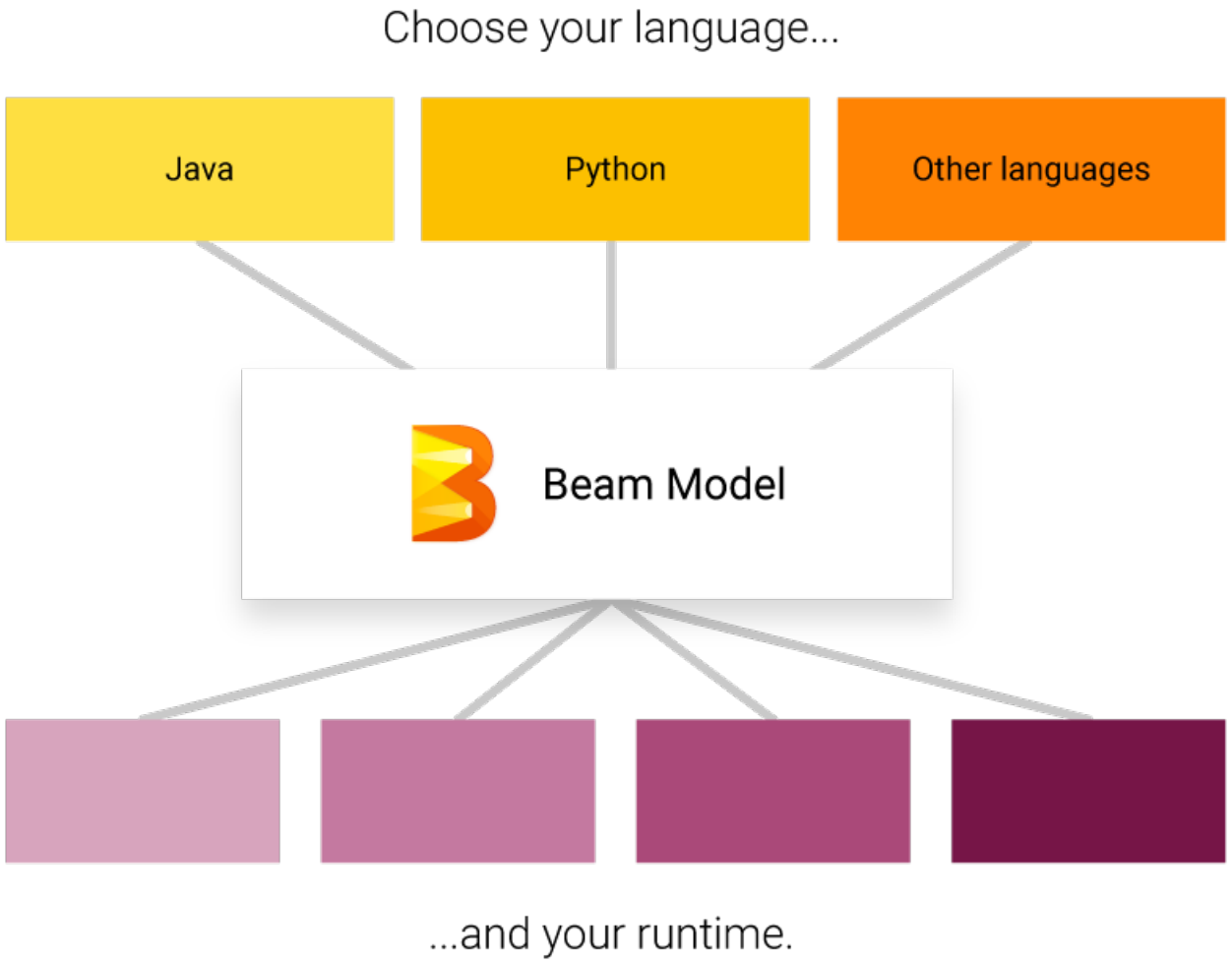
HBase on Beam

Apache Beam

- Apache Beam is an open source, unified programming model for defining both batch and streaming data-parallel processing pipelines.
- It was initialized and contributed by Google.
- Published the first stable release on May 17, 2017.

Apache Beam



https://beam.apache.org/images/beam_architecture.png

Apache Beam

- A unified model for batch and streaming applications.
- Runners for famous open-source batch and streaming engines, for instance Spark and Flink.
- Multi-languages are available for end users to build their own pipelines, now Java and Python are supported.
- Implement once, run almost everywhere.

Apache Beam

- Pipeline: The processing pipeline which includes data input, transform and output.
- PCollection: The representation for both bounded and unbounded data
- Transform
  - ParDo
  - GroupByKey
  - Combine
  - Flatten
  - …

Data Sources

- In-memory data: Array, Collection, Map
- Text
- HDFS
- Kafka
- HBase
- …

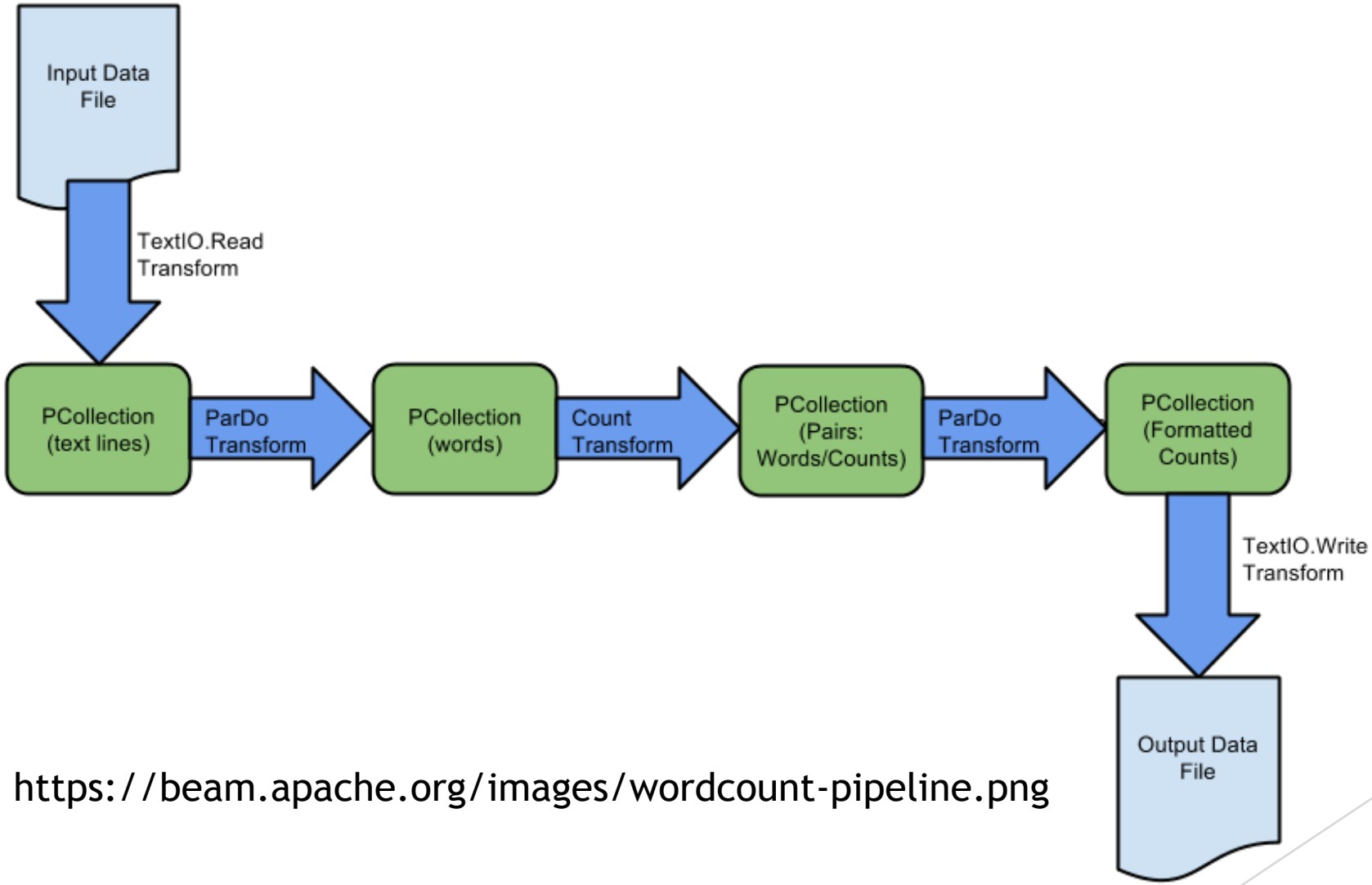
Windowing

- Fixed time windows
- Sliding time windows
- Session windows
- Single global window

Serialization

- Every Transform must be serializable!
  - CustomCoder
    - Register coder for classes
    - Register coder for the output of transform
  - Serializable

Example: Count the Words



https://beam.apache.org/images/wordcount-pipeline.png

Examples: Count the Words

```
Pipeline pipeline = Pipeline.create(options);
PCollection<String> inputs = pipeline.apply(TextIO.read().from("A local file"));
inputs.apply("ExtractWords", ParDo.of(new DoFn<String, String>() {
  @ProcessElement
  public void processElement(ProcessContext c, BoundedWindow window) throws Exception {
    for (String e : c.element().split(" ")) {
      c.output(e);
    }
  }
})).apply(Count.<String> perElement())
    .apply("FormatResults", ParDo.of(new DoFn<KV<String, Long>, String>() {
      @ProcessElement
      public void processElement(ProcessContext c, BoundedWindow window) throws Exception {
        KV<String, Long> kv = c.element();
        c.output(kv.getKey() + ":" + kv.getValue());
      }
    })).apply(TextIO.write().to("Another local file"));
pipeline.run().waitUntilFinish();
```

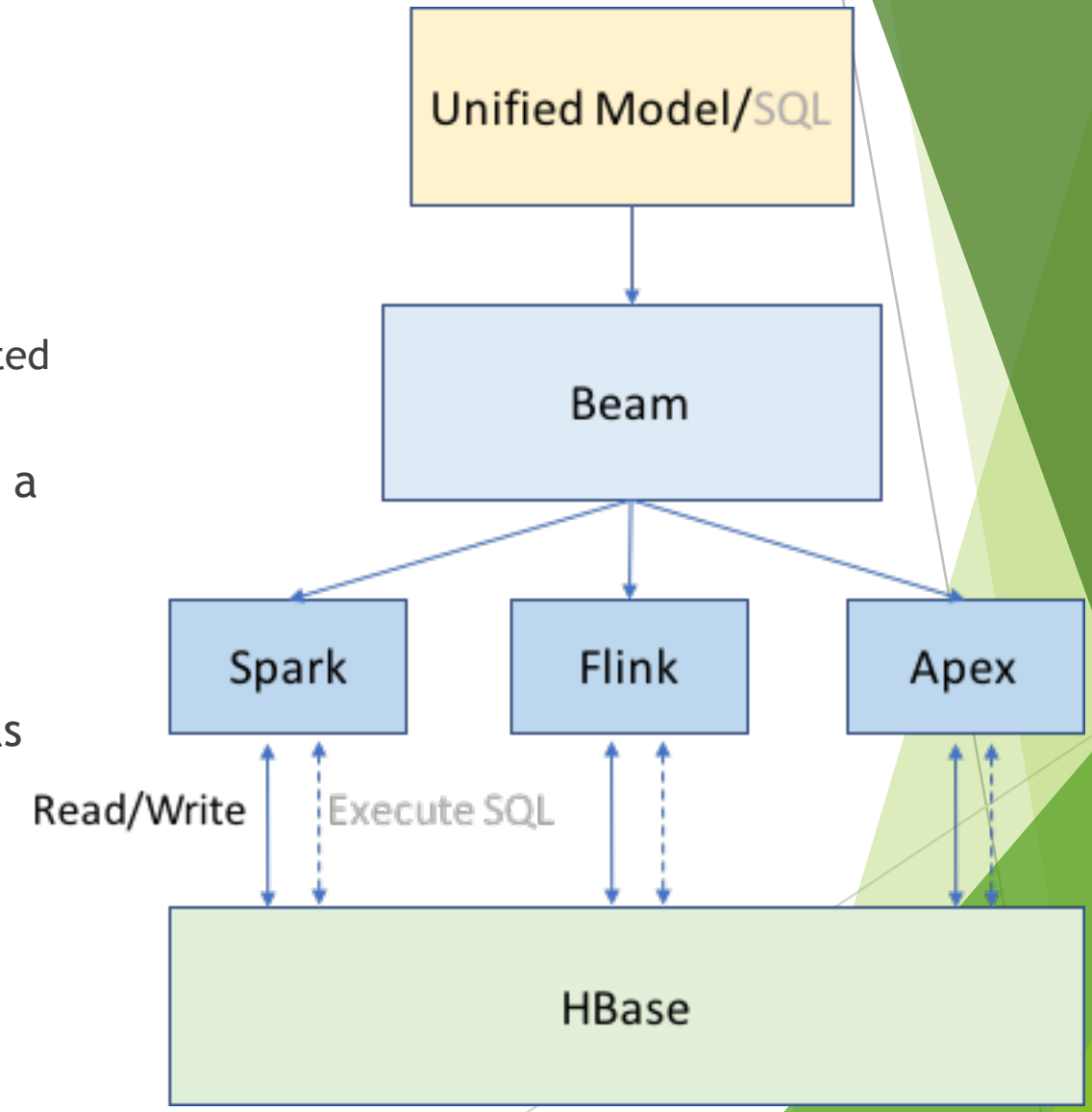
Capability Matrix

	Beam Model	Google Cloud Dataflow	Apache Flink	Apache Spark	Apache Apex	Apache Gearpump
ParDo	✓	✓	✓	✓	✓	✓
GroupByKey	✓	✓	✓	~	✓	✓
Flatten	✓	✓	✓	✓	✓	✓
Combine	✓	✓	✓	✓	✓	✓
Composite Transforms	✓	~	~	~	~	~
Side Inputs	✓	✓	✓	✓	✓	✓
Source API	✓	✓	✓	✓	✓	✓
Splittable DoFn	~	~	×	×	×	×
Metrics	~	~	~	~	×	×
Stateful Processing	✓	~	~	×	×	×

https://beam.apache.org/documentation/runners/capability-matrix/

HBase + Beam

- Inspired by HBase + Spark
  - Similar functions, Beam SQL is not supported yet.
- Use HBase as a bounded data source, and a target data store in both batch and streaming applications
- Customized Transforms for HBase bulk operations, and HBasePipelineFunctions as the entry to start the pipeline.



Operations

- Operations for both batch and streaming manners
  - Scan (Already implemented in Beam)
  - BulkGet
  - BulkPut
  - BulkDelete
  - MapPartitions
  - ForeachPartition
  - BulkLoad
  - BulkLoadThinRows

Examples: Scan

- Read data from HBase table by scan

```
Read read = HBaseIO.read().withConfiguration(conf).withTableId(tableName).withKeyRange(startRow,
  stopRow);
PCollection<Result> results = p.apply("Read", read);
```

Examples: BulkGet

- Implement MakeFunctions to convert input to Get, and convert Result to output

```
PCollection<byte[]> results = HBasePipelineFunctions.bulkGet(conf, tableNameAsString, 10,
  inputs, new MakeFunction<byte[], Get>() {

    @Override
    public Get make(byte[] input) {
      return new Get(input);
    }
  }, null, new MakeFunction<Result, byte[]>() {

    @Override
    public byte[] make(Result input) {
      return input.getRow();
    }
  }, ByteArrayCoder.of());
```

Examples: BulkPut

- Implement MakeFunction to convert input to Put.

```
PCollection<String> inputs =
    pipeline.apply("createDataset", Create.of(cellStrings));
HBasePipelineFunctions.bulkPut(conf, tableNameAsString, inputs,
  new MakeFunction<String, Mutation>() {

    @Override
    public Mutation make(String input) {
      String[] strs = input.split(" ");
      if (strs.length == 2) {
        Put put = new Put(Bytes.toBytes(strs[0]));
        put.addColumn(FAMILY, QUALIFIER, Bytes.toBytes(strs[1]));
        return put;
      } else {
        return null;
      }
    }
  });
```

Examples: BulkDelete

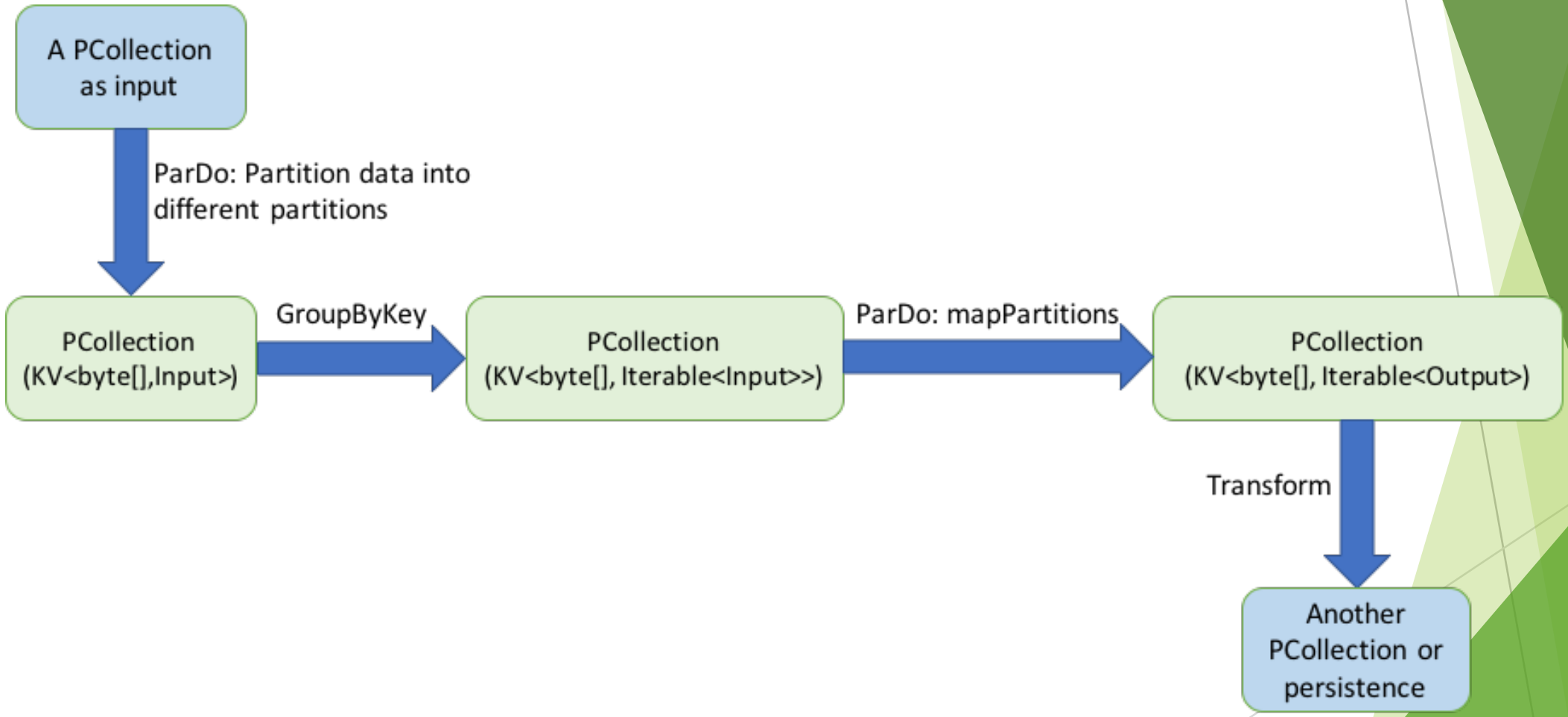
- Implement MakeFunction to convert input to Delete.

```
PCollection<byte[]> inputs =
    pipeline.apply("createDataset", Create.of(deleteRows));
HBasePipelineFunctions.bulkDelete(conf, tableNameAsString, inputs,
  new MakeFunction<byte[], Mutation>() {

    @Override
    public Mutation make(byte[] input) {
      return new Delete(input);
    }
  });
```

Examples: MapPartitions

```
PCollection<String> inputs =
    pipeline.apply("createDataset" + UUID.randomUUID().toString(), Create.of(cellStrings));
Configuration tempConf = new Configuration(conf);
tempConf.set("beam.test.tablename", tableNameAsString);
HBasePipelineFunctions.mapPartitions(tempConf, inputs, null,
  new MapPartitionsFunc<String, Result>() {

    @Override
    public Iterable<Result> execute(Configuration conf, Connection conn,
        Iterable<String> partition) throws IOException {
      String tableName = conf.get("beam.test.tablename");
      List<Get> gets = new ArrayList<>();
      // pass the rows in the partition to the gets
      return gets.isEmpty() ? Collections.emptyList()
          : Arrays.asList(conn.getTable(TableName.valueOf(tableName)).get(gets));
    }
  }, HBaseResultCoder.of());
```

Examples: MapPartitions



Examples: ForeachPartition

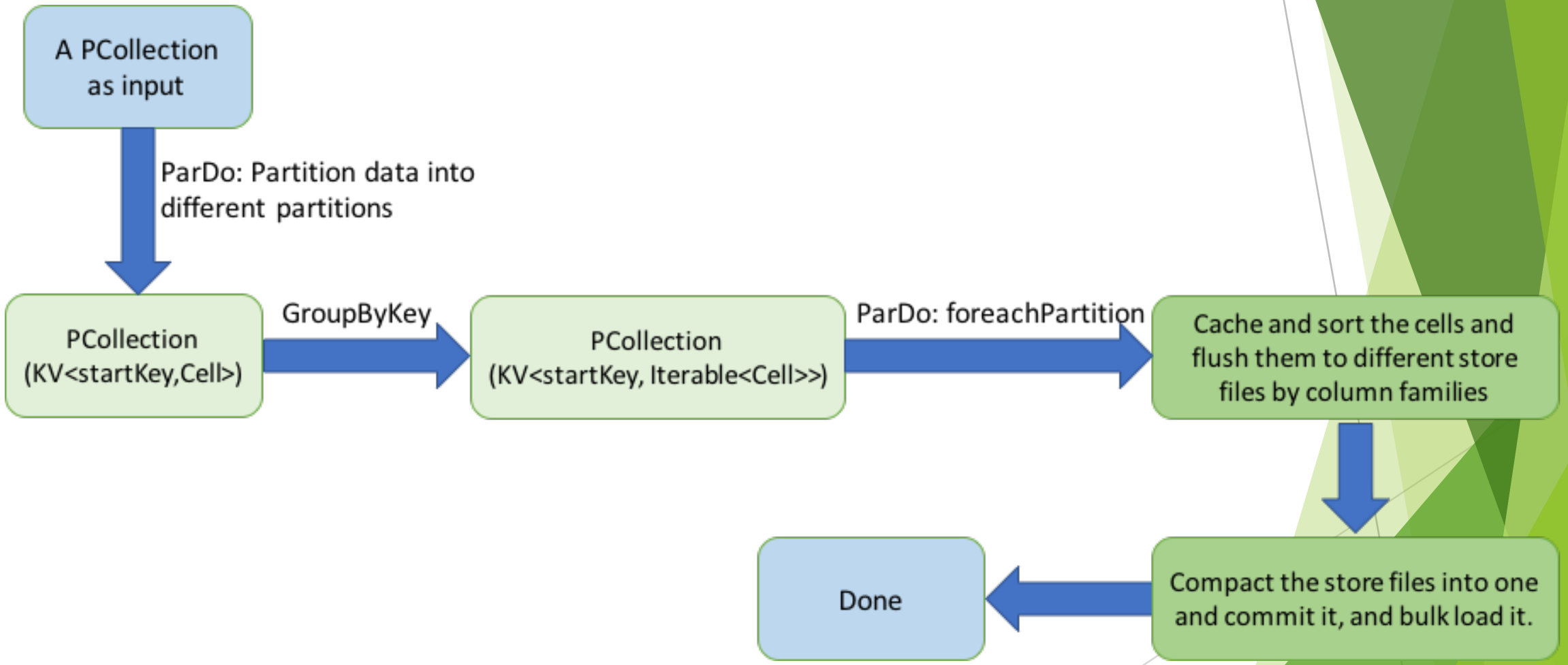
```
PCollection<String> inputs =
    pipeline.apply("createDataset" + UUID.randomUUID().toString(), Create.of(cellStrings));
Configuration tempConf = new Configuration(conf);
tempConf.set("beam.test.tablename", tableNameAsString);
HBasePipelineFunctions.foreachPartition(tempConf, inputs, null,
  new ForeachPartitionFunc<String>() {

    @Override
  public void execute(Configuration conf, Connection conn, Iterable<String> partition)
      throws IOException {
    String tableName = conf.get("beam.test.tablename");
    BufferedMutator mutator = conn.getBufferedMutator(TableName.valueOf(tableName));
    try {
      // extract data from the partition and do the mutation
      mutator.flush();
    } finally {
      mutator.close();
    }
  }
});
```

Examples: BulkLoad

- Implement MakeFunction to convert each input into a Cell.

```
HBasePipelineFunctions.bulkLoad(conf, tableNameAsString, inputs, stagingPath.toUri(),
  tmpPath.toUri(), new MakeFunction<String, Cell>() {

    @Override
    public Cell make(String input) {
      String[] strs = input.split(" ");
      if (strs.length == 4) {
        return new KeyValue(Bytes.toBytes(strs[0]), Bytes.toBytes(strs[1]),
            Bytes.toBytes(strs[2]), EnvironmentEdgeManager.currentTime(),
            Bytes.toBytes(strs[3]));
      }
      return null;
    }
  });
```

Examples: BulkLoad

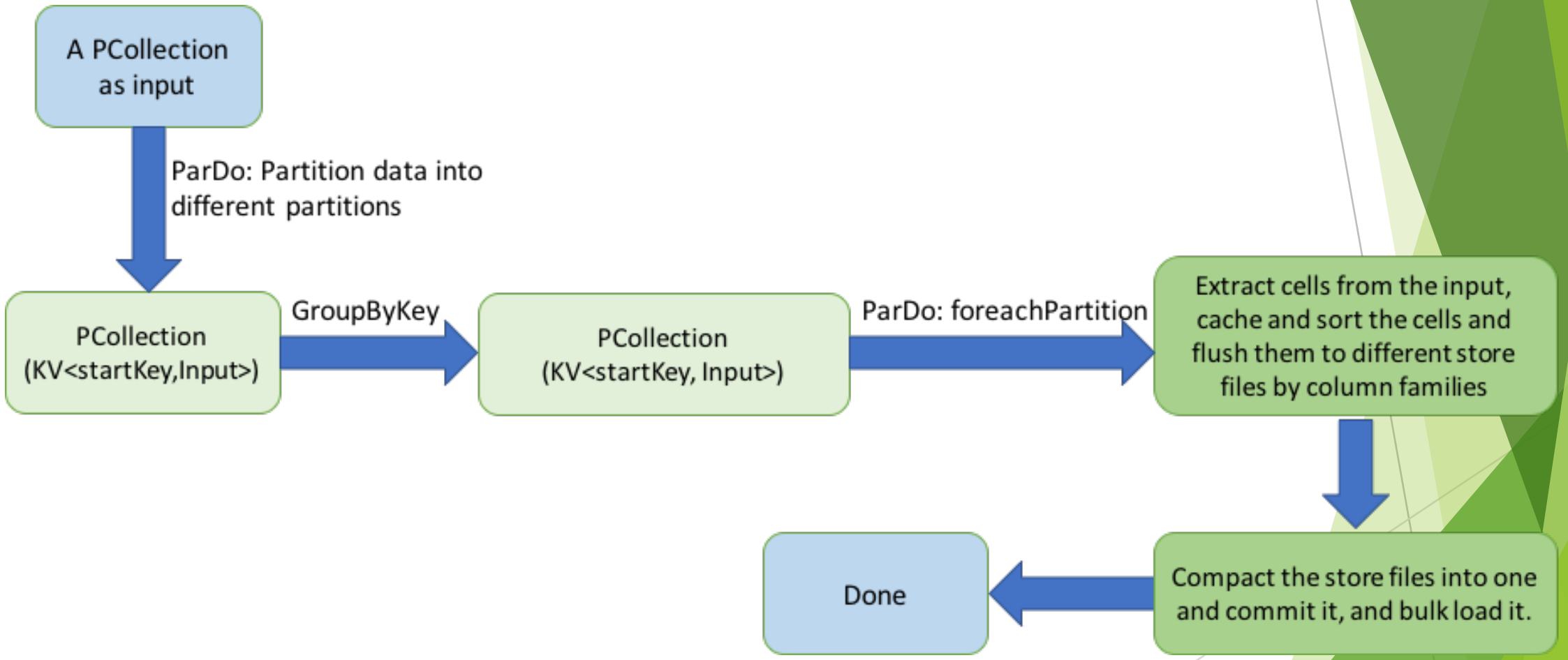


Example: BulkLoadThinRows

- Implement MakeFunctions to convert each input into row keys and cells.

```
HBasePipelineFunctions.bulkLoadThinRows(conf, tableNameAsString, inputs, stagingPath.toUri(),
  tmpPath.toUri(), new MakeFunction<String, byte[]>() {
    @Override
    public byte[] make(String input) {
      byte[] rowKey = null;
      // steps to extract input into row key
      return rowKey;
    }
  }, new MakeFunction<String, Iterable<Cell>>() {
    @Override
    public Iterable<Cell> make(String input) {
      List<Cell> cells = new ArrayList<>();
      // steps to extract input into cells
      return cells;
    }
  }, StringUtf8Coder.of());
```

Example: BulkLoadThinRows



Future

- Contribute the code to Apache Beam
- Support Beam SQL in HBase

Thank You!