Apache Spark

For High-Throughput Systems

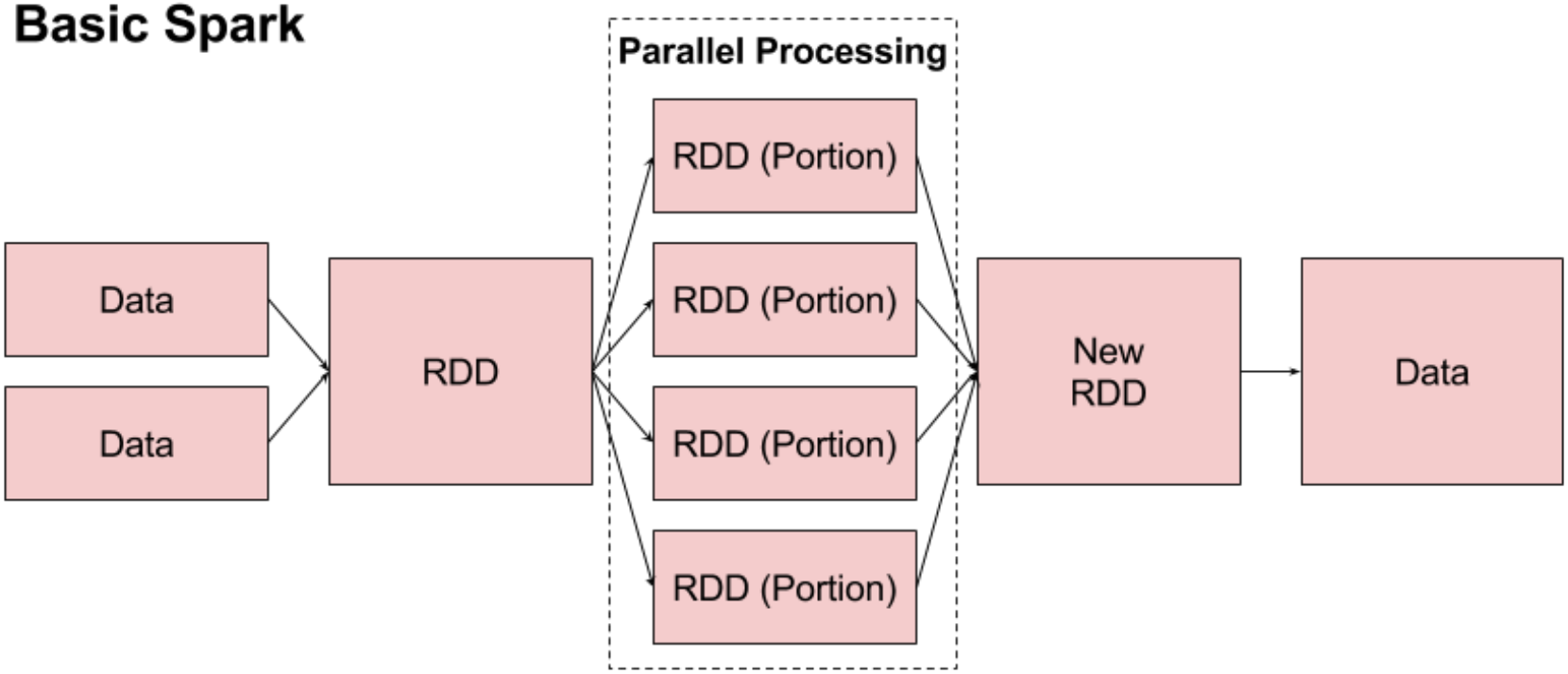
Michael Starch – NASA Jet Propulsion Laboratory

Agenda

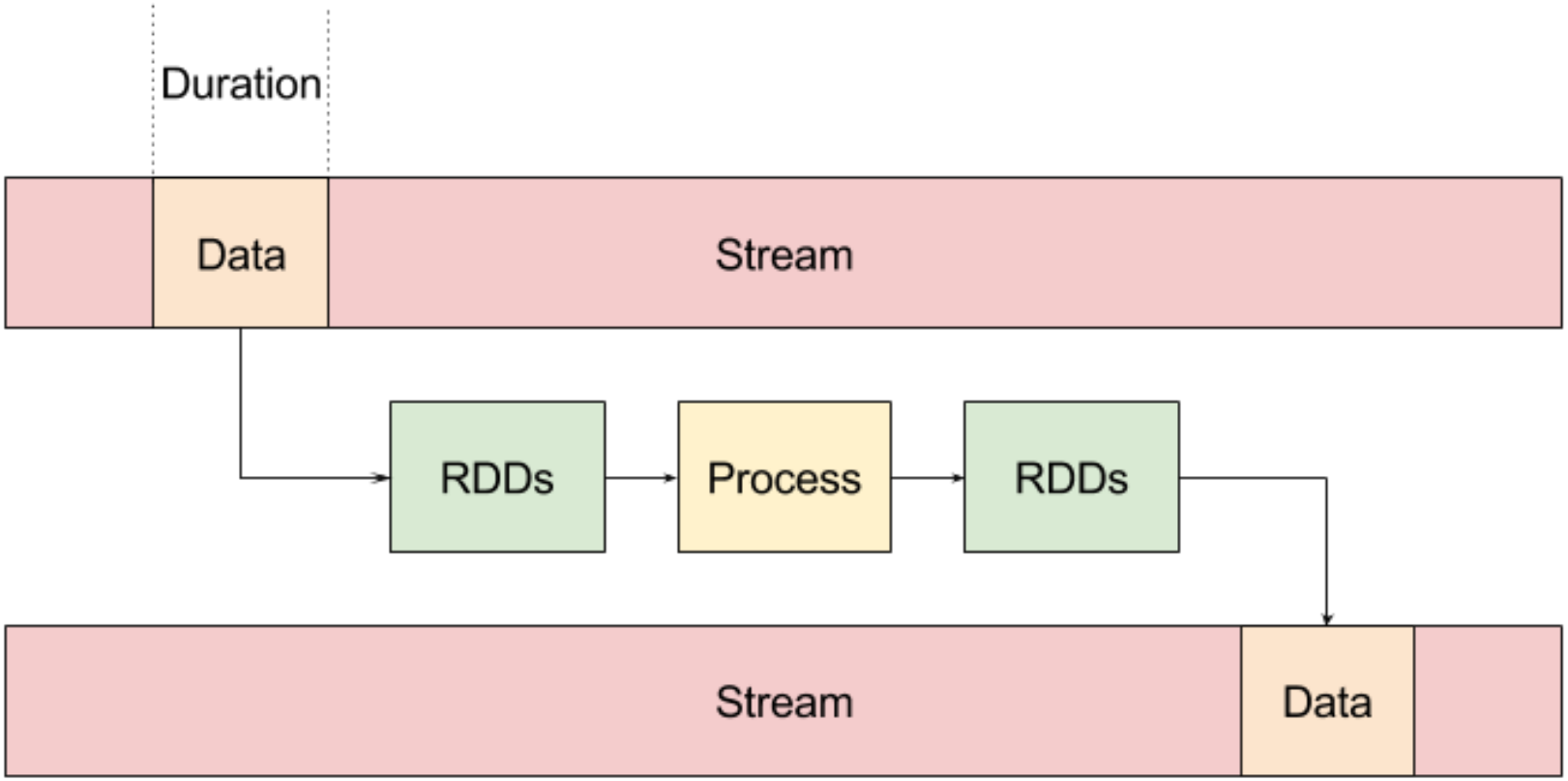
- Basic Concepts
- Setup
- Naïve Implementation
- Optimizations
- Better Implementation
- Code
- Recommendations
- Questions

Basic Concepts

Basic Spark

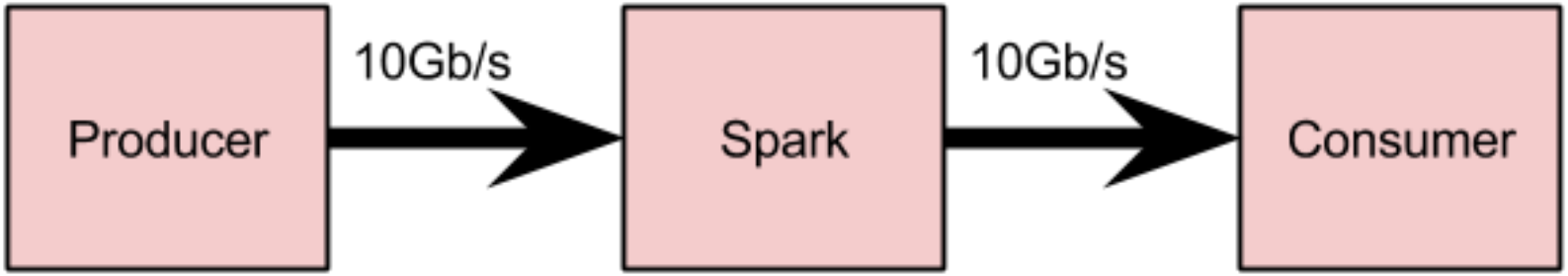

Parallel Processing
RDD (Portion)
RDD (Portion)
RDD (Portion)
RDD (Portion)
Data
Data
RDD
New
RDD
Data

Spark Streaming

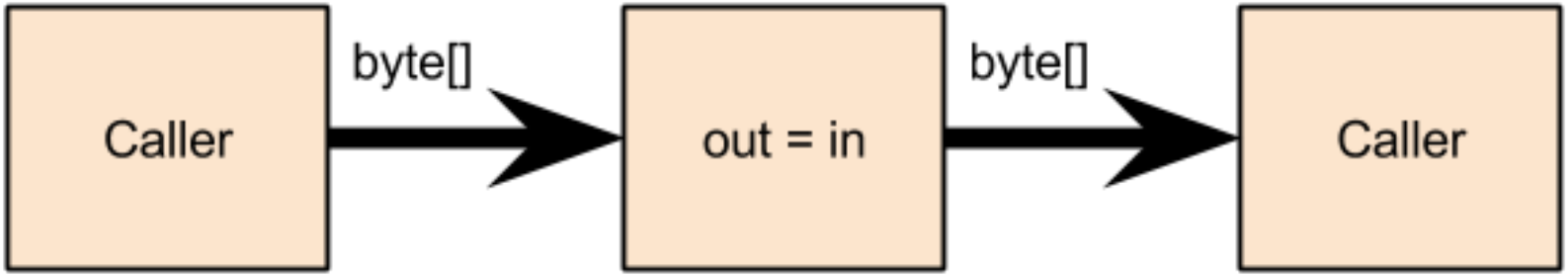


Setup

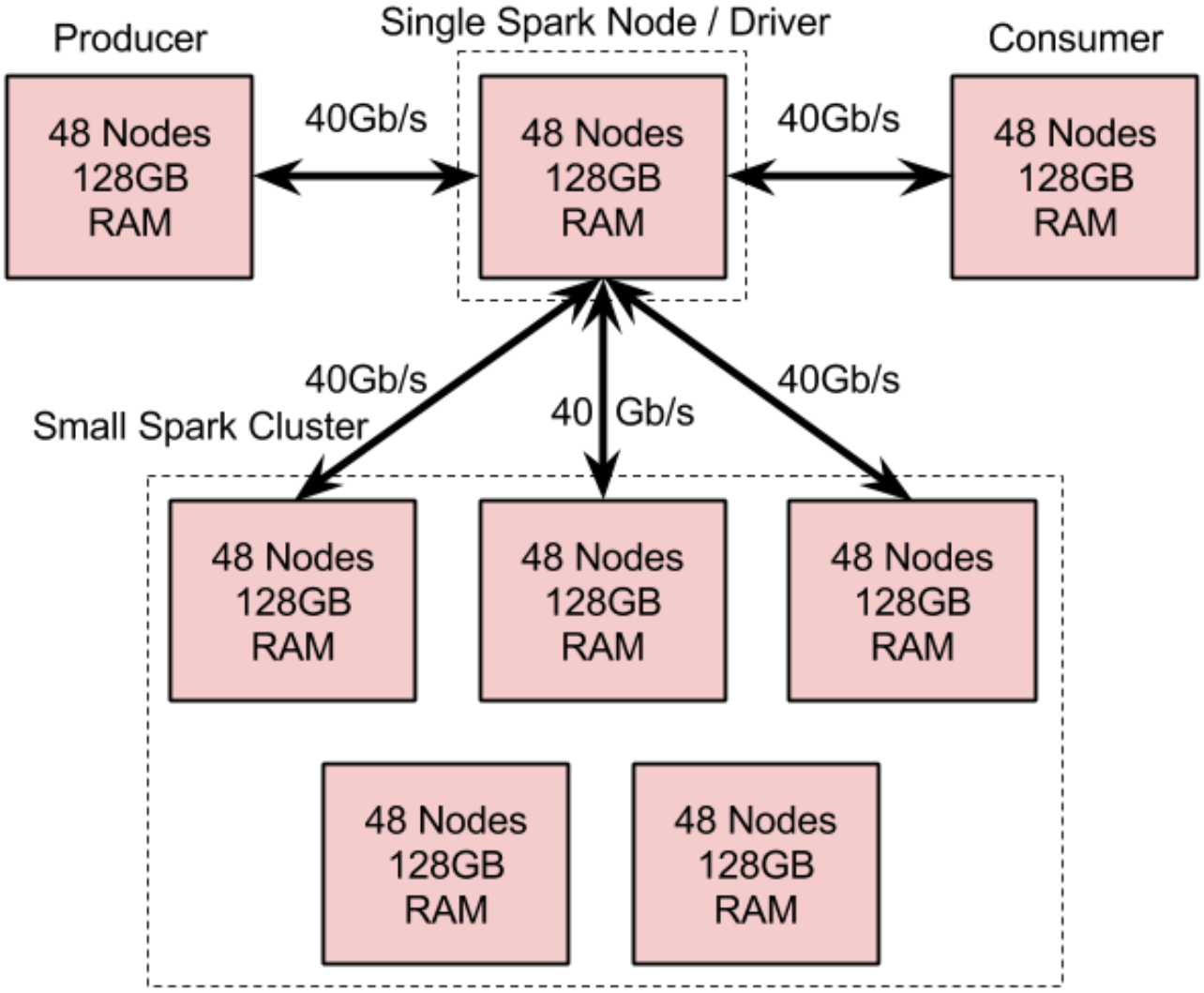
Task Goal



Computation

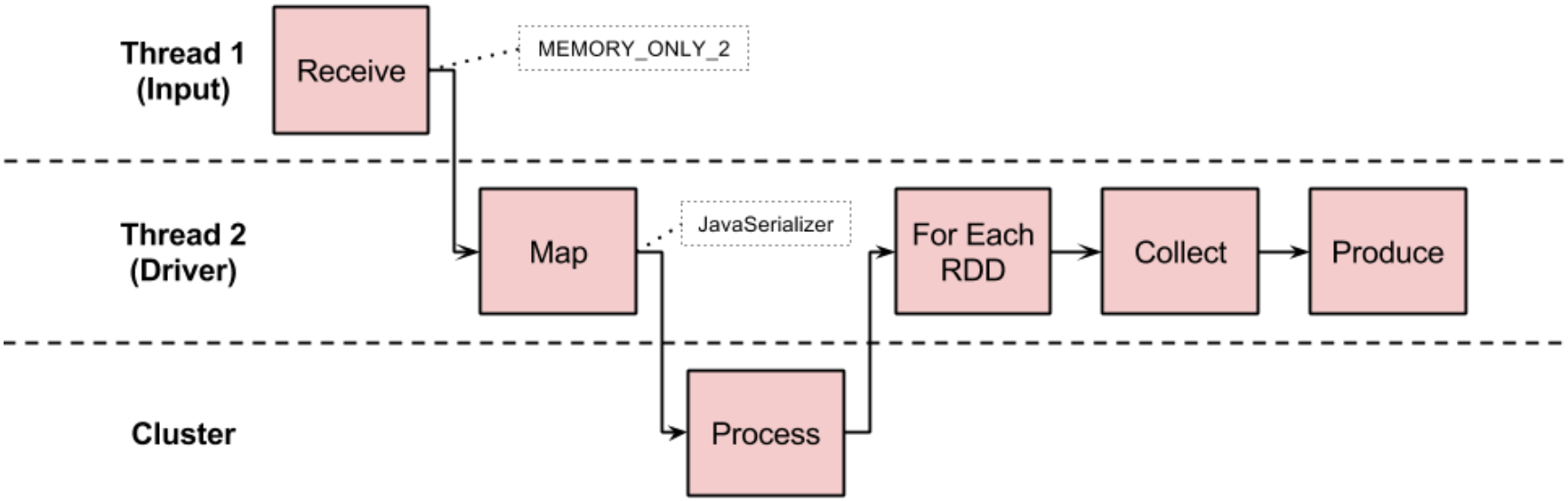


Environment - HPC Cluster



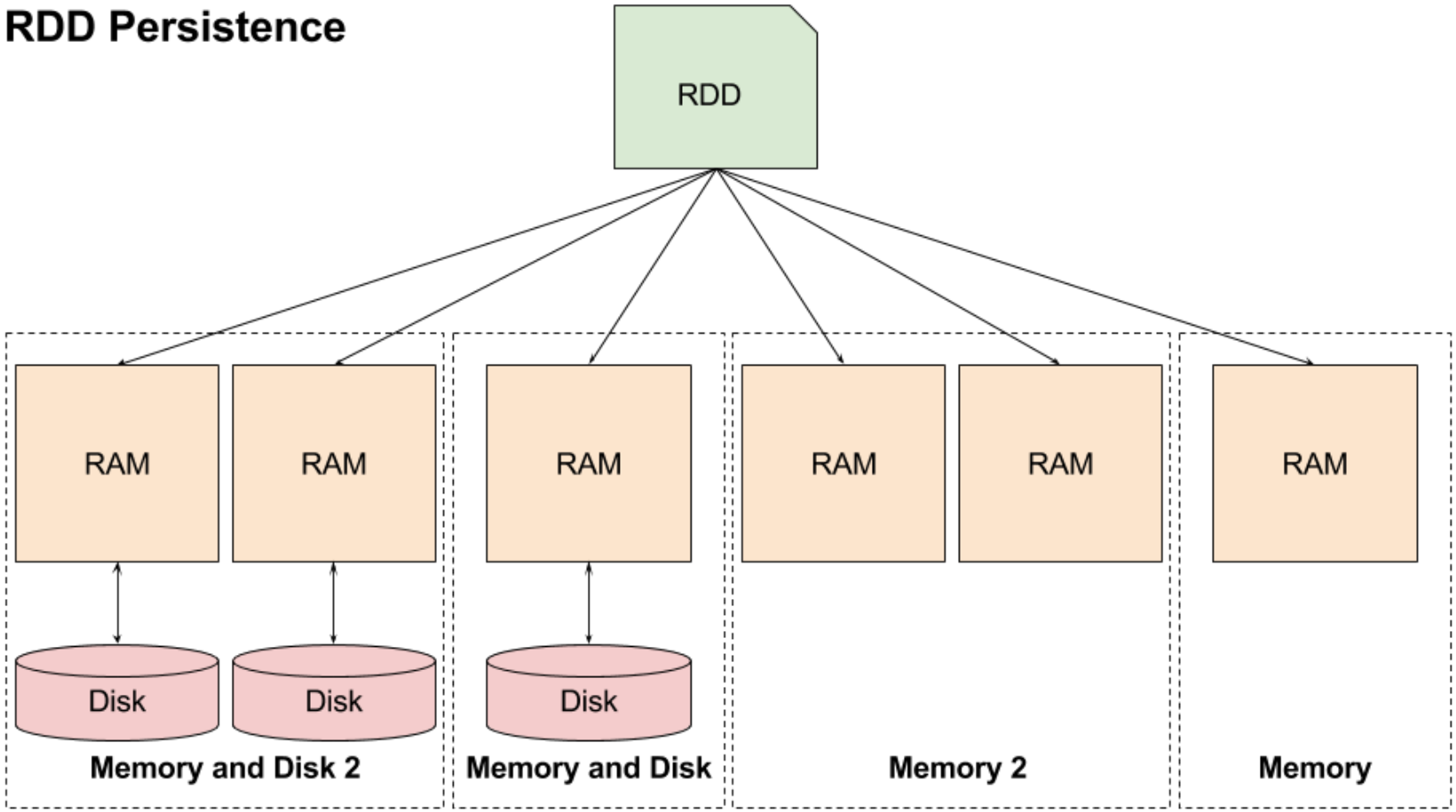
Naïve Implementation

Naïve Implementation

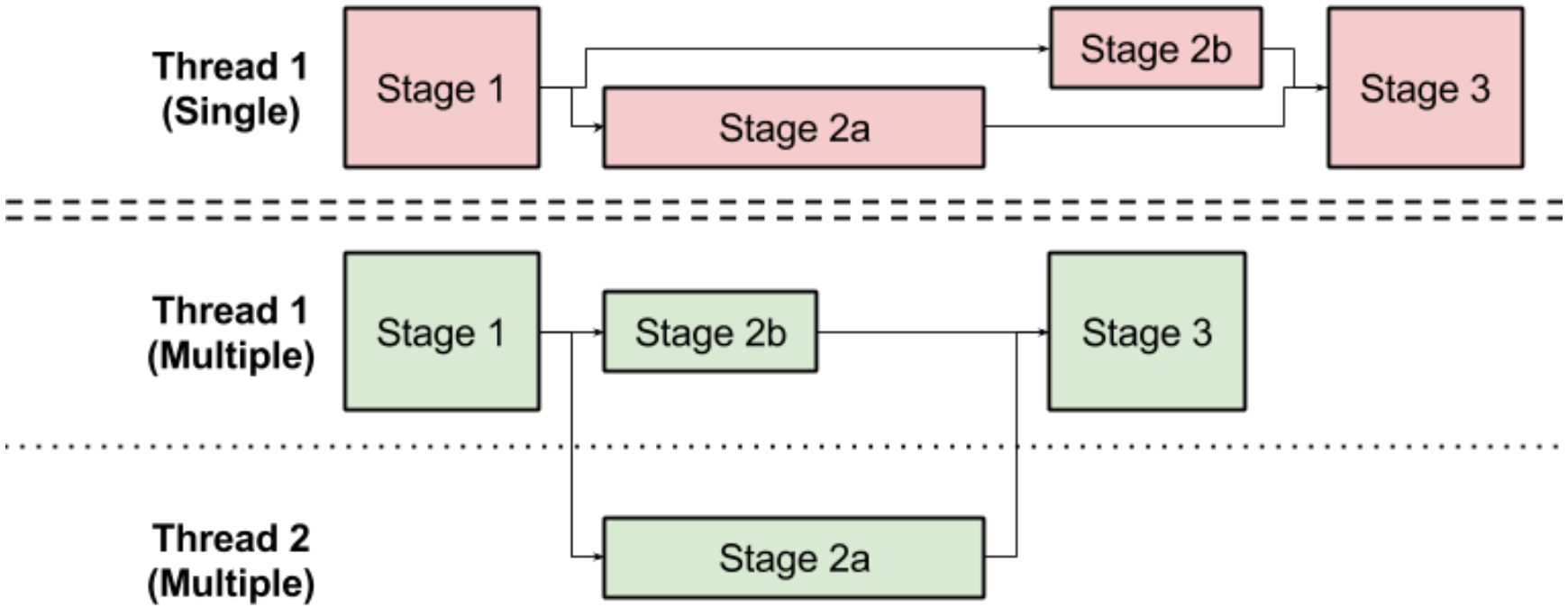


Optimizations

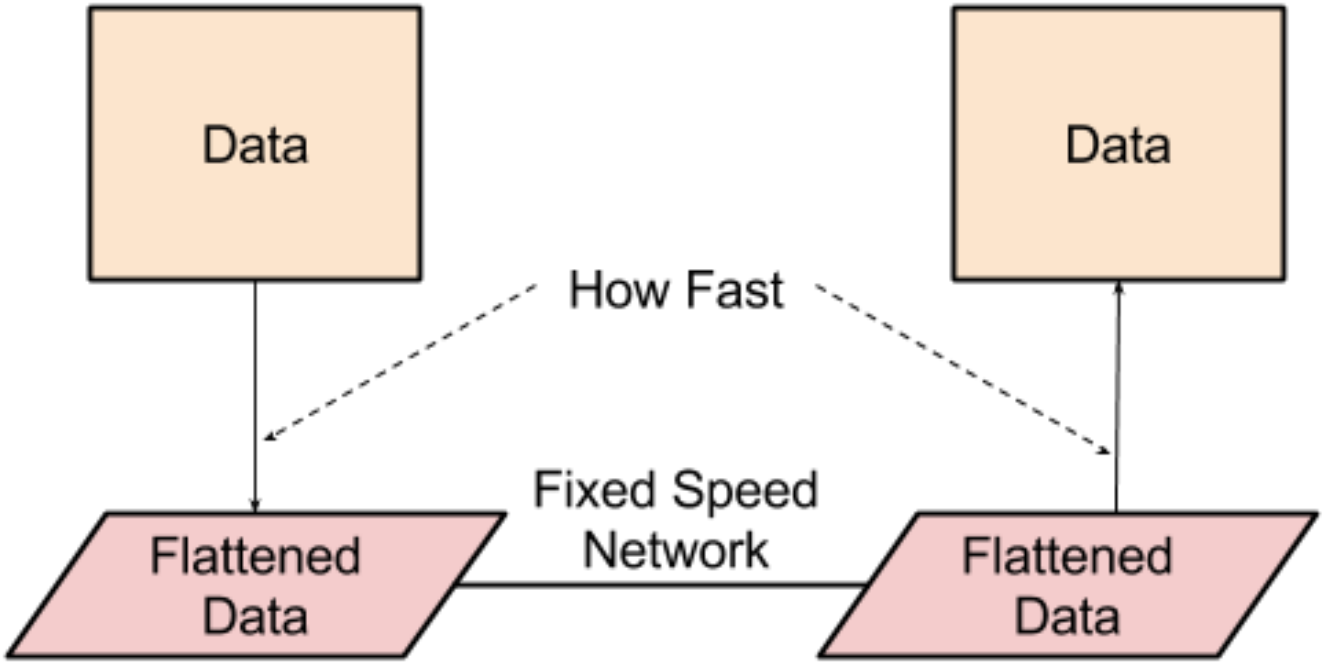
RDD Persistence



Threading

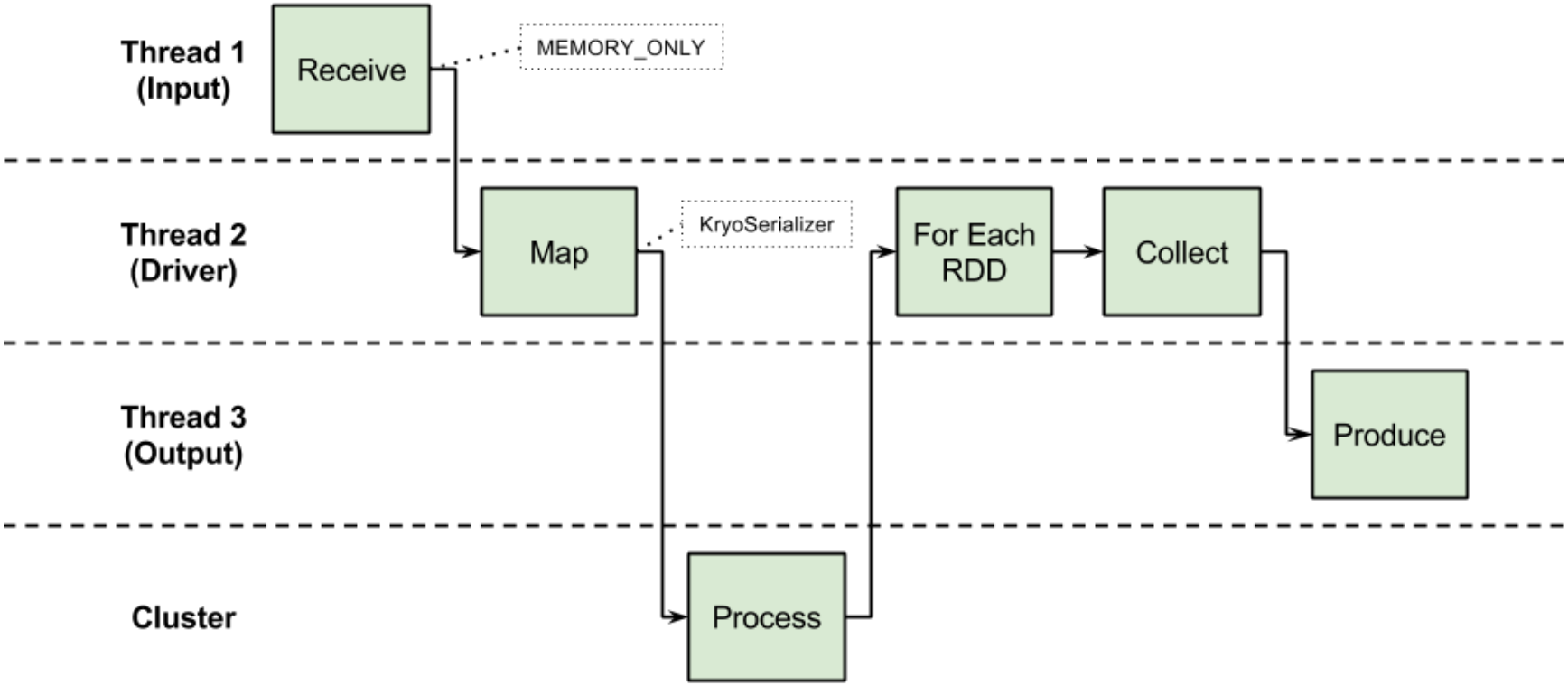

Thread 1
(Single)
Stage 1
Stage 2b
Stage 2a
Stage 3
Thread 1
(Multiple)
Stage 1
Stage 2b
Stage 3
Thread 2
(Multiple)
Stage 2a

Serialization



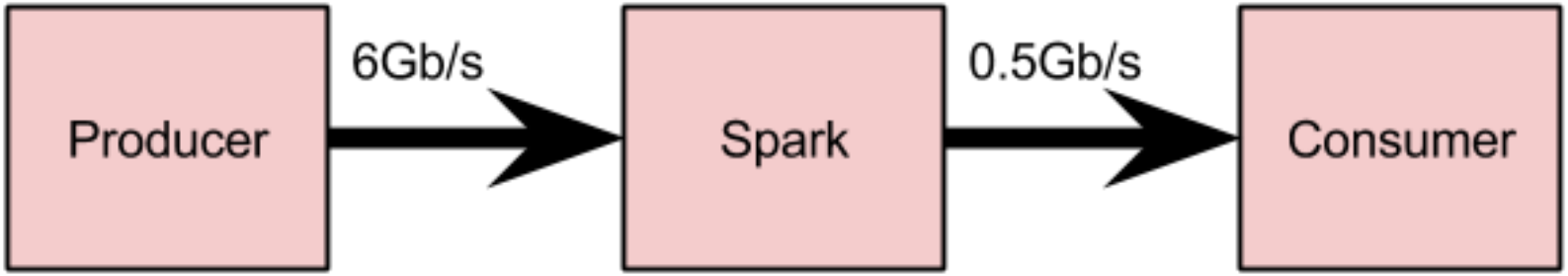
Better Implementation

Better Implementation



Performance

Spark Performance



Code

Code – Spark

```
SampleSetReceiver samples =
    new SampleSetReceiver(StorageLevel.MEMORY_ONLY() …
JavaSparkContext sc = new JavaSparkContext(c);
JavaStreamingContext ssc = new JavaStreamingContext(sc,
    new Duration(duration));
//Processing chain
JavaDStream<byte[]> stream = ssc.receiverStream(samples);
JavaDStream<byte[]> output = stream.map(fourier);
… .foreachRDD(new Function<JavaRDD<byte[]>,Void>()
    {
        private static final long serialVersionUID = 1L;
        @Override
        public Void call(JavaRDD<byte[]> rdd) throws Exception {
            for (byte[] bytes : rdd.collect()) {
                outFn.call(bytes);
            }
            return null;
        }
    });
ssc.start();
ssc.awaitTermination();
ssc.close();
```

Code – Input

```
public class SampleSetReceiver extends Receiver<byte[]> {
  …
    @Override
    public void onStart() {
         …
             Thread thread = new Thread(
                 new ReadSocketThread(this.host,this.port));
             thread.start();
        …
    }
    class ReadSocketThread implements Runnable {
         …
        @Override
        public void run() {
            …
                while (!SampleSetReceiver.this.isStopped()) {
                    byte[] bytes = this.sock.read();
                    SampleSetReceiver.this.store(bytes);
                }
            …
        }
}
```

Code – Output

```
class WriteSocketThread implements Runnable {
        ConcurrentLinkedQueue<byte[]> queue;
        …

        WriteSocketThread(String host,int port) throws … {
            this.queue = new ConcurrentLinkedQueue<byte[]>();
            …
        }

        public void queueWrite(byte[] data) {
            this.queue.add(data);
        }


        @Override
        public void run() {
            while (true) {
                byte[] data = null;
                while(null == (data = this.queue.poll()));
                …
             }
        }
…
```

Code – Kryo Serialization

```
c.set("spark.serializer", "org.apache.spark.serializer.KryoSerializer");
c.set("spark.kryoserializer.buffer.max", "512m");
c.set("spark.kryoserializer.buffer", "512m");
c.set("spark.kryoserializer.referenceTracking", "false");
```

… basic implementation only

Recommendations

Recommendations

- Keep Input and Output on separate threads
  - Receiver is always its own thread
  - Output of system should also
- Use Kryo Serialization
- Be careful with storage level
- Tune your chunking duration
- Avoid high-throughput single streams
  - Hit all the bottlenecks
  - Much work

Long-Shots

- Do not be afraid of long-shot solutions

```
/*
 * Class:     org_dia_benchmarking_spark_jni_FastNetwork
 * Method:    write
 * Signature: ([B)V
 */
JNIEXPORT jint JNICALL Java_org_dia_benchmarking_spark_jni_FastNetwork_writeX(JNIEnv * env, jobject
thiso,jint conn, jbyteArray data) {
    jint len = (*env)->GetArrayLength(env, data);
    size_t tmp = 0;
    jbyte *buffer = (*env)->GetPrimitiveArrayCritical(env, data, NULL);
    if (buffer == NULL) {
       (*env)->ReleasePrimitiveArrayCritical(env, data, buffer, JNI_ABORT);
       return -1;
    }
    while (tmp < len) {
         size_t ret = -1;
         size_t towr = (len-tmp>=BLOCK_SIZE)?BLOCK_SIZE:len-tmp;
         ret = write(conn,buffer+tmp,towr);
         if (ret < 0) {
             (*env)->ReleasePrimitiveArrayCritical(env, data, buffer, JNI_ABORT);
             return ret;
         }
         tmp += ret;
    }
    (*env)->ReleasePrimitiveArrayCritical(env, data, buffer, JNI_ABORT);
    return tmp;
}
```

Where can I get the code?

It's Open Source! Jump on in!

https://github.com/LeStarch/hyper-socket

https://github.com/LeStarch/hyper-spark

Acknowledgments

NASA Jet Propulsion Laboratory

Research & Technology Development

"Archiving, Processing and Dissemination for the Big Data Era"

Apache Software Foundation

Apache Spark, Apache OODT

Texas Advanced Computing Center

Wrangler

Avez-vous des questions?

你有沒有問題？

Haben Sie Fragen?

Questions?

¿Tienen preguntas?