

# Big Data Status (Draft)

Jim Kowalkowski

## 1 Introduction

The purpose of this document is to give a brief comparison of the three LEM solutions that have been proposed, worked on since early 2015, and demonstrated to some degree. In this document, each solution will be referred to as a demonstration system. The three demonstration systems are: (1) Spark, (2) Aerospyke, and (3) MPI. Section 7 provides two views for comparing features of the demonstration systems. This document will also provide an overview and status of the MPI demonstration system.

The Spark and Aerospyke systems utilize available Big Data technology to address the LEM question. The MPI system uses standard HPC technology to address it. The MPI system exists to form what I would a baseline solution so that the question of how much better is Big Data technology than a system constructed using well-established hand-coded distributed applications practices. MPI provides the facilities for quickly building the distributed application. The MPI system employs Big Data techniques: dataset held in a large distributed memory with the calculation active where the data is. It implements the algorithm using a variation of map-reduce, available through MPI primitive operations. The MPI system will also provide a data for calculating a total cost to build, deploy, and operate the final production system. All the underlying toolkits and libraries used in the creation of the MPI system are already supported by FNAL SCD, which makes estimating total cost straightforward.

The limitations of LEM as a Big Data problem were acknowledged from the beginning of the project. The expression of the analysis (the calculation) is fixed i.e. the question asked of the dataset is always the same. The dataset being processed is secondary: it is a fixed library used by the primary data stream being processed. There is no variety across analysis queries made to this system: all multi-user queries are the same.

There are known deficiencies in the dataset that we have to work with. It is 77M events with particle type distributions that match reality, but not what is needed for the LEM library. Production of the full LEM dataset with uniform particle distributions has not been started.

The current ROOT-based LEM algorithm that is used in the production system was never made available in a test system. This test system is necessary to verify that algorithm implementations and system operations in the demonstration systems are giving correct answers. It is also needed to have a standard performance number to compare with.

## **2 Current status**

Section 8 shows the configuration of the MPI demonstration system. The code for this system is in a redmine repository<sup>1</sup> under the cpp directory. The 77M event JSON dataset spread over 200 files was pulled from the NOvA cluster and converted into a condensed custom binary format and placed onto cluck.fnal.gov. There are a number of utilities in the cpp directory for approximately event distributing of the 200 files to the five grunt nodes. Each grunt has 40 files of approximately the same total size. The main2.cc file contains the application in its current state. The program will automatically calculate which files must be opened by each MPI process based on rank and node. Each MPI process reads all the event data into memory and creates a sorted index by number of hits and metadata attribute theta1. These steps are shown in the diagram below above “initialize”.

As depicted in the diagram, a head node broadcasts on an event to be matched. Each process does not examine its entire dataset; it reduces the range by looking at the number of hits and the theta1 attribute. The program produces histograms of number of hits and theta1. These histograms were being used to determine reasonable range sizes.

The LEM algorithm from the lembigdata/python area was recoded in C++ using the Armadillo library. This code is available in the cpp directory. It is not integrated into the processing stream of main2.cc.

## **3 What needs to be completed?**

## **4 What R&D activities need to be completed for the LEM problem?**

NA.

## **5 How does the MPI application compare with the other solutions?**

NA.

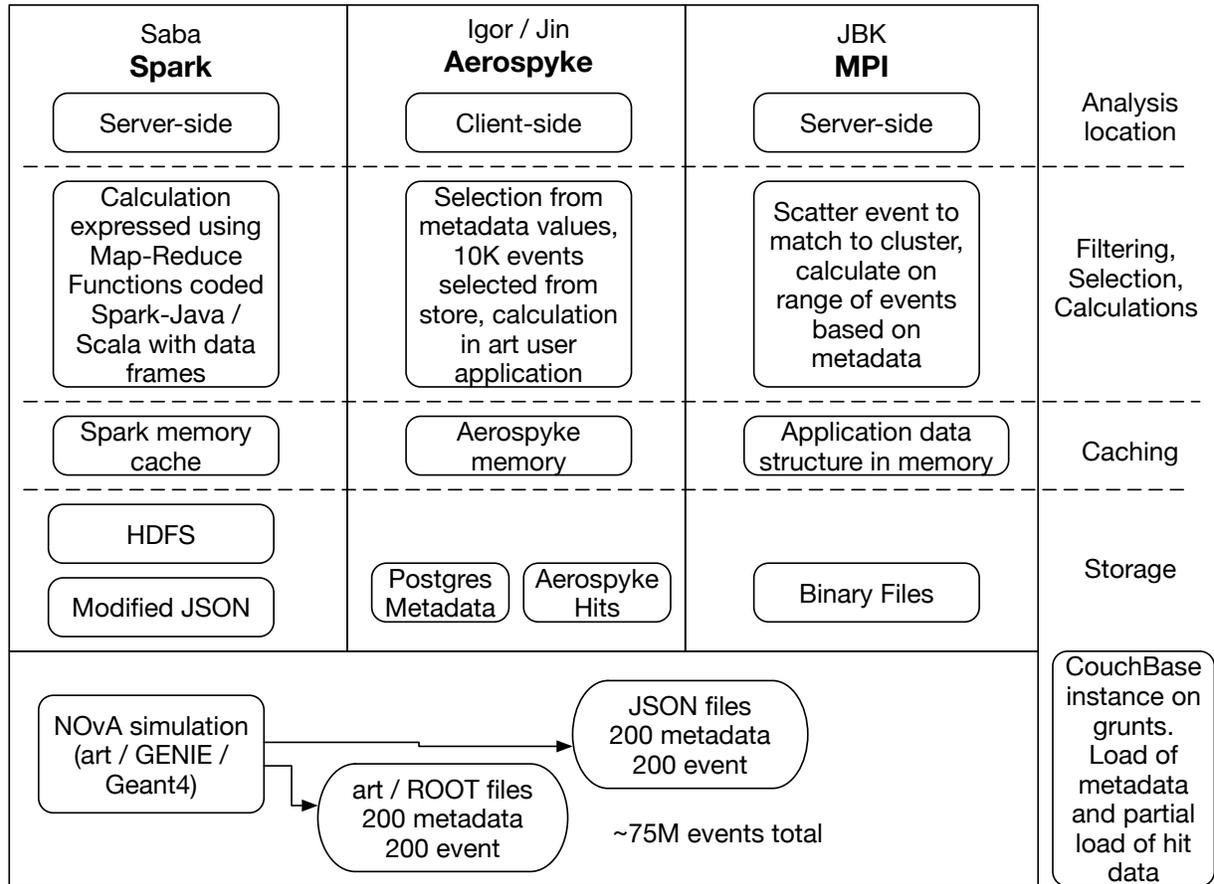
## **6 How does this fit into the bigger Big Data picture?**

NA.

---

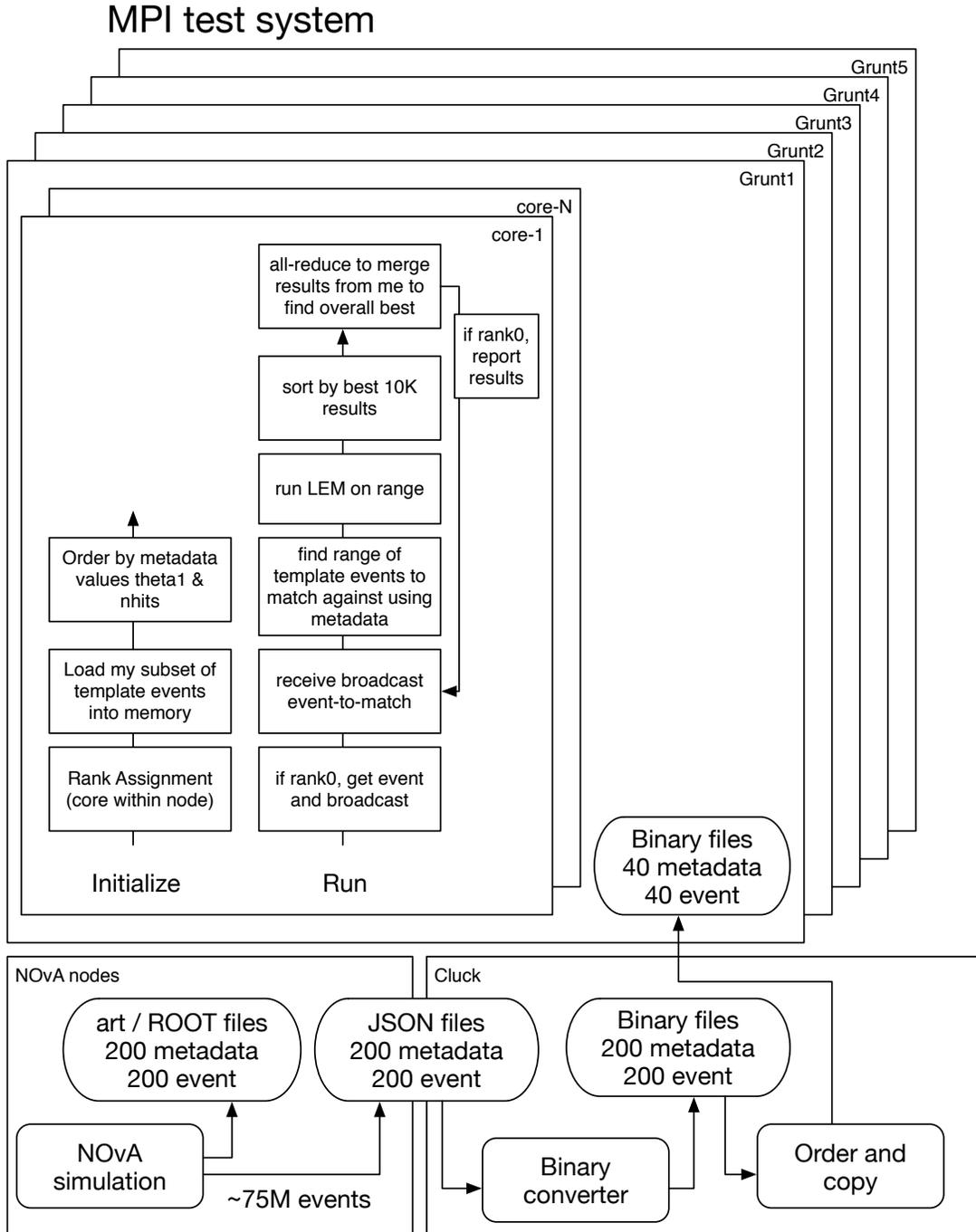
<sup>1</sup> <ssh://p-lembigdata@cdcvs.fnal.gov/cvs/projects/lembigdata>

## 7 View of ongoing work



Common features	Spark	Aerospyke	MPI
Distributed calculation	Spark map-reduce	Client application	MPI application
Locate / query events	Spark / Java	Client application	MPI application
Filter	Spark / Java	Postgres	MPI application
Map into memory	Spark system	Aerospyke	MPI application
Storage	HDFS	Aerospyke / post-gres	Binary files

## 8 View of MPI test system



## 9 Original system view

