A SCALABLE AND DISTRIBUTED SEISMIC DATA ANALYTICS TOOLKIT ON BIG DATA PLATFORM

Chao Chen

Outline

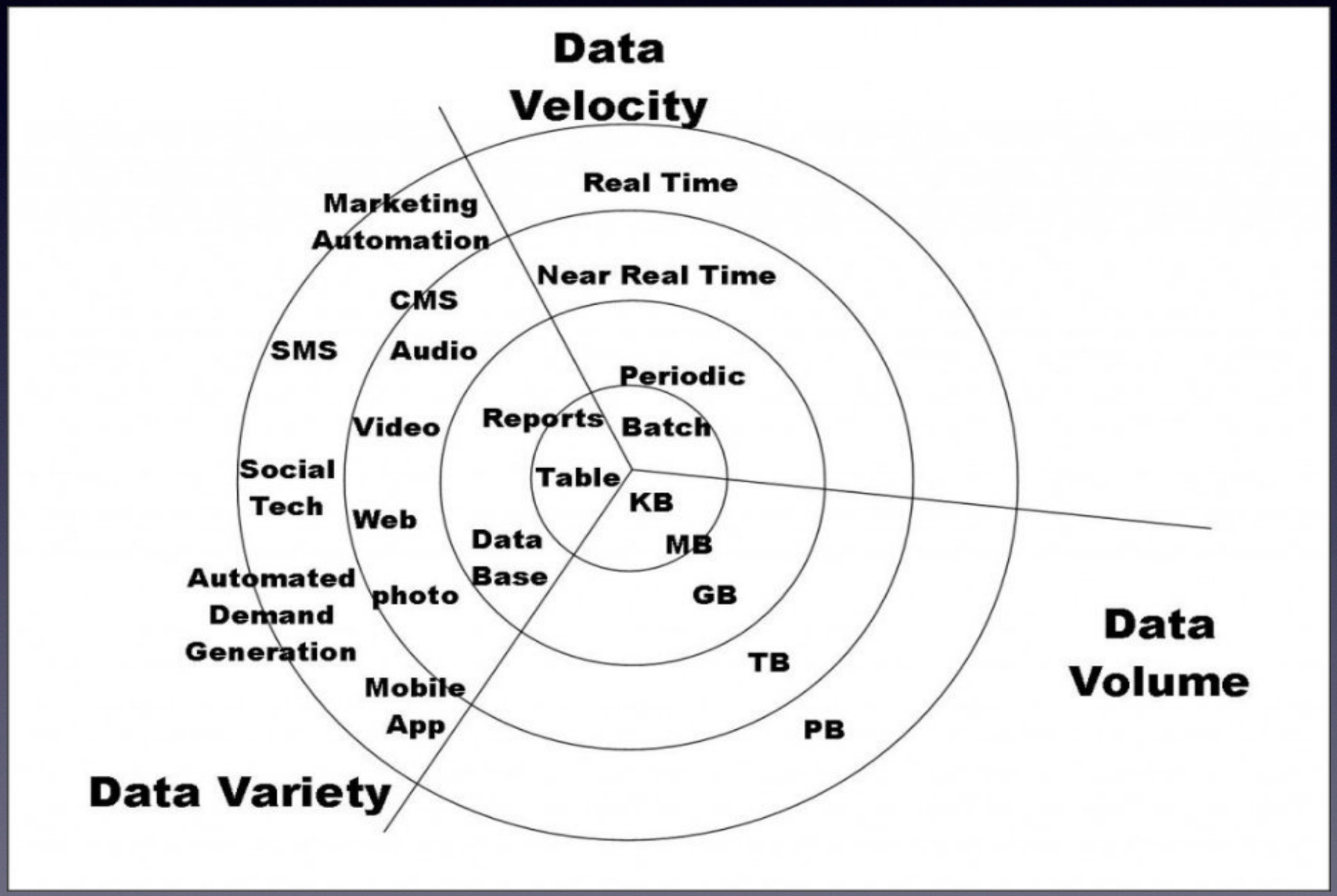
- What is Big Data
- Seismic Data Analytics
- Why Spark
- Design & Architecture
- Implement
- Parallel Templates
- Scalability and Performance
- Cloud Platform Services (Web)
- Demo

“The world is one big data problem.”

– Andrew McAfee

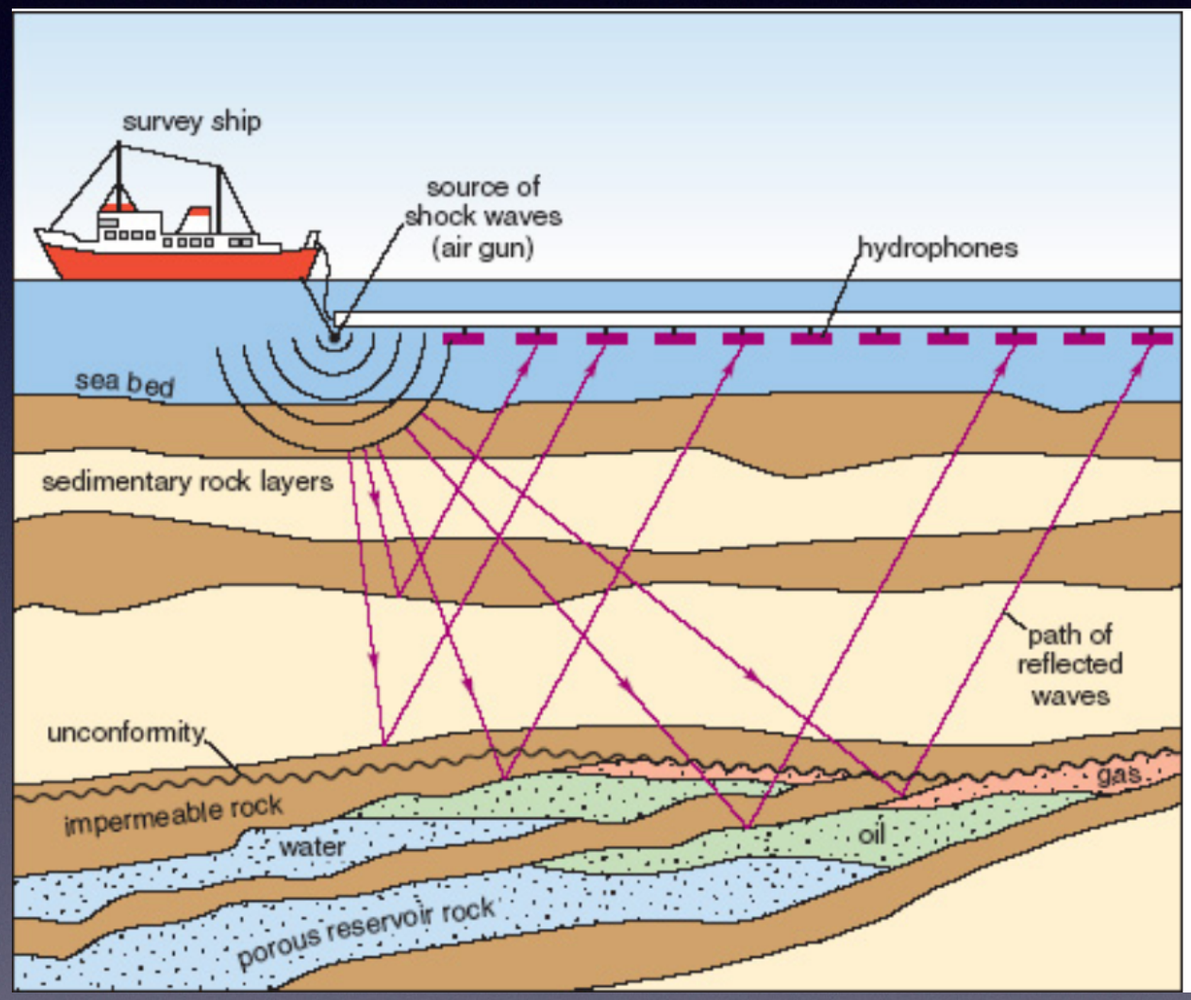
What is Big Data

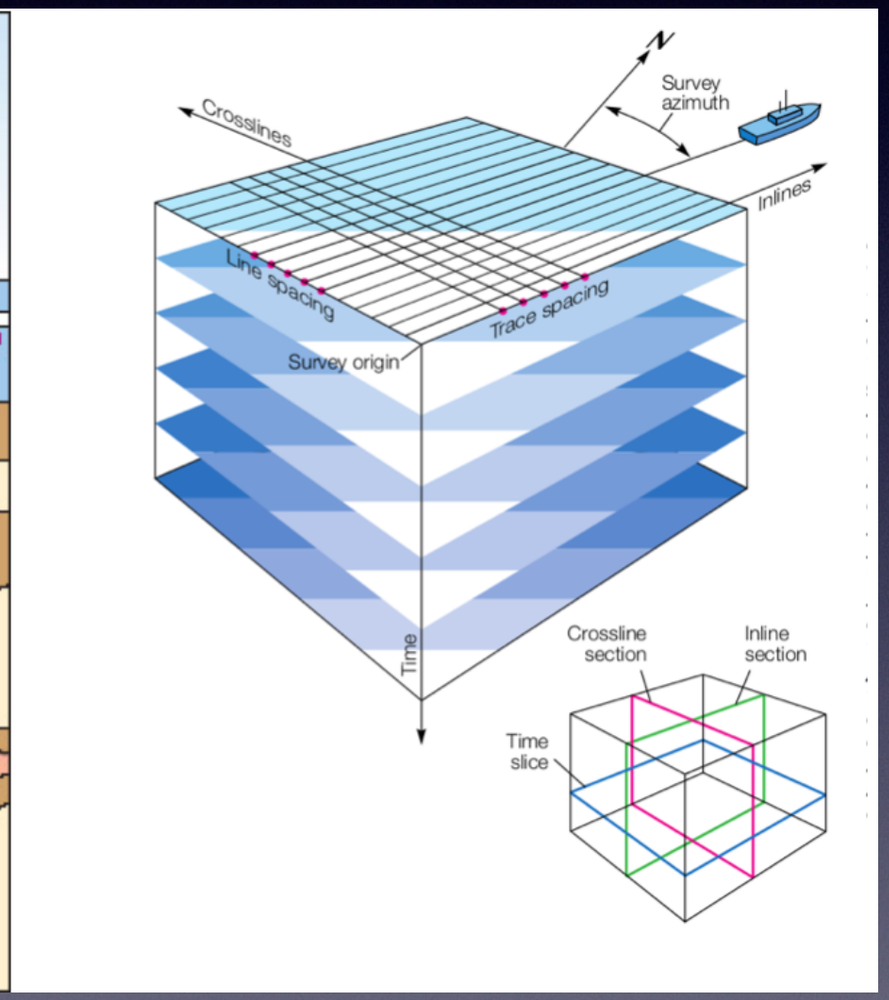
- 3Vs



Seismic Data Analytics

- Seismic Data Acquisition

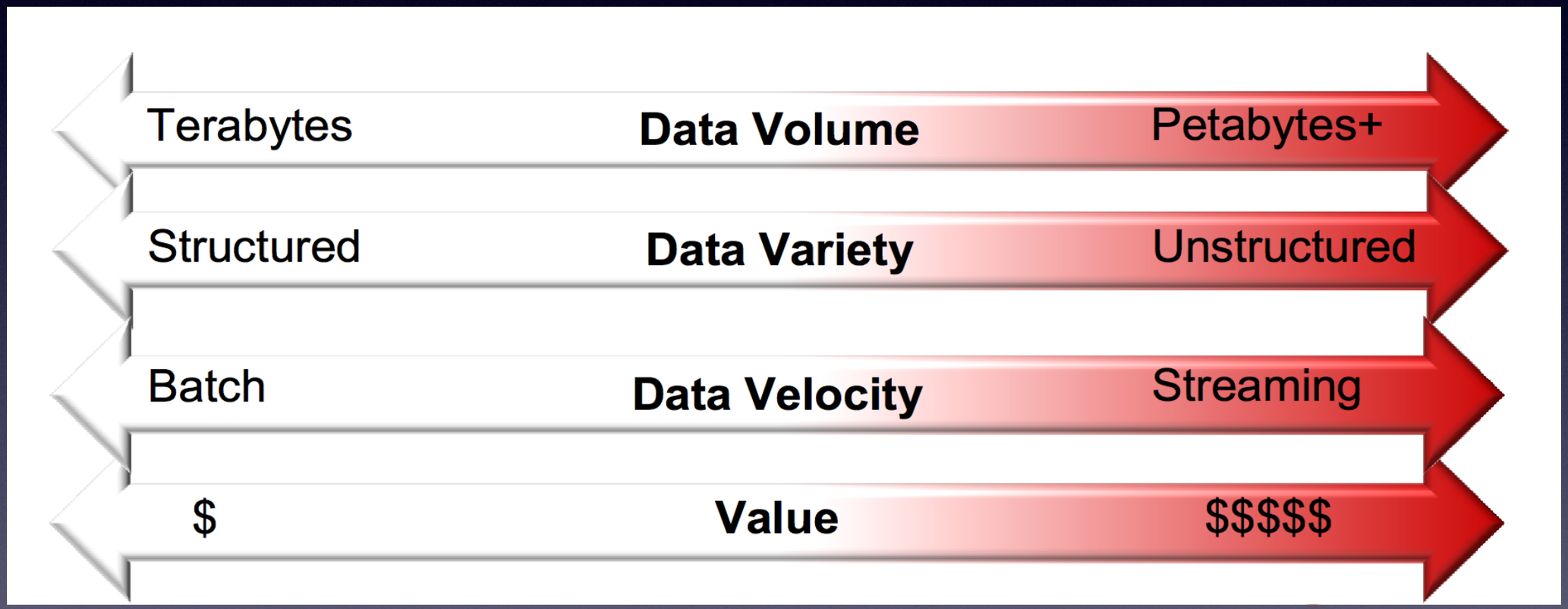




Seismic Data Analytics

- 3Vs
  - **Volume** (Seismic data acquisition, wide azimuth)
    - 300MB/km^2 (90s) ~ 25GB/km^2 (2006) ~ PBs/km^2 (Now)
  - **Variety**
    - Structured(SEGY) / Unstructured(Images, well logs) / Semistructured (Analysis, reports)
  - **Velocity** (Full 3D Acquisition, Real-time Sensors)
    - 2GB/s ~ 20GB/s ~ 50GB/s

Seismic Data Analytics



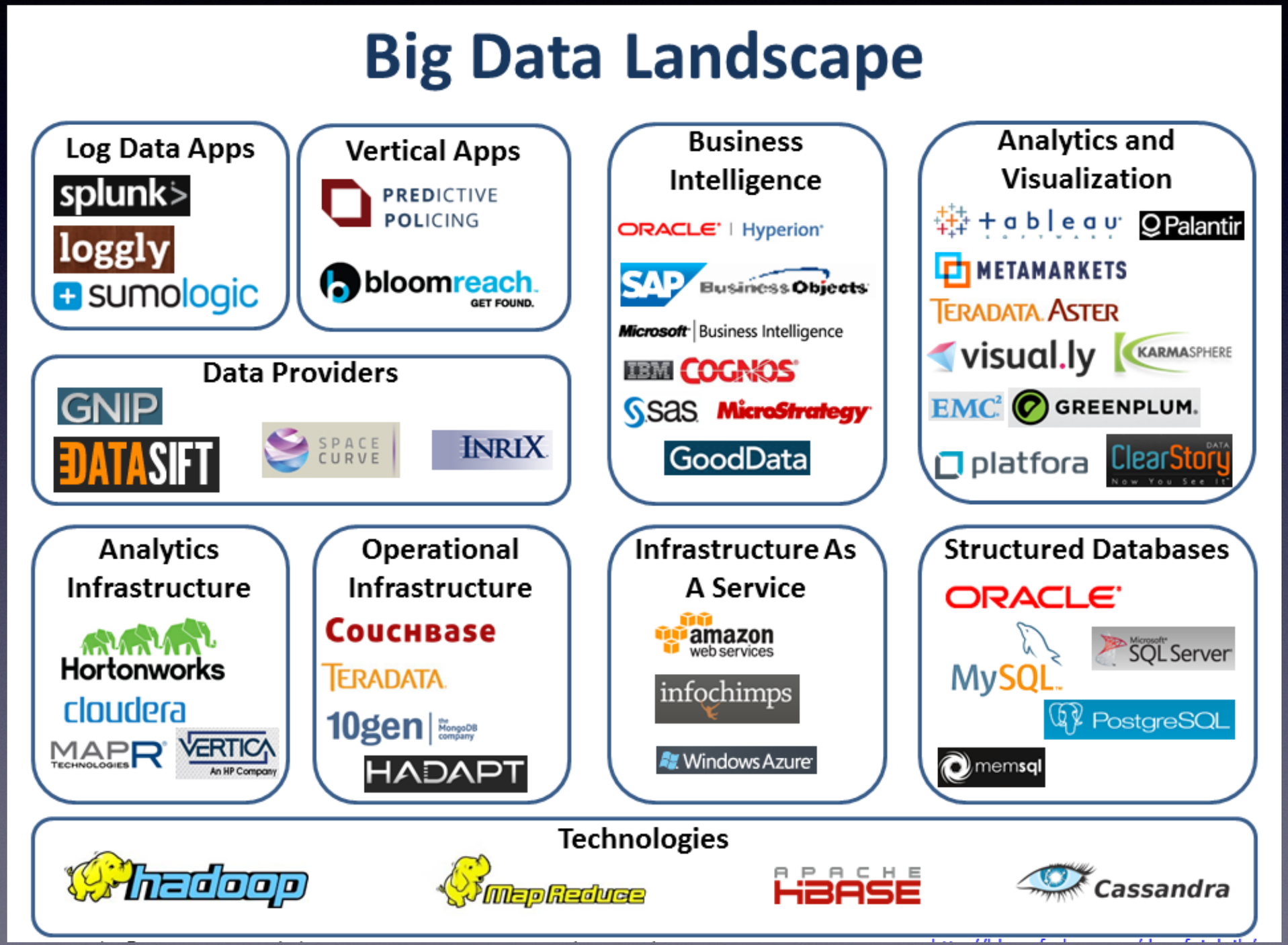
Seismic Data Analytics

- Conventional Workflow
  - Data Interpretation (Geophysicists)
  - Data Visualization (Desktop Software)
  - Matlab Models (Geophysicists)
  - MPI Models (Geophysicists & Programmer)
  - Execution (Workstation / HPC)

Seismic Data Analytics

- Big Data Challenges
  - Performance
  - Scalability
  - Complicated Workflow

Why Spark


Big Data Landscape
Log Data Apps
splunk>
loggly
sumologic
Vertical Apps
PREDICTIVE POLICING
bloomreach
GET FOUND.
Data Providers
GNIP
DATASIFT
SPACE CURVE
INRIX
Business Intelligence
ORACLE | Hyperion
SAP BusinessObjects
Microsoft | Business Intelligence
IBM COGNOS
SAS MicroStrategy
GoodData
Analytics and Visualization
+ableau
Palantir
METAMARKETS
TERADATA ASTER
visual.ly
KARMASPHERE
EMC²
GREENPLUM.
platfora
ClearStory
Analytics Infrastructure
Hortonworks
cloudera
MAPR TECHNOLOGIES
VERTICA An HP Company
Operational Infrastructure
CouchBase
TERADATA
10gen the MongoDB company
HADAPT
Infrastructure As A Service
amazon web services
infochimps
Windows Azure
Structured Databases
ORACLE
MySQL
Microsoft SQL Server
PostgreSQL
memsql
Technologies
hadoop
Map Reduce
APACHE HBASE
Cassandra

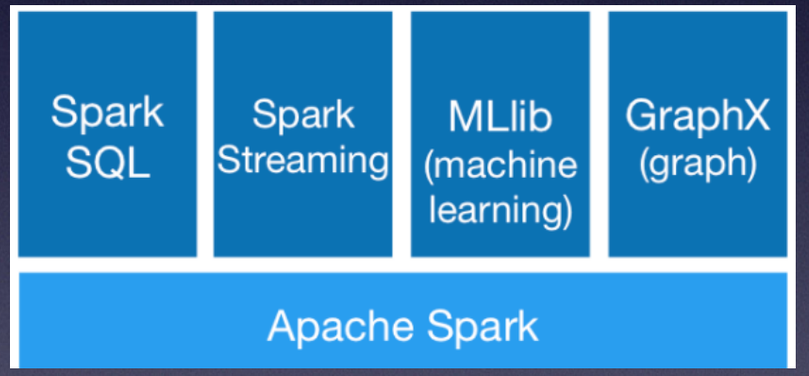
Why Spark

- Ease of Use

```
text_file = spark.textFile("hdfs://...")

text_file.flatMap(lambda line: line.split())
    .map(lambda word: (word, 1))
    .reduceByKey(lambda a, b: a+b)
```

Word count in Spark's Python API

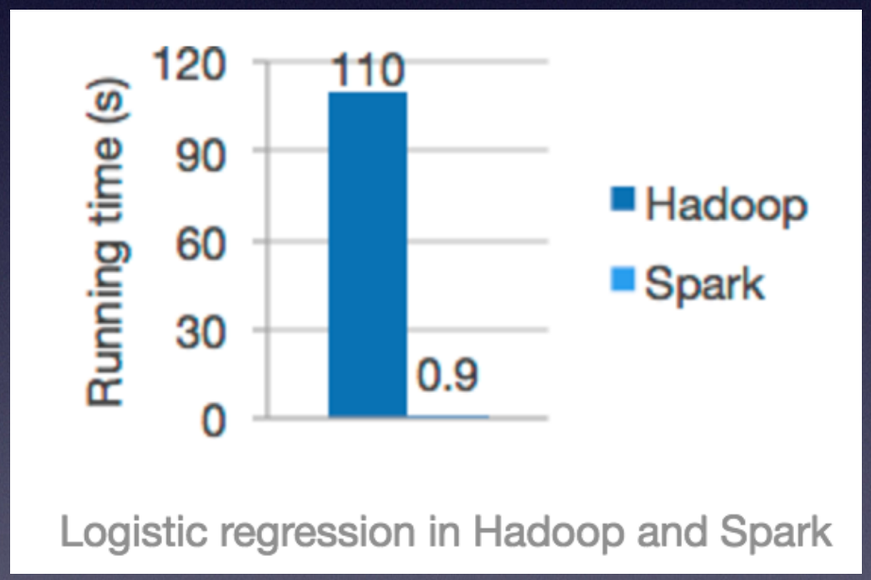
- Generality



- Ecosystem

Why Spark

- Speed
  - Run programs up to 100x faster than Hadoop MapReduce

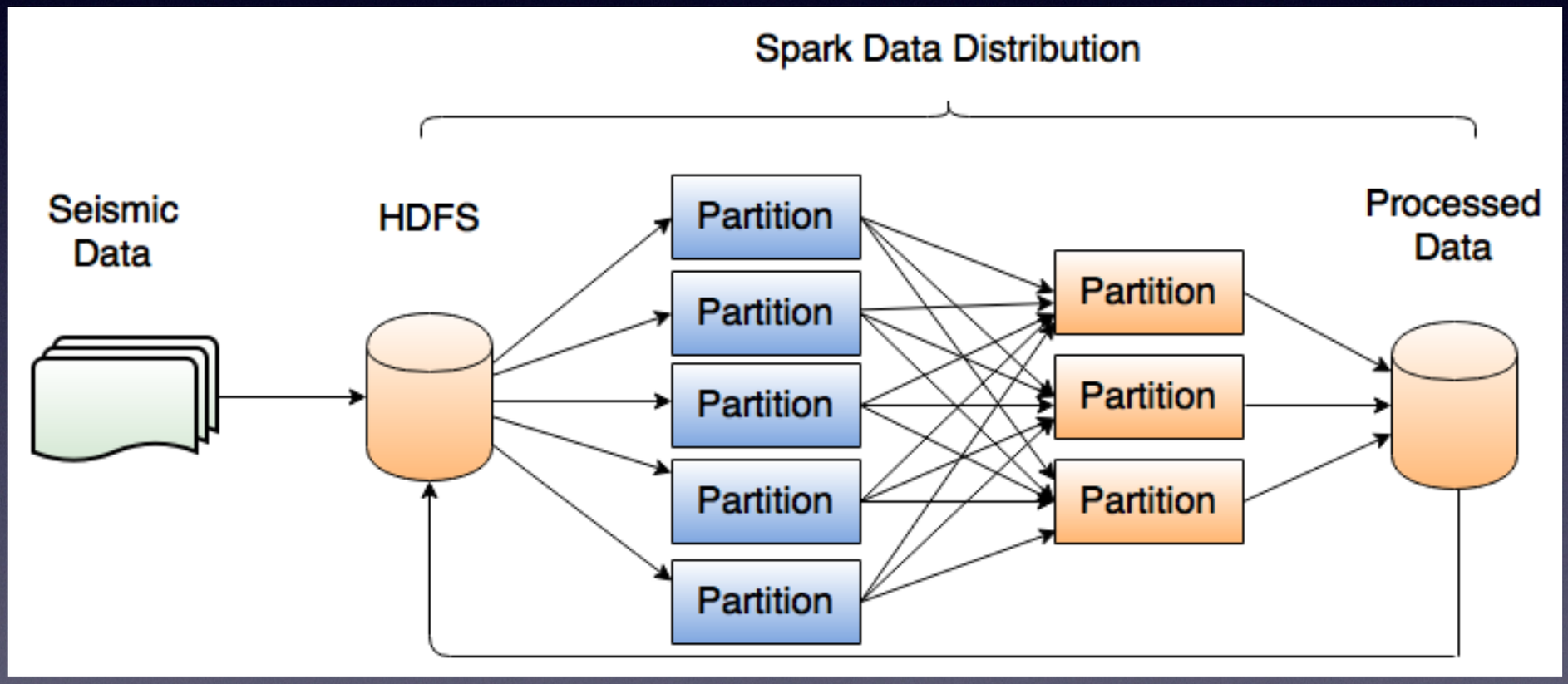


- The magic is …

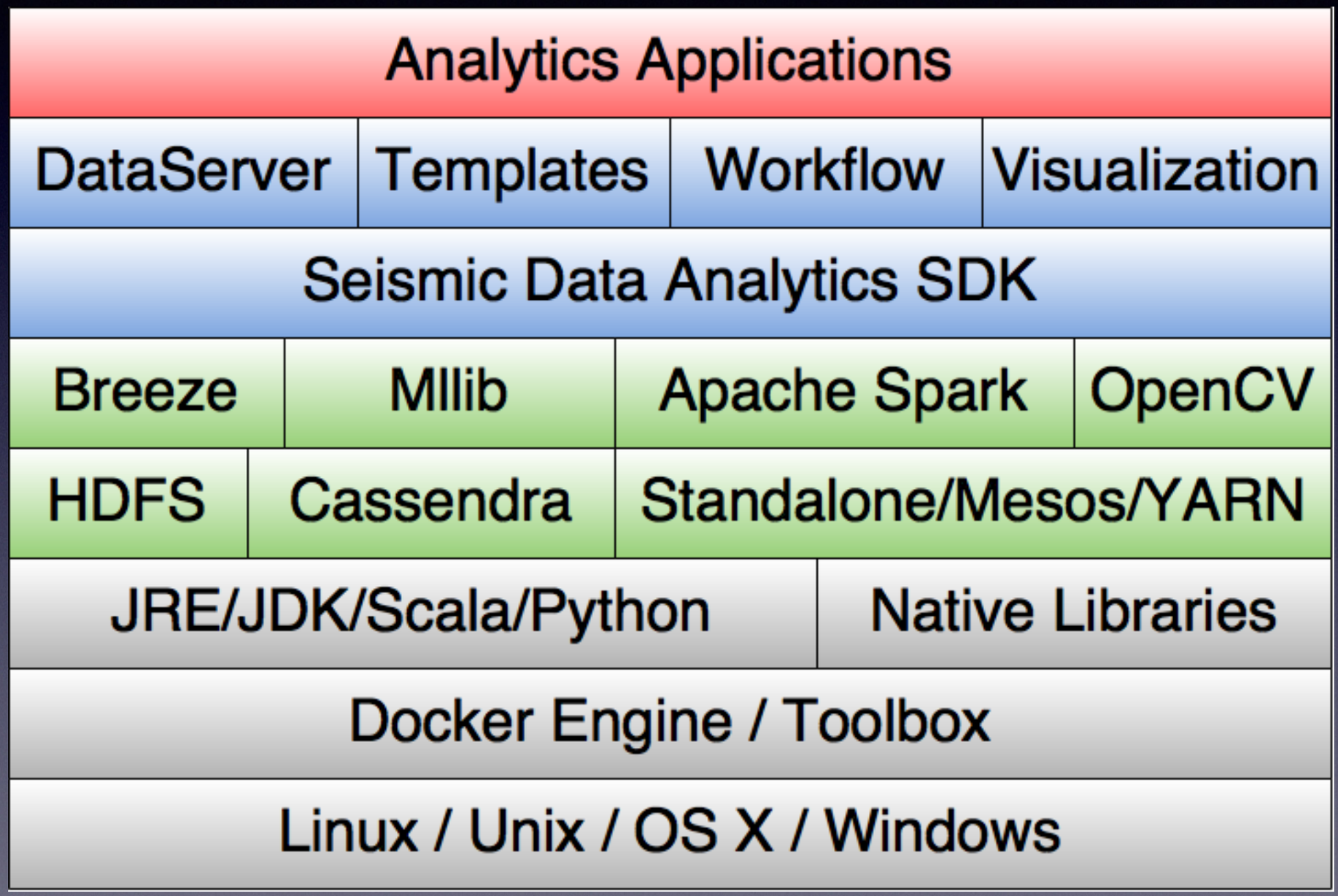
Why Spark

- RDD (Resilient Distributed Datasets)
  - Distributed data collection
  - Parallel Transformations(map, filter, join…)
  - Persisting(caching) data in memory across operations
  - Fault-tolerance

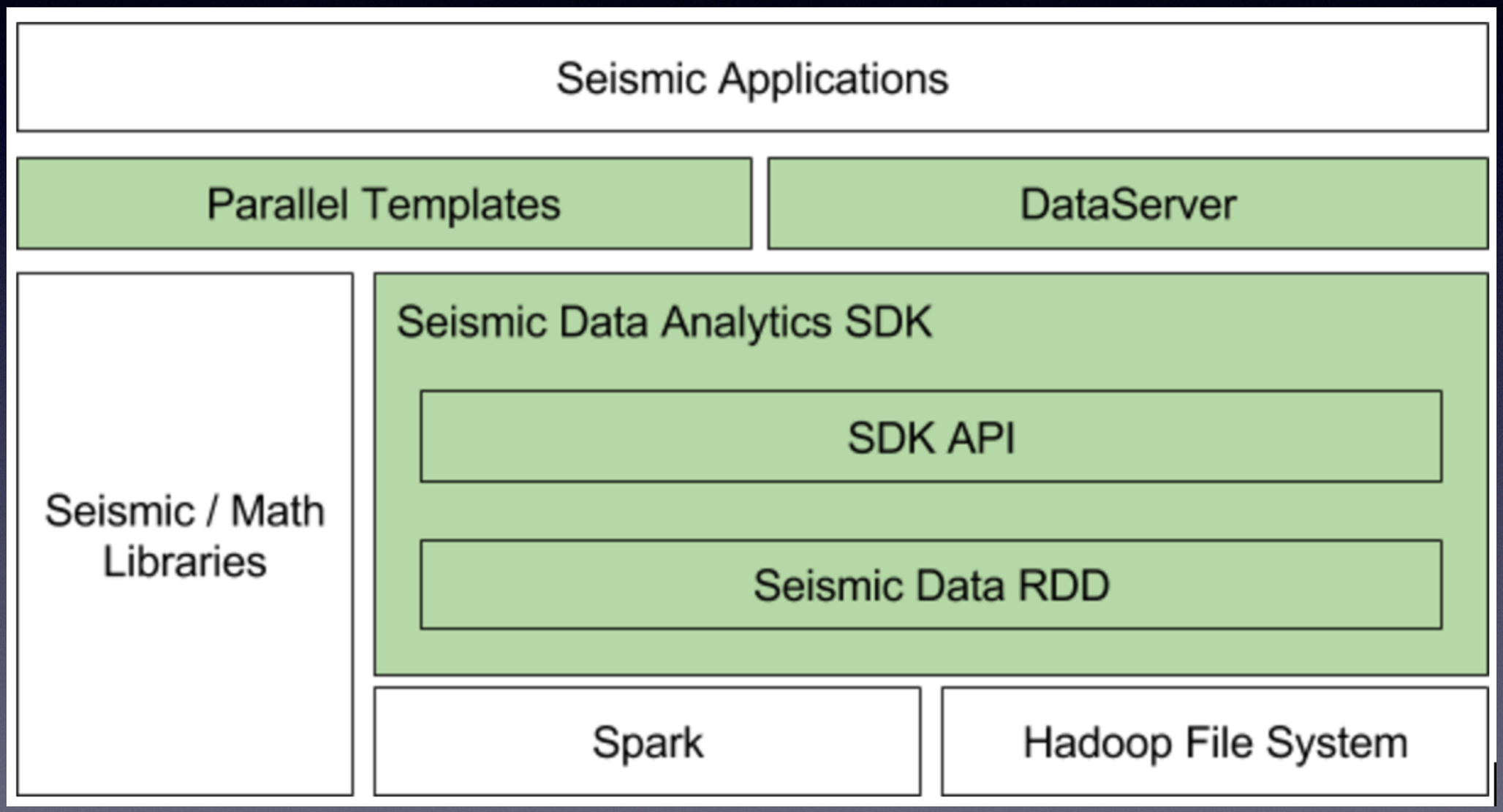
Why Spark


Spark Data Distribution
Seismic Data
HDFS
Partition
Partition
Partition
Partition
Partition
Partition
Partition
Partition
Processed Data

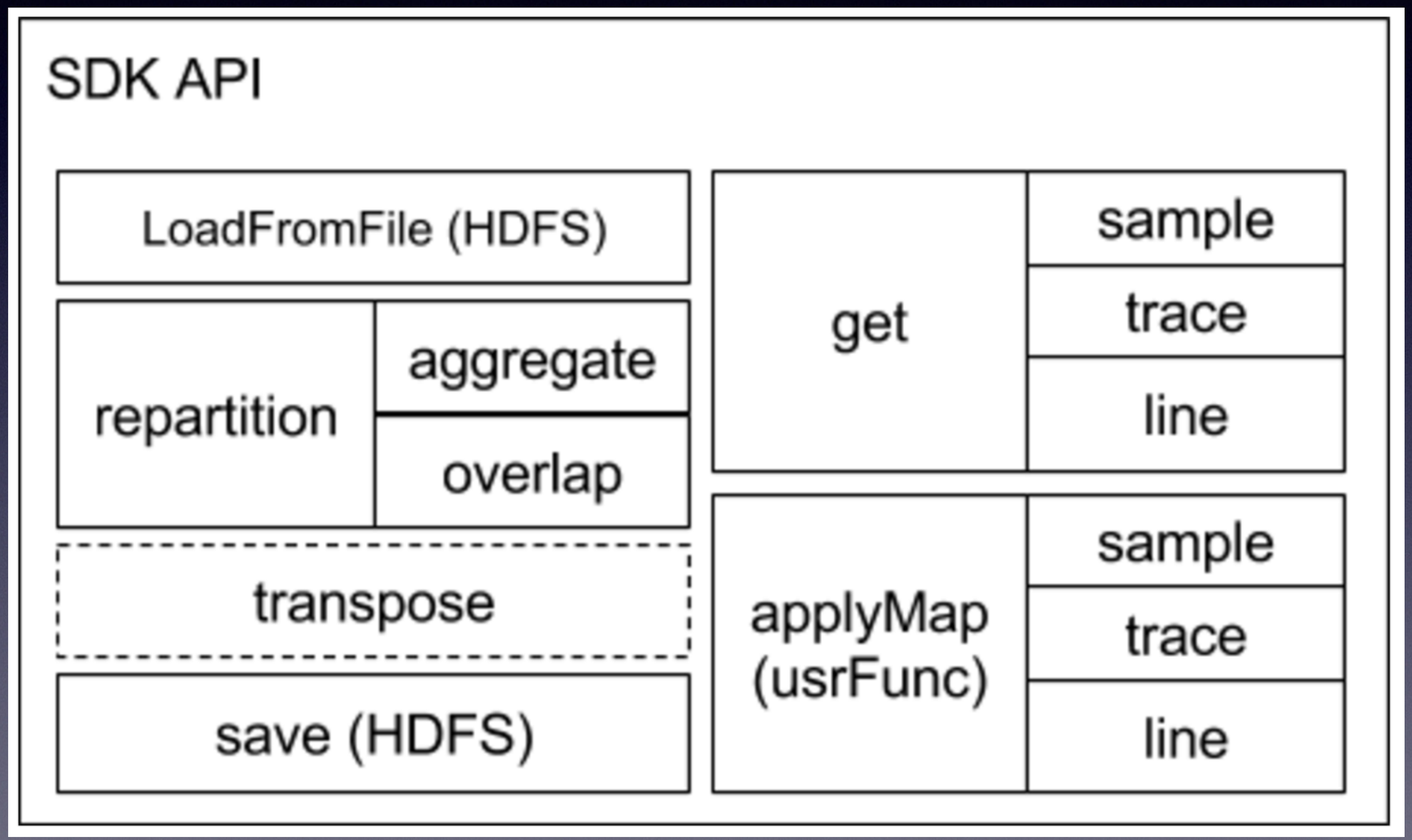
Design & Architecture



Design & Architecture

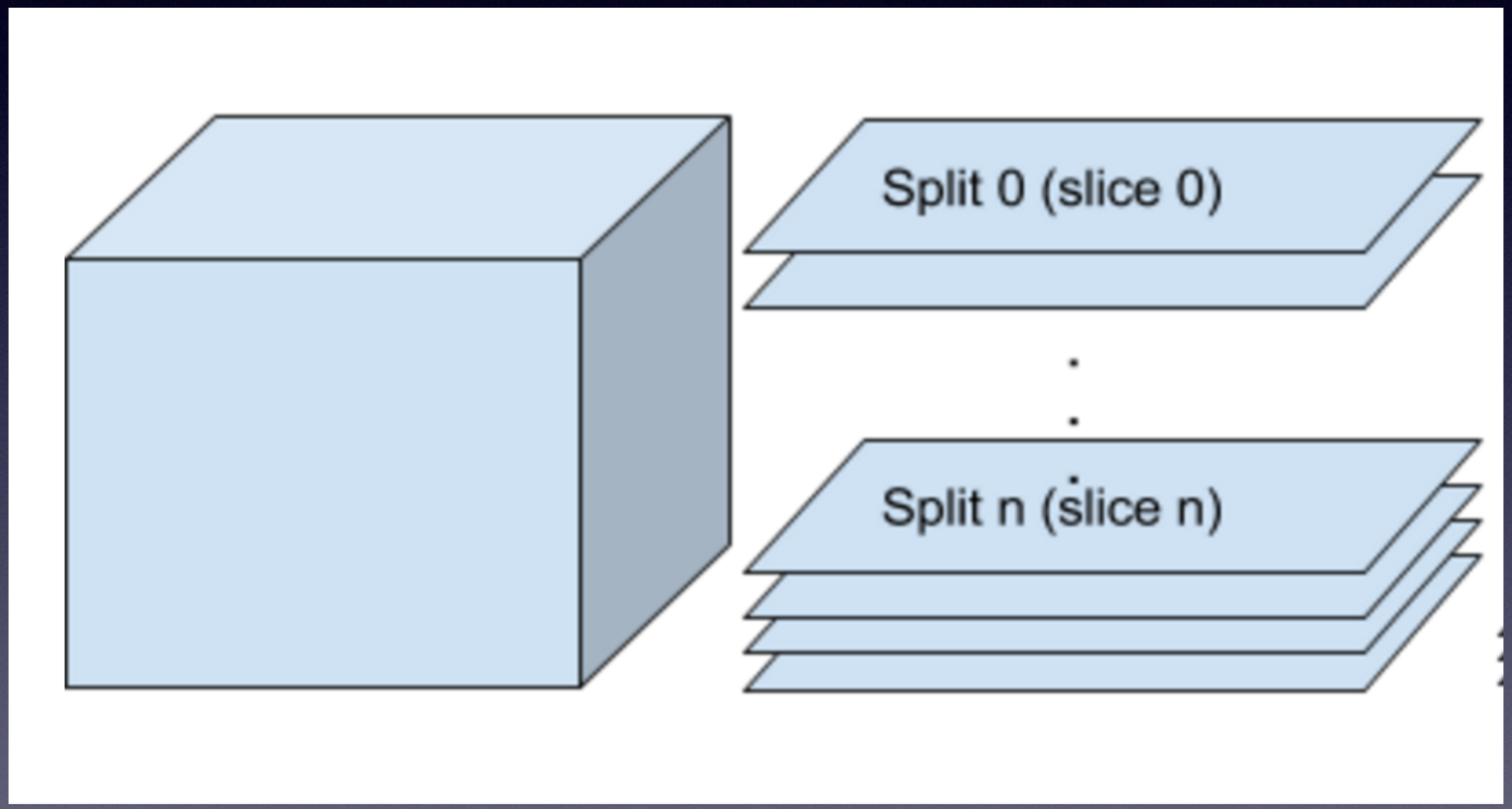


Design & Architecture



Implementation

- Loading and Distribution(**Default**)



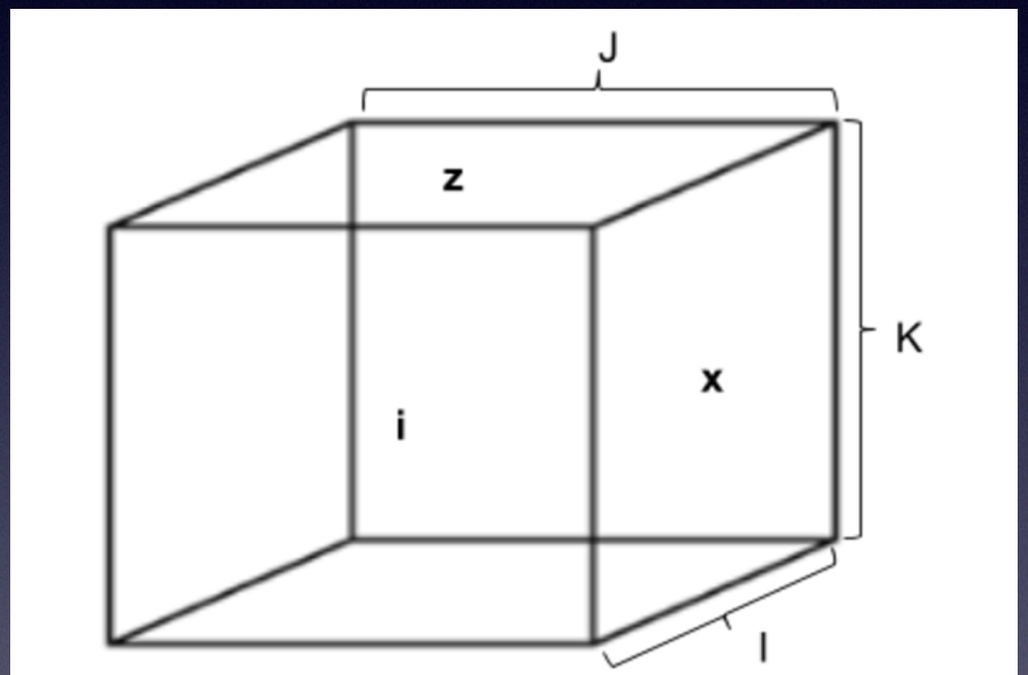
Implementation

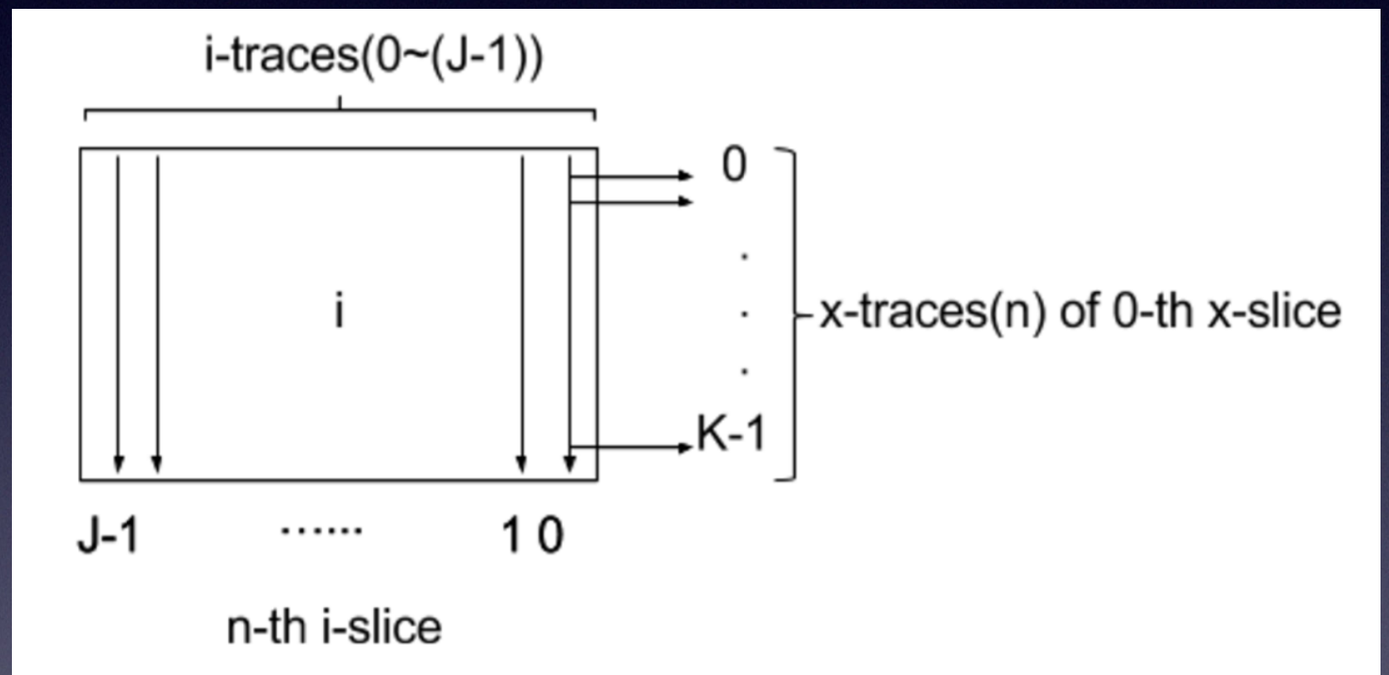
- Data Accessing

```
val sc = new SparkContext(new SparkConf());
val sv = SeismicVolume.loadFromFile[Float](sc, seismicXml);

sv.getLine(1, 0);
sv.getLine(2, 0);
sv.getLine(4, 0);
```

Implementation

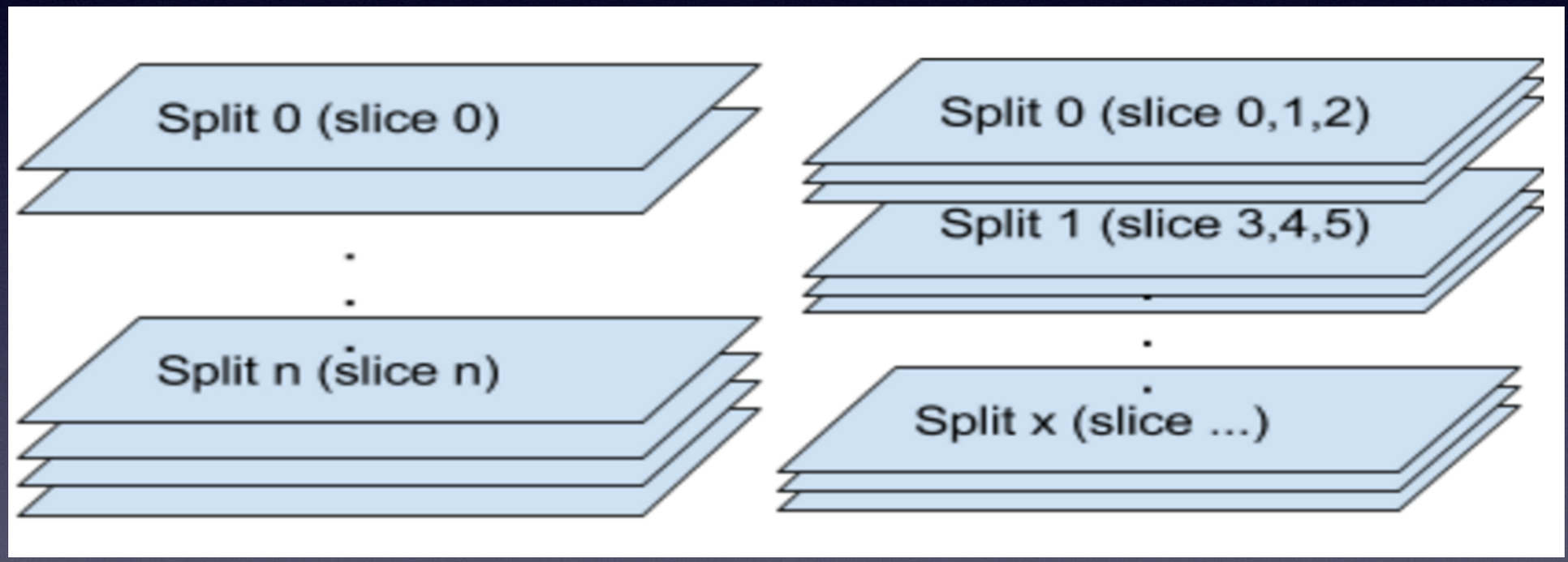
- 3D Transposing
  - (Implemented by RDD flatmap, group and sort operations)





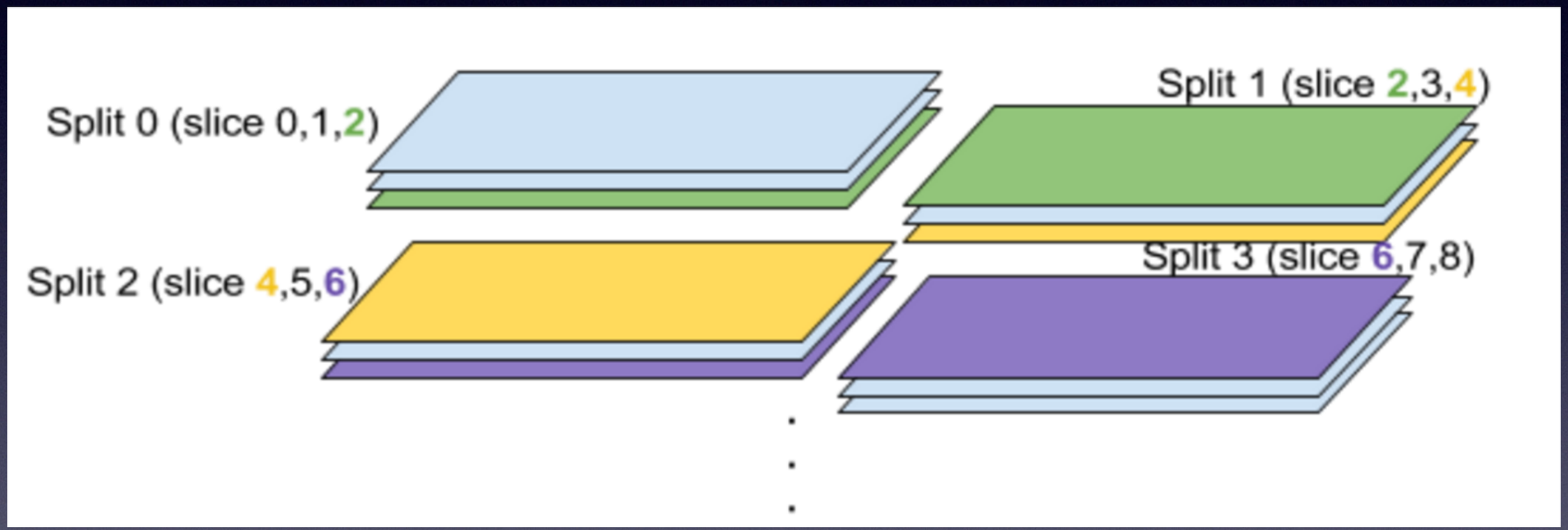
Implementation

- repartition(**aggregate**, overlap):SeismicVolume[T]



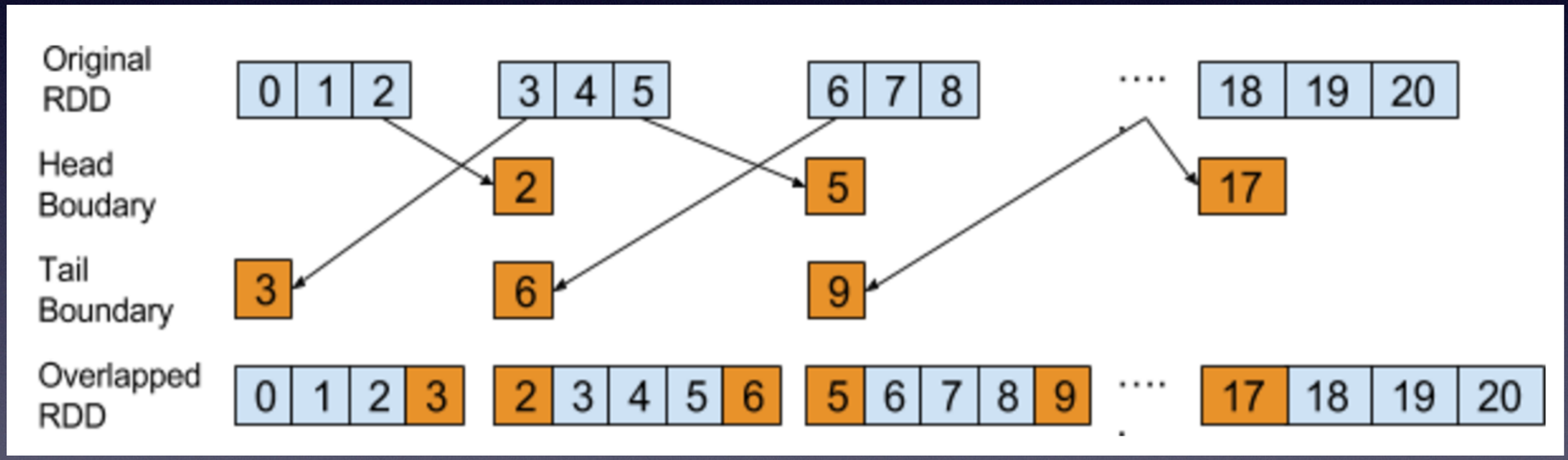
Implementation

- repartition(aggregate, **overlap**):SeismicVolume[T]



Implementation

- Create the Overlapping from Boundary RDDs



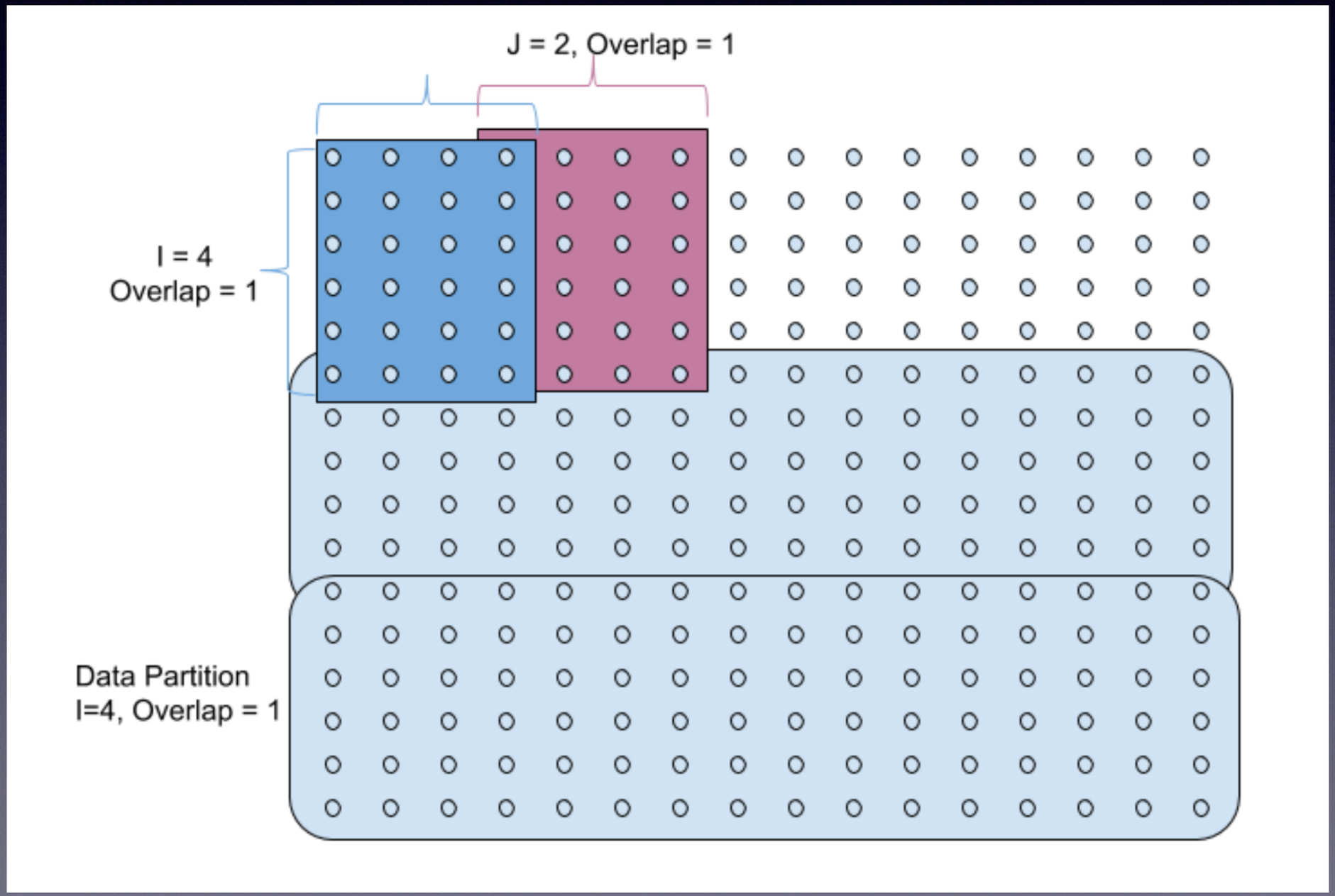
Implementation

- applyMap(dir, grain, **func**):SeismicVolume[T]

```
def fftFunc[T:ClassTag](e:(Long,Array[T])):(Long,Array[T]) = {
    val k = e._1;
    val v = e._2;

    var w = kVar;
    var h = jVar;
    if (dir == 2) {
        w = kVar;
        h = iVar;
    }
    else if (dir == 4) {
        w = jVar;
        h = iVar;
    }

    val planeSize = w * h;
    val num = v.length/(planeSize);
    var res = new Array[T](0);
    for(i <- 0 until num) {
        val line = new DenseMatrix(h, w, v.slice(planeSize*i,planeSize*i+planeSize));
        val lined= line.mapValues(a=>a.asInstanceOf[Double]);
        val fft = fourierTr(lined);
        val tmp = fft.mapValues(a=>a.abs.asInstanceOf[T]).data;
        res = res ++ tmp;
    }
    (k,res)
}

println("start to apply FFT function in dir " + dir + ": " + Calendar.getInstance().getTime());
val newsv = sv.applyMap(dir, 0, fftFunc);
println("start to save result as : " + path + " " + args(3) + " " + Calendar.getInstance().getTime())
newsv.save(path, tp);
```

Parallel Templates

- 3D Stencil Problem



Parallel Templates

- Template of Sub-volume with Overlapping

```
class TraceOlpFilter(override val inputList:Array[SeismicVolume[Float]], override val distPrm:OverlapDistParams)
    extends SeismicAppTraceOverlapTmpl[Float,Float](inputList, distPrm)
{
    override def proc(inputData:Array[Array[Array[Array[Float]]]],
                              olpWindow:TraceOverlapWindow):Array[Array[Array[Array[Float]]]] = {

        val len = inputData.length;
        val outData = new Array[Array[Array[Array[Float]]]](len);

        for (i <- 0 until len) {
            val centerVolume = new Array[Array[Array[Float]]](olpWindow.olpCenterI);

            /*extract center data from olp window*/
            for (o1 <- olpWindow.olpHeadI until (olpWindow.olpHeadI + olpWindow.olpCenterI)) {
                val centerSlice = new Array[Array[Float]](olpWindow.olpCenterJ);

                for (o2 <- olpWindow.olpHeadJ until (olpWindow.olpHeadJ + olpWindow.olpCenterJ)) {
                    val trace = inputData(i)(o1)(o2);
                    centerSlice(o2 - olpWindow.olpHeadJ) = trace;
                }
                centerVolume(o1 - olpWindow.olpHeadI) = centerSlice;
            }
            outData(i) = centerVolume;
        }
        outData;
    }
}
```

Parallel Templates

- Run a template

```
/*create an overlap layout:
I x J: 26 x 21 in center and with 4 traces overlapping in each of I and J direction
*/
val olpSpec = new OverlapDistParams(26, 21, 4, 4);

val olpTmpl = new TraceOlpFilter(invols, olpSpec);

// Run the filter to produce filtered volume
val fvol = olpTmpl.exec(INLINE);
val res = fvol(0);
```

Scalability & Performance

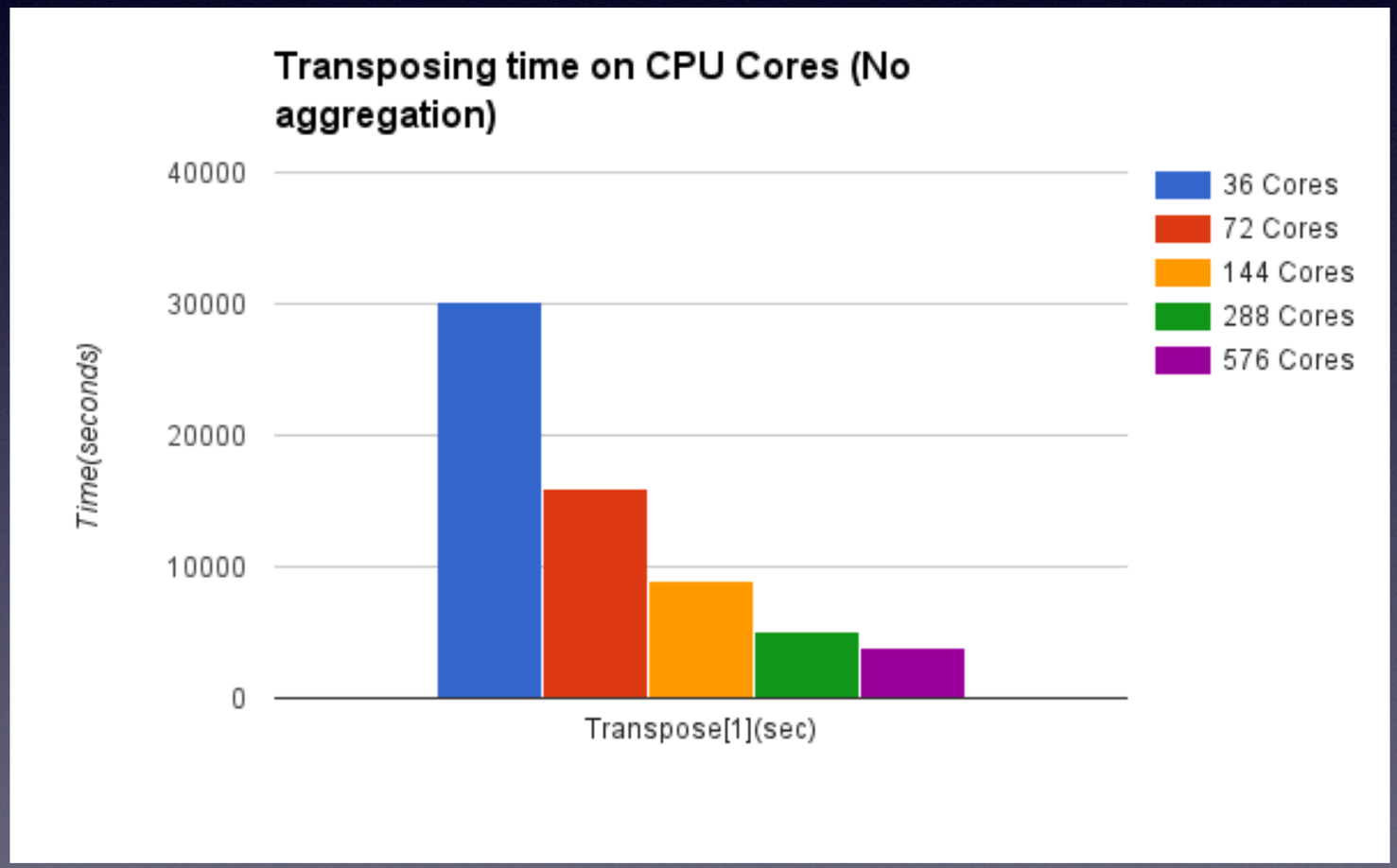
- PVAMU Cloud (288/576 Cores)
  - 24 Nodes
  - Each Node:
    - CPU:

      Intel Xeon E5-2640 Sandy Bridge

      2.5 GHz, 12 Cores (24 in Hyper-threading)
    - 64 GB DDR3

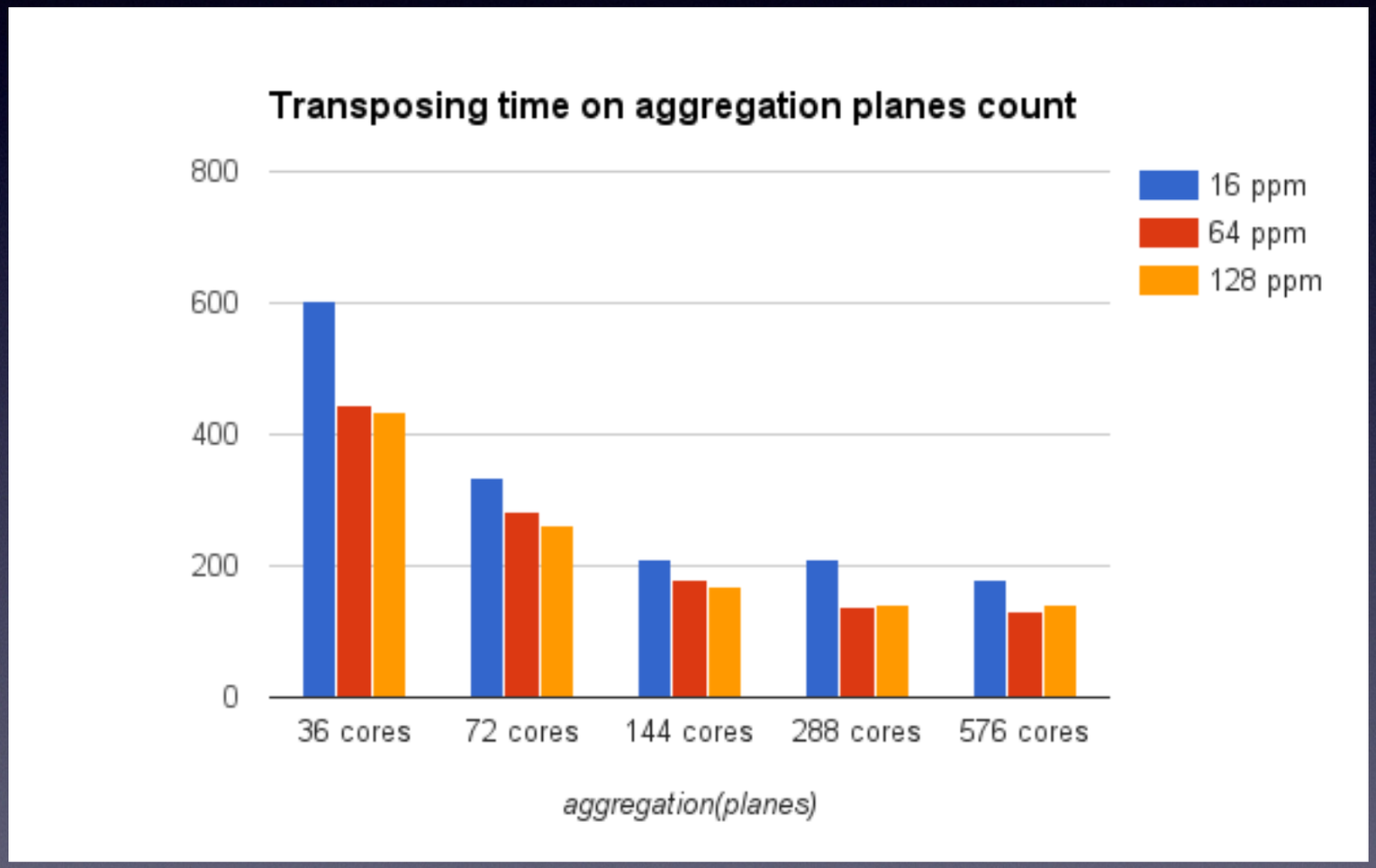
Scalability & Performance

- Transpose (Default distribution)



Scalability & Performance

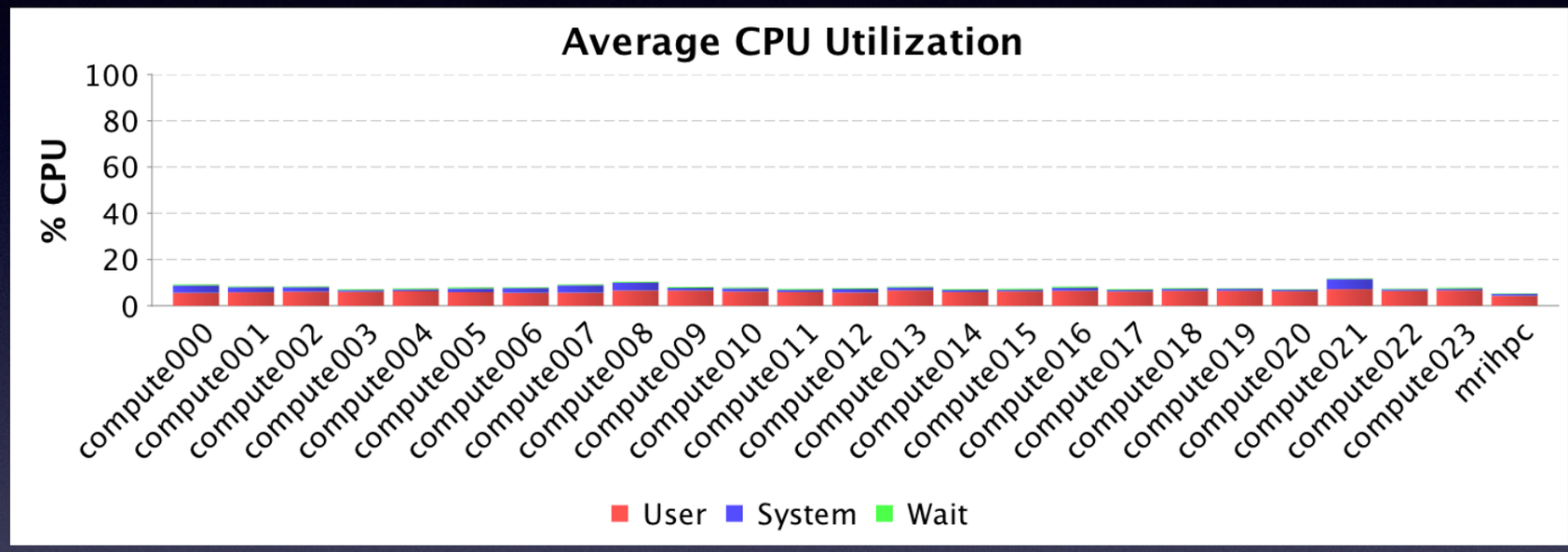
- Transpose (Across different aggregations)

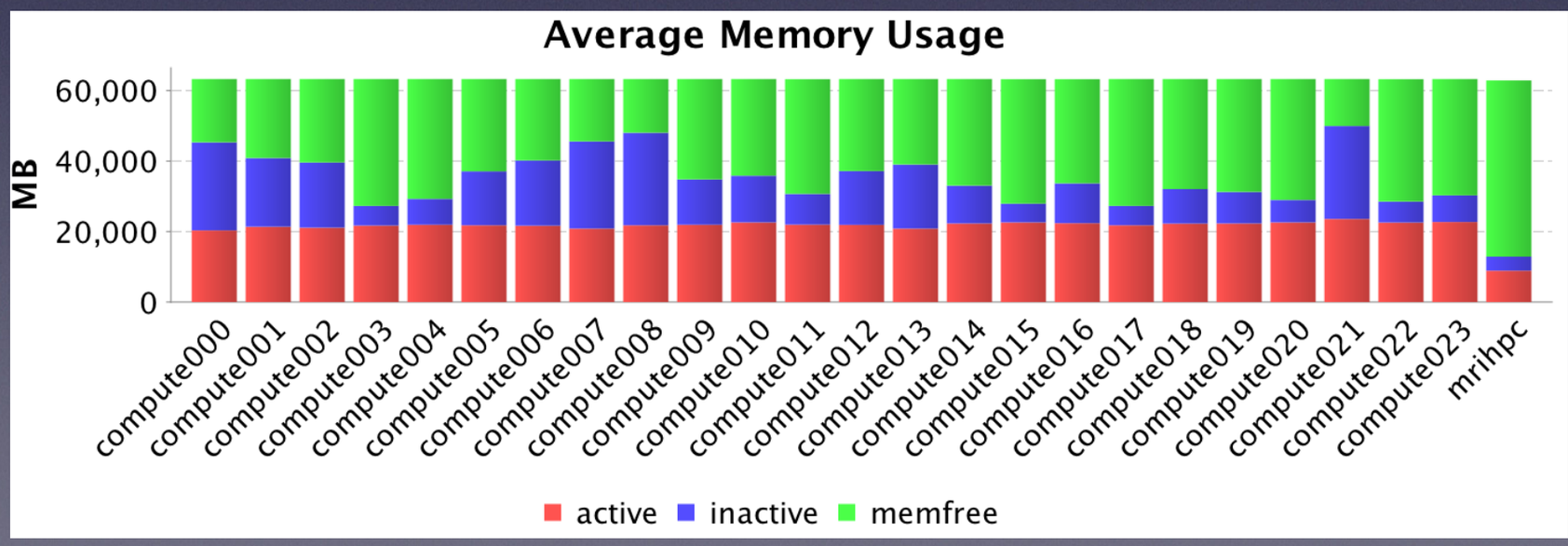


ppm - planes per map

Scalability & Performance

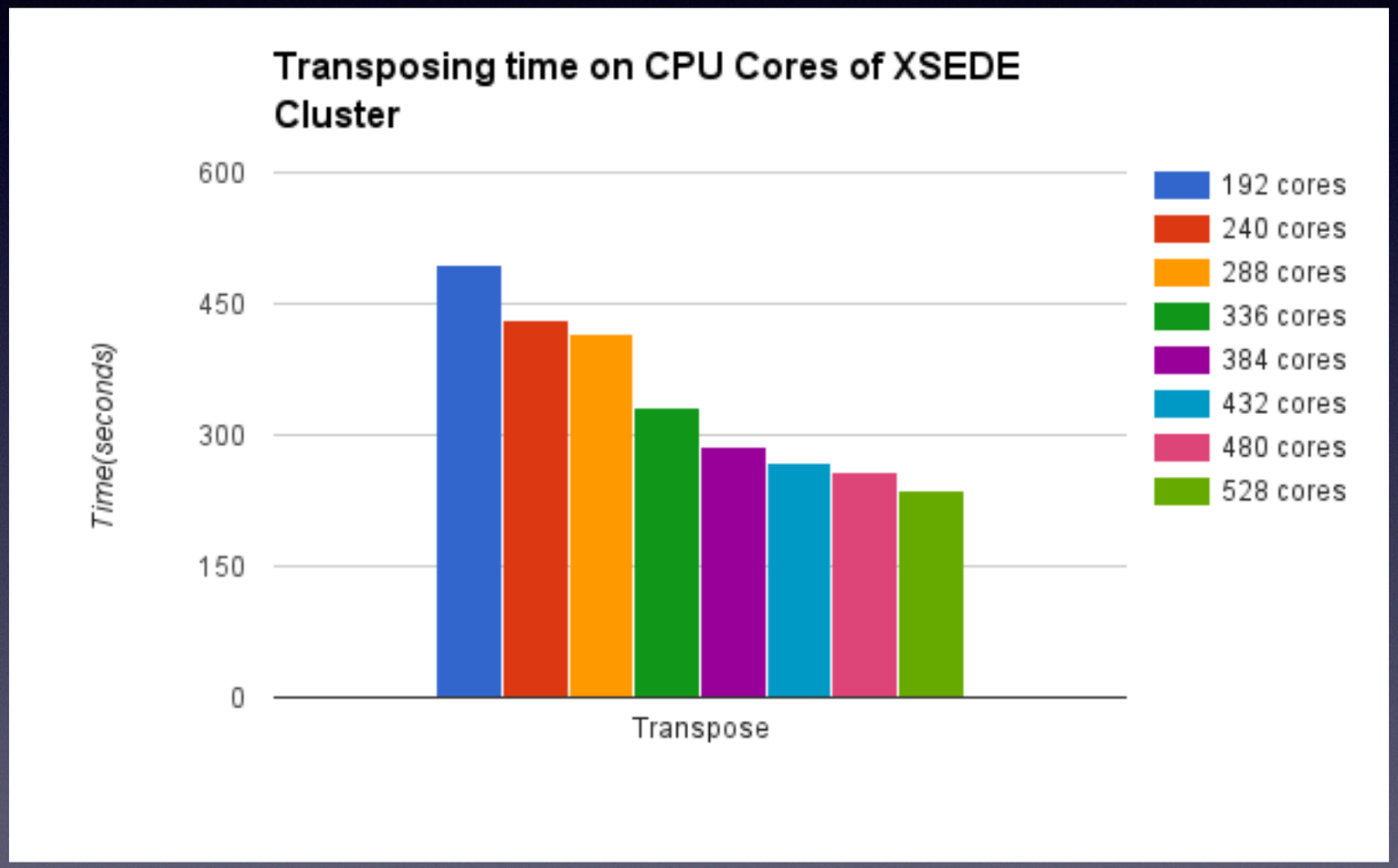
- Distributed Resource Utilizing





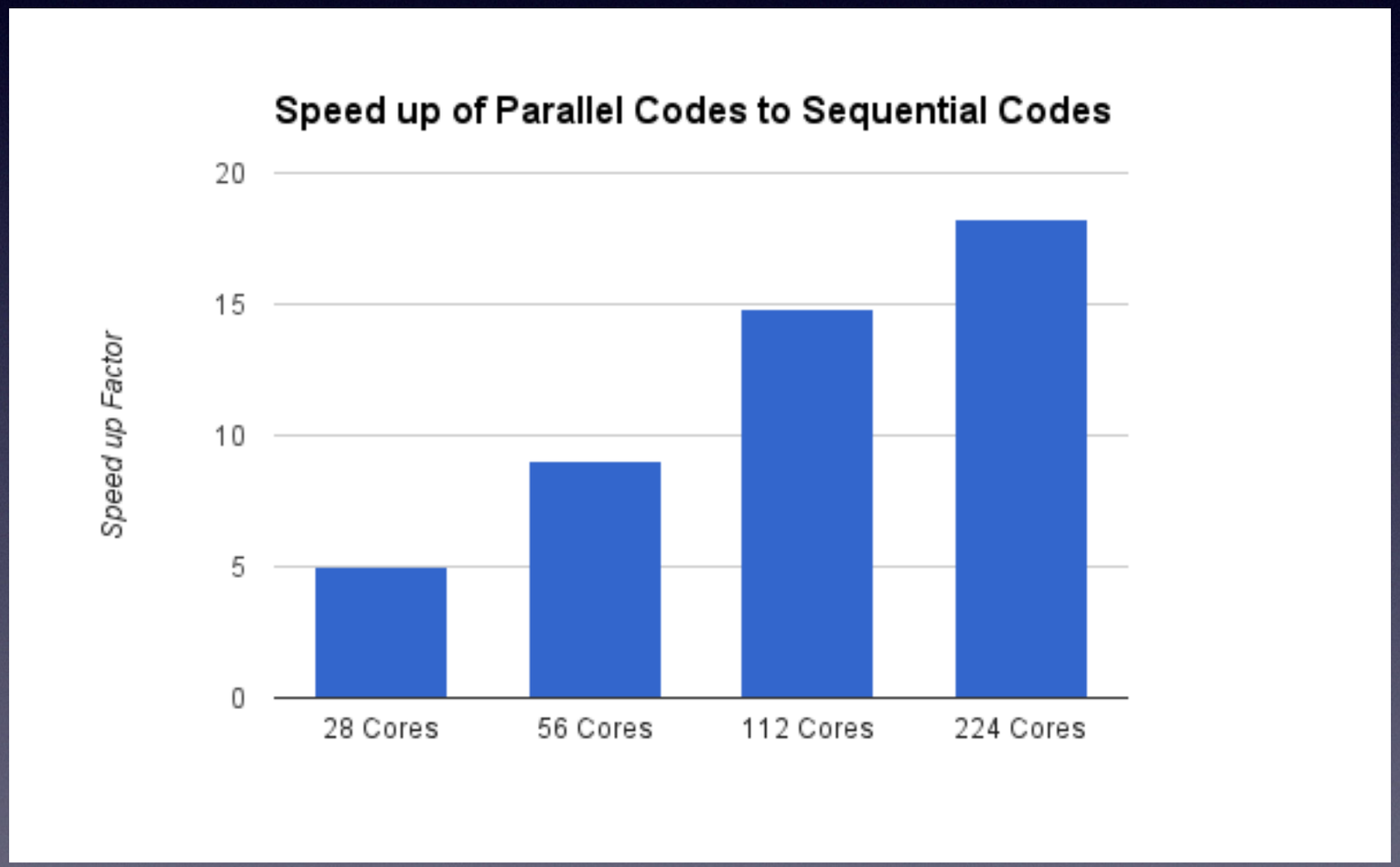
Scalability & Performance

- Transpose on XSEDE Cluster (www.xsede.org)



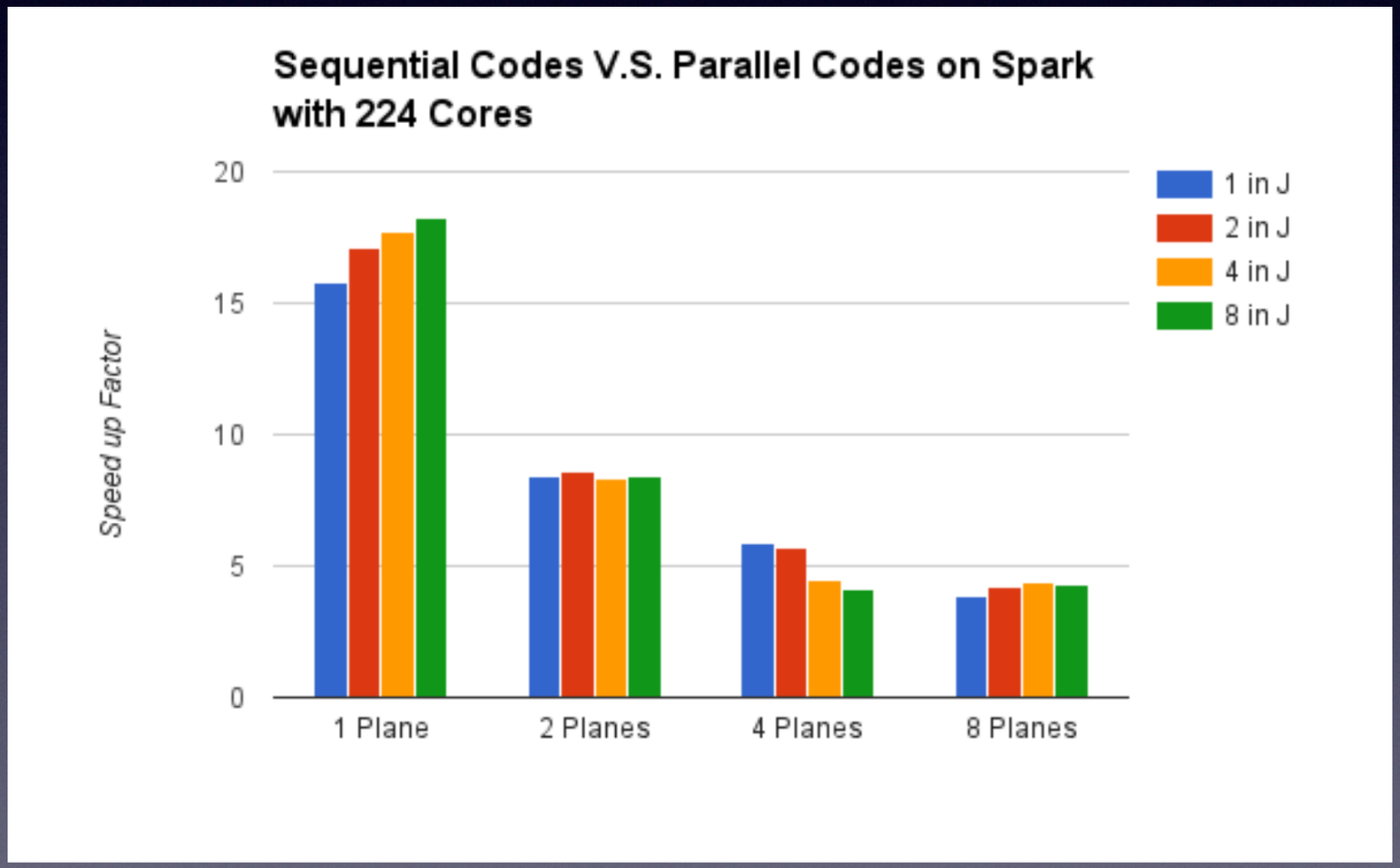
Scalability & Performance

- 3D Stencil Performance (Template Use case)



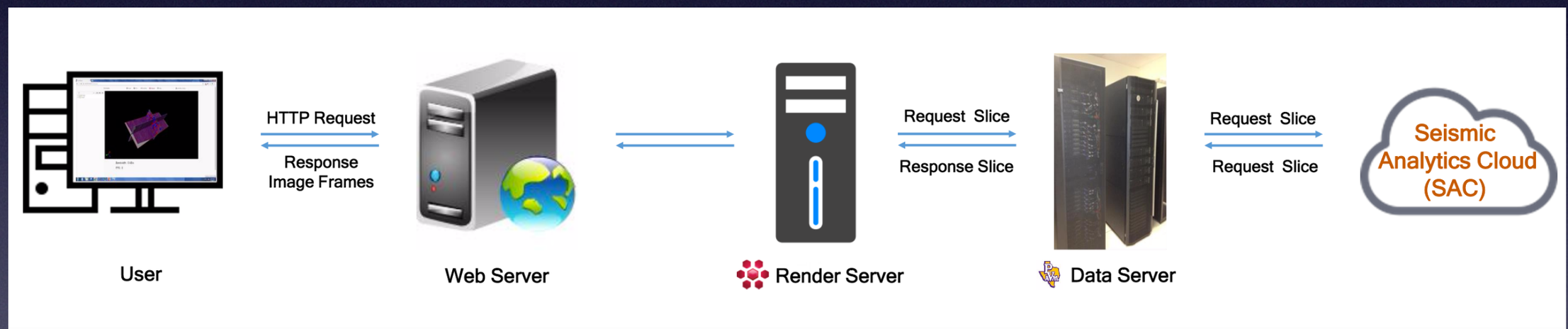
Scalability & Performance

- 3D Stencil Performance (Template Use case)



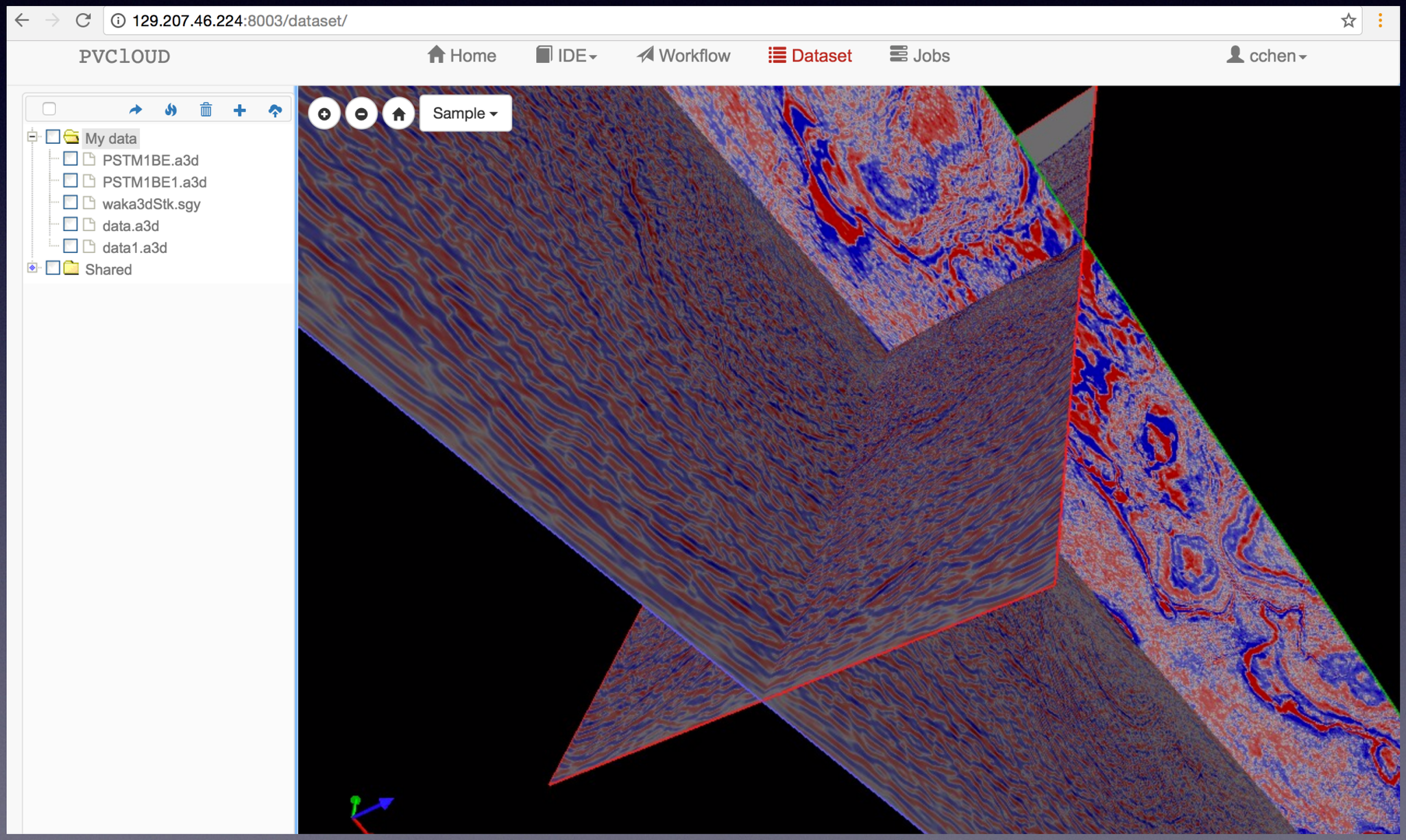
Cloud Platform Services

- Data Server and Remote Visualization



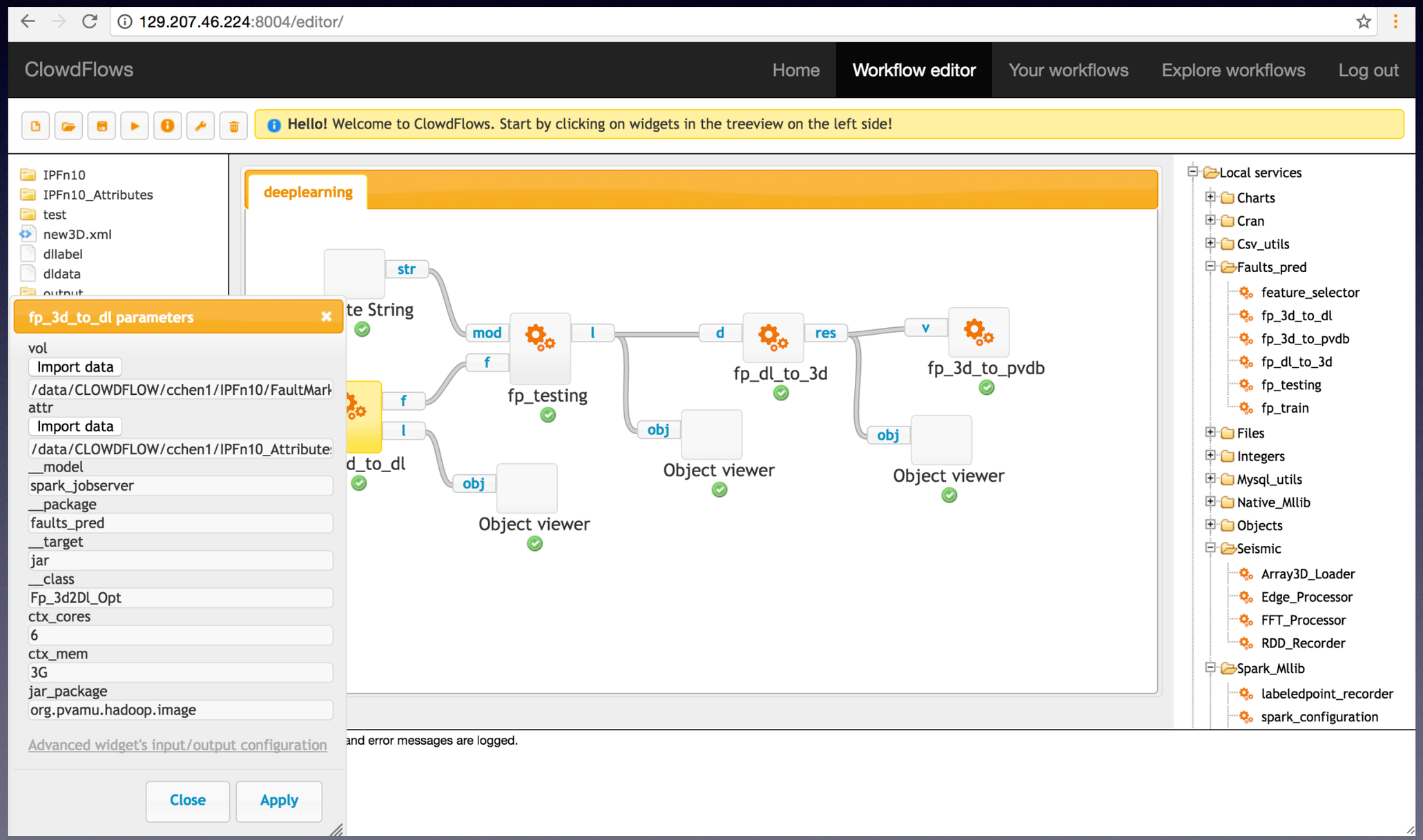
Cloud Platform Services

- Data Server and Remote Visualization



Cloud Platform Services

- Workflow



Demo

Thanks