central (...including snapshots).
- Look at our CHANGES file.

Hello World: Counting K-mers

Here's an example ADAM CLI command that will count 10-mers in a test `.sam` file that lives in this repository:

```
$ adam-submit count_kmers /tmp/small.adam /tmp/kmers.adam 10
$ head /tmp/kmers.adam/part-*
(AATTGGCACT,1)
(TTCCGATTTT,1)
(GAGCAGCCTT,1)
(CCTGCTGTAT,1)
(TTTTAAGGTT,1)
(GGCCAGGACT,1)
(GCAGTCCCTC,1)
(AACTTTGAAT,1)
(GATGACGTGG,1)
(CTGTCCCTGT,1)
```

More than K-mer Counting

ADAM does much more than just k-mer counting. Running the ADAM CLI without arguments or with `--help` will display available commands, e.g.

```
$ adam-submit

      e            888~-_             e                e    e
     d8b           888   \           d8b              d8b  d8b
    /Y88b          888    |         /Y88b            d888bdY88b
   /  Y88b         888    |        /  Y88b          / Y88Y Y888b
  /____Y88b        888   /        /____Y88b        /   YY   Y888b
 /      Y88b       888_-~        /      Y88b      /          Y888b

Choose one of the following commands:

ADAM ACTIONS
               depth : Calculate the depth from a given ADAM file, at each vari
ant in a VCF
         count_kmers : Counts the k-mers/q-mers from a read dataset.
   count_contig_kmers : Counts the k-mers/q-mers from a read dataset.
           transform : Convert SAM/BAM to ADAM format and optionally perform re
ad pre-processing transformations
          adam2fastq : Convert BAM to FASTQ files
              plugin : Executes an ADAMPlugin
             flatten : Convert a ADAM format file to a version with a flattened
schema, suitable for querying with tools like Impala

CONVERSION OPERATIONS
             vcf2adam : Convert a VCF file to the corresponding ADAM format
            anno2adam : Convert a annotation file (in VCF format) to the corresp
onding ADAM format
             adam2vcf : Convert an ADAM variant to the VCF ADAM format
           fasta2adam : Converts a text FASTA sequence file into an ADAMNucleoti
deContig Parquet file which represents assembled sequences.
        features2adam : Convert a file with sequence features into corresponding
ADAM format
```

```
          wigfix2bed : Locally convert a wigFix file to BED format

PRINT
               print : Print an ADAM formatted file
         print_genes : Load a GTF file containing gene annotations and print th
e corresponding gene models
            flagstat : Print statistics on reads in an ADAM file (similar to sa
mtools flagstat)
          print_tags : Prints the values and counts of all tags in a set of rec
ords
            listdict : Print the contents of an ADAM sequence dictionary
         allelecount : Calculate Allele frequencies
           buildinfo : Display build information (use this for bug reports)
                view : View certain reads from an alignment-record file.
```

You can learn more about a command, by calling it without arguments or with `--help`, e.g.

```
$ adam-submit transform
Argument "INPUT" is required
 INPUT                                                    : The ADAM, BA
M or SAM file to apply the transforms to
 OUTPUT                                                   : Location to
write the transformed data in ADAM/Parquet format
 -coalesce N                                              : Set the numb
er of partitions written to the ADAM output directory
 -dump_observations VAL                                   : Local path t
o dump BQSR observations to. Outputs CSV format.
 -force_load_bam                                          : Forces Trans
form to load from BAM/SAM.
 -force_load_fastq                                        : Forces Trans
form to load from unpaired FASTQ.
 -force_load_ifastq                                       : Forces Trans
form to load from interleaved FASTQ.
 -force_load_parquet                                      : Forces Trans
```

```
 -force_load_parquet                                                : Force trans
form to load from Parquet.
 -h (-help, --help, -?)                                             : Print help
 -known_indels VAL                                                  : VCF file inc
luding locations of known INDELs. If none is provided, default
                                                                      consensus mo
del will be used.
 -known_snps VAL                                                    : Sites-only V
CF giving location of known SNPs
 -log_odds_threshold N                                              : The log-odds
threshold for accepting a realignment. Default value is 5.0.
 -mark_duplicate_reads                                              : Mark duplica
te reads
 -max_consensus_number N                                            : The maximum
number of consensus to try realigning a target region to. Default
                                                                      value is 30.
 -max_indel_size N                                                  : The maximum
length of an INDEL to realign to. Default value is 500.
 -max_target_size N                                                 : The maximum
length of a target region to attempt realigning. Default length is
                                                                      3000.
 -parquet_block_size N                                              : Parquet bloc
k size (default = 128mb)
 -parquet_compression_codec [UNCOMPRESSED | SNAPPY | GZIP | LZO]    : Parquet comp
ression codec
 -parquet_disable_dictionary                                        : Disable dict
ionary encoding
 -parquet_logging_level VAL                                         : Parquet logg
ing level (default = severe)
 -parquet_page_size N                                               : Parquet page
size (default = 1mb)
 -print_metrics                                                     : Print metric
s to the log on completion
 -realign_indels                                                    : Locally real
ign indels present in reads.
 -recalibrate_base_qualities                                        : Recalibrate
```

```
the base quality scores (ILLUMINA only)
 -repartition N                                                  : Set the numb
er of partitions to map data to
 -sort_fastq_output                                              : Sets whether
to sort the FASTQ output, if saving as FASTQ. False by default.
                                                                   Ignored if n
ot saving as FASTQ.
 -sort_reads                                                     : Sort the rea
ds by referenceId and read position
```

The ADAM `transform` command allows you to mark duplicates, run base quality score recalibration (BQSR) and other pre-processing steps on your data.

Getting Started

Installation

Binary Distributions

Bundled release binaries can be found on our releases page.

Building from Source

You will need to have Maven installed in order to build ADAM.

> ***Note:*** *The default configuration is for Hadoop 2.2.0. If building against a different version of Hadoop, please edit the build configuration in the* `<properties>` *section of the* `pom.xml` *file.*

```
$ git clone https://github.com/bigdatagenomics/adam.git
$ cd adam
$ export MAVEN_OPTS="-Xmx512m -XX:MaxPermSize=256m"
$ mvn clean package -DskipTests
...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 9.647s
[INFO] Finished at: Thu May 23 15:50:42 PDT 2013
[INFO] Final Memory: 19M/81M
[INFO] ------------------------------------------------------------------------
```

You might want to take a peek at the `scripts/jenkins-test` script and give it a run. It will fetch a mouse chromosome, encode it to ADAM reads and pileups, run flagstat, etc. We use this script to test that ADAM is working correctly.

Installing Spark

You'll need to have a Spark release on your system and the `$SPARK_HOME` environment variable pointing at it; prebuilt binaries can be downloaded from the Spark website. Currently, our continuous builds use Spark 1.1.0 built against Hadoop 2.3 (CDH5), but any more recent Spark distribution should also work.

Helpful Aliases

You might want to add the following to your `.bashrc` to make running ADAM easier:

```
alias adam-submit="${ADAM_HOME}/bin/adam-submit"
alias adam-shell="${ADAM_HOME}/bin/adam-shell"
```

`$ADAM_HOME` should be the path to a binary release or a clone of this repository on your local filesystem.

These aliases call scripts that wrap the `spark-submit` and `spark-shell` commands to set up ADAM.Once they are in place, you can run adam by simply typing `adam-submit` at the command line, as demonstrated above.

Running ADAM

Now you can try running some simple ADAM commands:

`transform`

Make your first `.adam` file like this:

```
adam-submit transform $ADAM_HOME/adam-core/src/test/resources/small.sam /tmp/sm
all.adam
```

If you didn't obtain your copy of adam from github, you can grab `small.sam` here.

`flagstat`

Once you have data converted to ADAM, you can gather statistics from the ADAM file using `flagstat`. This command will output stats identically to the samtools `flagstat` command.

If you followed along above, now try gathering some statistics:

```
$ adam-submit flagstat /tmp/small.adam
20 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 primary duplicates
0 + 0 primary duplicates - both read and mate mapped
0 + 0 primary duplicates - only read mapped
0 + 0 primary duplicates - cross chromosome
0 + 0 secondary duplicates
0 + 0 secondary duplicates - both read and mate mapped
0 + 0 secondary duplicates - only read mapped
0 + 0 secondary duplicates - cross chromosome
20 + 0 mapped (100.00%:0.00%)
0 + 0 paired in sequencing
0 + 0 read1
0 + 0 read2
0 + 0 properly paired (0.00%:0.00%)
0 + 0 with itself and mate mapped
0 + 0 singletons (0.00%:0.00%)
0 + 0 with mate mapped to a different chr
0 + 0 with mate mapped to a different chr (mapQ>=5)
```

In practice, you'll find that the ADAM `flagstat` command takes orders of magnitude less time than samtools to compute these statistics. For example, on a MacBook Pro `flagstat NA12878_chr20.bam` took 17 seconds to run while `samtools flagstat NA12878_chr20.bam` took 55 seconds. On larger files, the difference in speed is even more dramatic. ADAM is faster because it's multi-threaded and distributed and uses a columnar storage format (with a projected schema that only materializes the read flags instead of the whole read).

`adam-shell`

The `adam-shell` command opens an interpreter that you can run ad-hoc ADAM commands in.

For example, the following code snippet will generate a result similar to the k-mer-counting example above, but with the k-mers sorted in descending order of their number of

occurrences. To use this, save the code snippet as `kmer.scala` and run `adam-shell -i kmer.scala`.

`kmer.scala`

```
import org.bdgenomics.adam.rdd.ADAMContext
import org.bdgenomics.adam.projections.{AlignmentRecordField, Projection}

val ac = new ADAMContext(sc)
// Load alignments from disk
val reads = ac.loadAlignments(
  "/data/NA21144.chrom11.ILLUMINA.adam",
  projection = Some(
    Projection(
      AlignmentRecordField.sequence,
      AlignmentRecordField.readMapped,
      AlignmentRecordField.mapq
    )
  )
)

// Generate, count and sort 21-mers
val kmers =
  reads
    .flatMap(_.getSequence.sliding(21).map(k => (k, 1L)))
    .reduceByKey(_ + _)
    .map(_.swap)
    .sortByKey(ascending = false)

// Print the top 10 most common 21-mers
kmers.take(10).foreach(println)
```

`adam-shell -i kmer.scala`

```
$ adam-shell -i kmer.scala
…
(121771,TTTTTTTTTTTTTTTTTTTT)
(44317,ACACACACACACACACACACA)
(44023,TGTGTGTGTGTGTGTGTGTGT)
(42474,CACACACACACACACACACAC)
(42095,GTGTGTGTGTGTGTGTGTGTG)
(33797,TAATCCCAGCACTTTGGGAGG)
(33081,AATCCCAGCACTTTGGGAGGC)
(32775,TGTAATCCCAGCACTTTGGGA)
(32484,CCTCCCAAAGTGCTGGGATTA)
…
```

Running on a cluster

The `adam-submit` and `adam-shell` commands can also be used to submit ADAM jobs to a Spark cluster, or to run ADAM interactively. Cluster mode can be enabled by passing the same flags you'd pass to Spark, e.g. `--master yarn --deploy-mode client`.

Running Plugins

ADAM allows users to create plugins via the ADAMPlugin trait. These plugins are then imported using the Java classpath at runtime. To add to the classpath when using appassembler, use the `$CLASSPATH_PREFIX` environment variable. For an example of how to use the plugin interface, please see the adam-plugins repo.

Under the Hood

ADAM relies on several open-source technologies to make genomic analyses fast and massively parallelizable…

Apache Spark

Apache Spark allows developers to write algorithms in succinct code that can run fast locally,

on an in-house cluster or on Amazon, Google or Microsoft clouds.

Apache Parquet

Apache Parquet is a columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language.

- Parquet compresses legacy genomic formats using standard columnar techniques (e.g. RLE, dictionary encoding). ADAM files are typically ~20% smaller than compressed BAM files.
- Parquet integrates with:
    - **Query engines**: Hive, Impala, HAWQ, IBM Big SQL, Drill, Tajo, Pig, Presto
    - **Frameworks**: Spark, MapReduce, Cascading, Crunch, Scalding, Kite
    - **Data models**: Avro, Thrift, ProtocolBuffers, POJOs
- Parquet is simply a file format which makes it easy to sync and share data using tools like `distcp`, `rsync`, etc
- Parquet provides a command-line tool, `parquet.hadoop.PrintFooter`, which reports useful compression statistics

In the counting k-mers example above, you can see there is a defined *predicate* and *projection*. The *predicate* allows rapid filtering of rows while a *projection* allows you to efficiently materialize only specific columns for analysis. For this k-mer counting example, we filter out any records that are not mapped or have a `MAPQ` less than 20 using a `predicate` and only materialize the `Sequence`, `ReadMapped` flag and `MAPQ` columns and skip over all other fields like `Reference` or `Start` position, e.g.

Sequence	**ReadMapped**	**MAPQ**	~~**Reference**~~	~~**Start**~~	**...**
~~GGTCCAT~~	~~false~~	-	~~chrom1~~	-	...
TACTGAA	true	30	~~chrom1~~	~~34232~~	...
~~TTGAATG~~	~~true~~	~~17~~	~~chrom1~~	~~309403~~	...

Apache Avro

- Apache Avro is a data serialization system.
- All Big Data Genomics schemas are published at https://github.com/bigdatagenomics/bdg-formats.
- Having explicit schemas and self-describing data makes integrating, sharing and evolving formats easier.

Our Avro schemas are directly converted into source code using Avro tools. Avro supports a number of computer languages. ADAM uses Java; you could just as easily use this Avro IDL description as the basis for a Python project. Avro currently supports c, c++, csharp, java, javascript, php, python and ruby.

Downstream Applications

There are a number of projects built on ADAM, e.g.

- RNAdam provides an RNA pipeline on top of ADAM with isoform quantification and fusion transcription detection
- Avocado is a variant caller built on top of ADAM for germline and somatic calling
- PacMin is an assembler for PacBio reads
- A `Mutect` port is nearly feature complete
- Read error correction
- a graphing and genome visualization library
- BDG-Services is a library for accessing a running Spark cluster through web-services or a Thrift- interface
- Short read assembly
- Variant filtration (train model via `MLlib`)

License

ADAM is released under an Apache 2.0 license.