MARISSA: MapReduce Implementation for Streaming Science Applications

Elif Dede, Zacharia Fadika, Jessica Hartog, Madhusudhan Govindaraju

❖Lavanya Ramakrishnan, Dan Gunter, Shane Canon

Department of Computer Science, Binghamton University (SUNY)

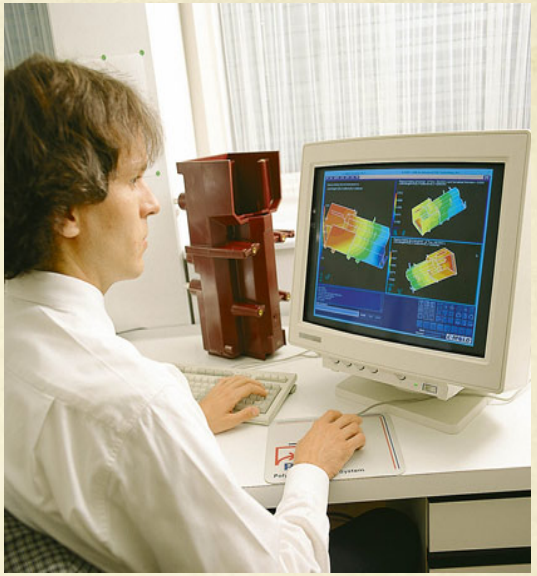
❖Lawrence Berkeley National Laboratory

BINGHAMTON UNIVERSITY
State University of New York

NERSC
National Energy Research Scientific Computing Center

BERKELEY LAB
Lawrence Berkeley National Laboratory

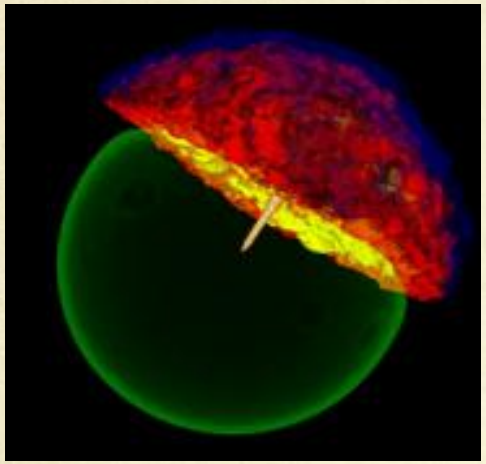
Evolution of Practice of Science

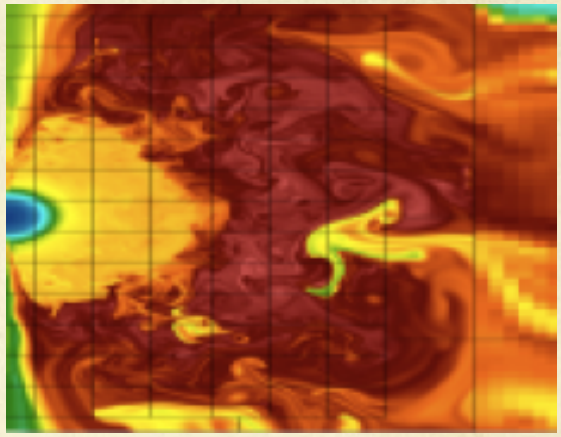


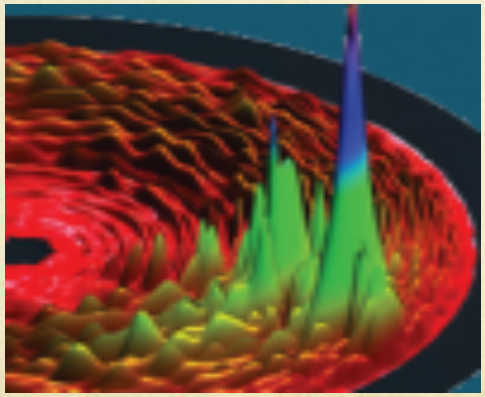
What do computers bring to the scientific world?

3-4 physical experiments or running hundreds of simulations?

Scientific Simulations on NERSC Systems



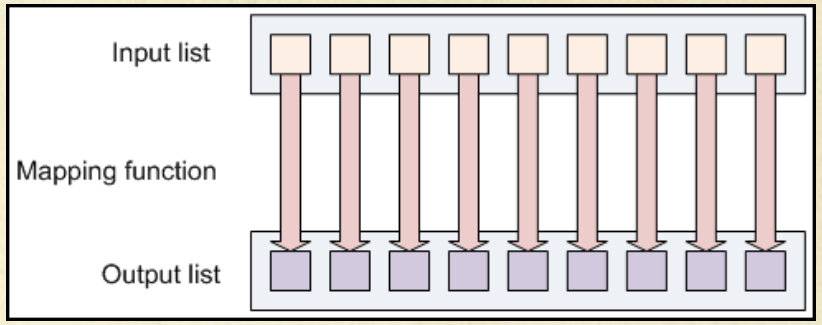


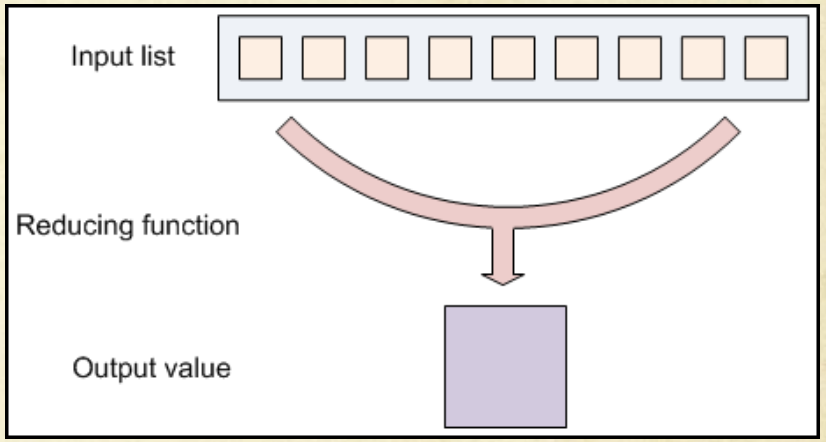


- Chemistry
- Material Science
- Climate and Earth Sciences
- Life Sciences
- Theoretical astrophysics and cosmology

Data Intensive Computing: MapReduce

- Introduced in OSDI 2004 by Dean and Ghemawat from Google
- Programming model for processing large data sets
- Exploits large a set of commodity machines
- **Characteristics of the model:**
  - Relaxed synchronization constraints
  - Locality optimization
  - Fault-tolerance
  - Load balancing

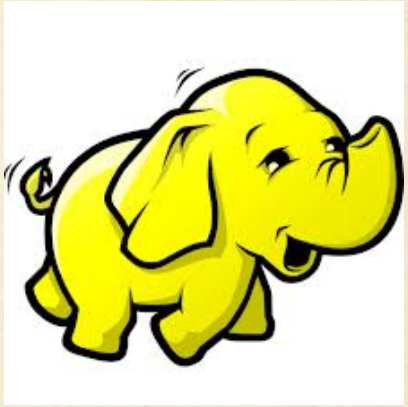




OSDI 2004

Apache Hadoop

- **Open-source MapReduce implementation in Java**
- **Easy scalability**
- **Built-in I/O management**
  - Hadoop Distributed File System(HDFS)
    - Data distribution, management and replication
- **Load balancing**
  - Handles stragglers
- **Fault tolerance**
  - Commodity hardware
  - Heartbeats
  - Speculative execution and data replication

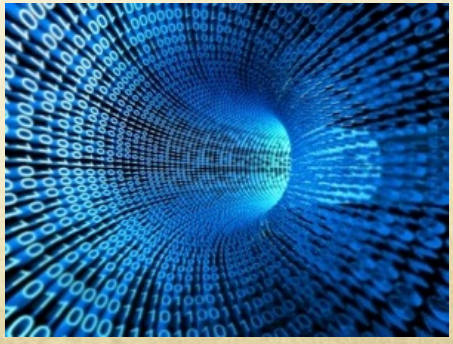


- **Hadoop Streaming**
  - **Create and run MapReduce jobs with any executable or script as the mapper and/or the reducer**

Scientific Computing and Hadoop

Hadoop provides:

- Data Flow Parallelism
  - Data goes through different steps of processing
- Similar Job Phases
  - Data preparation, transformation and reduction
  - MapReduce: maps (transformation) and reduces (reduction)
- Number of maps >>> Number of reduce
  - Data transformation is mostly more parallel then data reduction
- Fault Tolerance and Data Locality
  - Data intensive loads
  - Long running scientific jobs



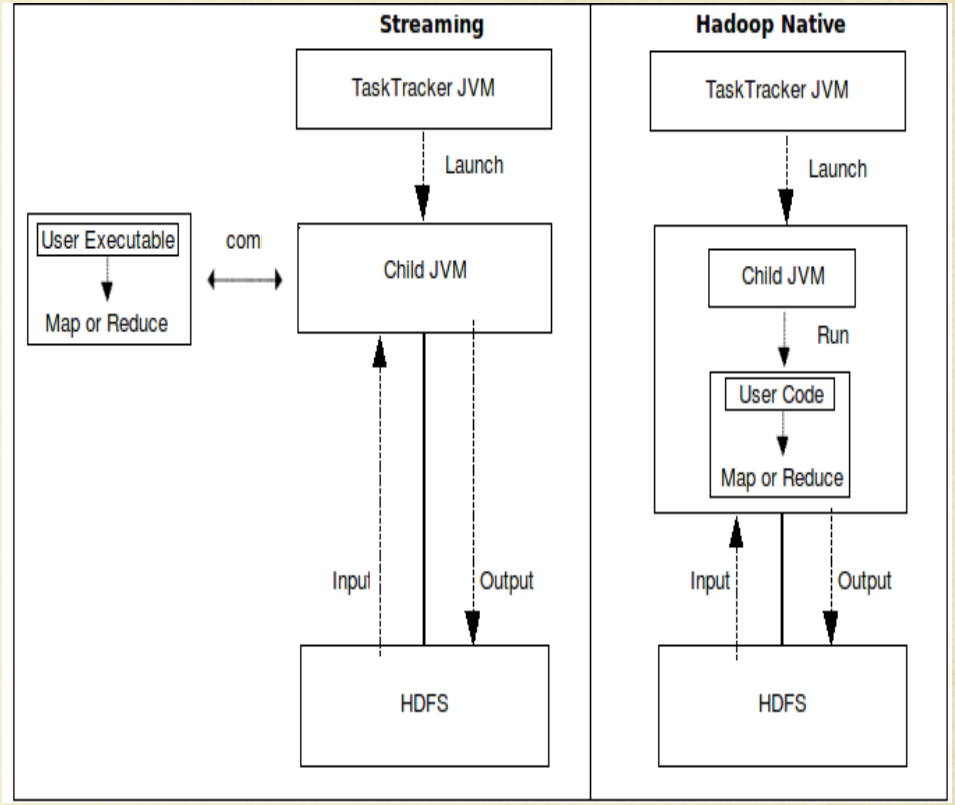
Scientific Computing and Hadoop (Cont.)

Hadoop does not provide:

- Java implementation
  - Legacy scientific code mostly is not in java and hard to rewrite as map and reduce functions
  - **Hadoop Streaming** allows other modes – but restricted model (next slide)
- HDFS is a non-POSIX file system
  - HDFS java library calls needed to create, read and write files
- Scientific data formats do not fit in the line-oriented inputs of typical Hadoop jobs
  - Scientific applications tend to work with an input file rather than a line
- Maps and reduces are considered identical
  - Identical executables with identical arguments
- No built-in dynamic and iterative support

Hadoop Streaming Challenges

- Hadoop Streaming requires STDIN/STDOUT
  - HDFS cannot be read or written to without the HDFS API calls
  - Scientific applications tend to work with files rather than STDIN/STDOUT
- Hadoop lacks of support for individual input files
  - Scientific applications tend to work with files rather than input lines
- Hadoop Streaming imposes a **performance penalty**
  - Experience shows poor processing performance



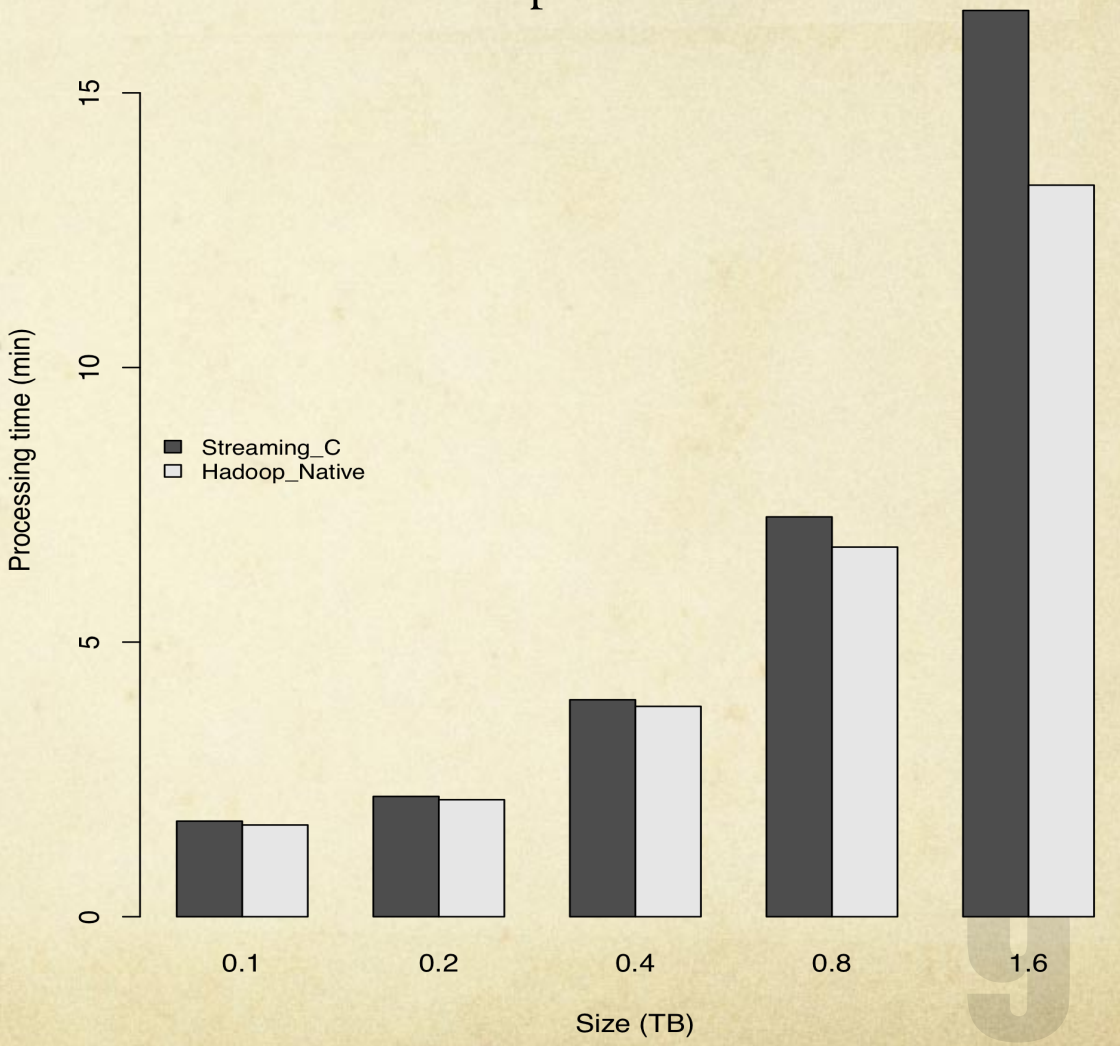
Hadoop vs Hadoop Streaming

On single node:

Input Size (TB)	C (min)	Java (min)
0.1	98.67	127.01
0.2	198.25	251.93

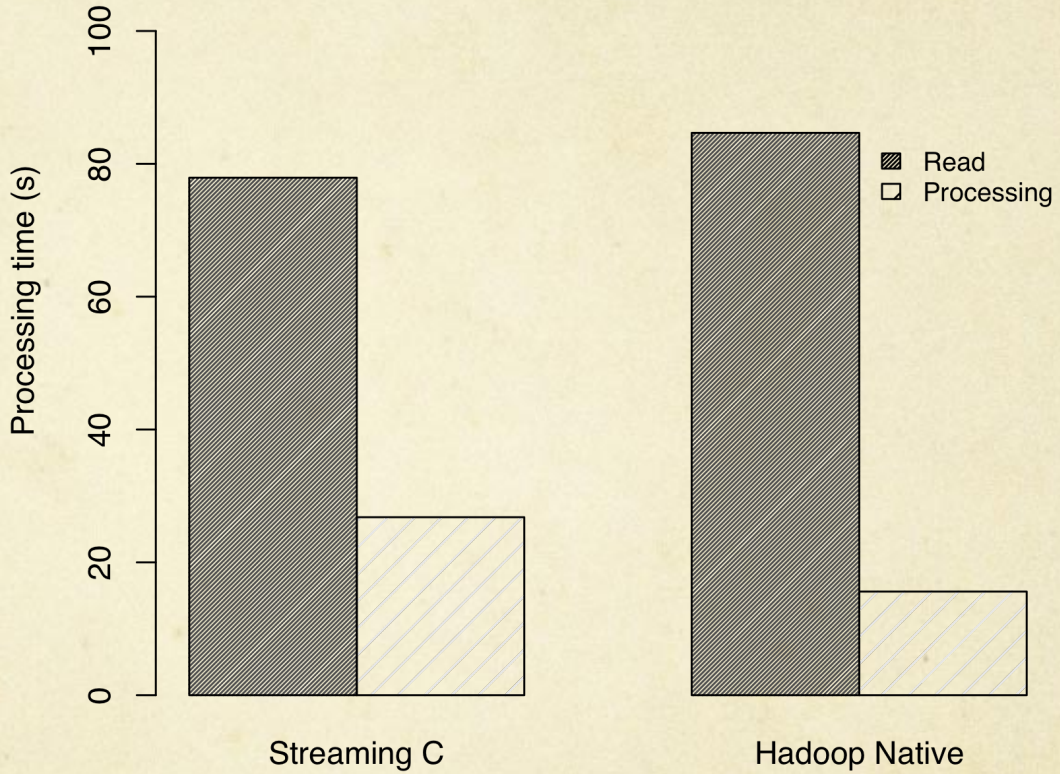
- The 'C" vs Java baseline experiments in the table:
  - "C" was 20% faster than Java for the same application
- Streaming counterpart: At 1.6TB the "C" streaming version is about 19% slower than Hadoop native.

On Hadoop cluster:

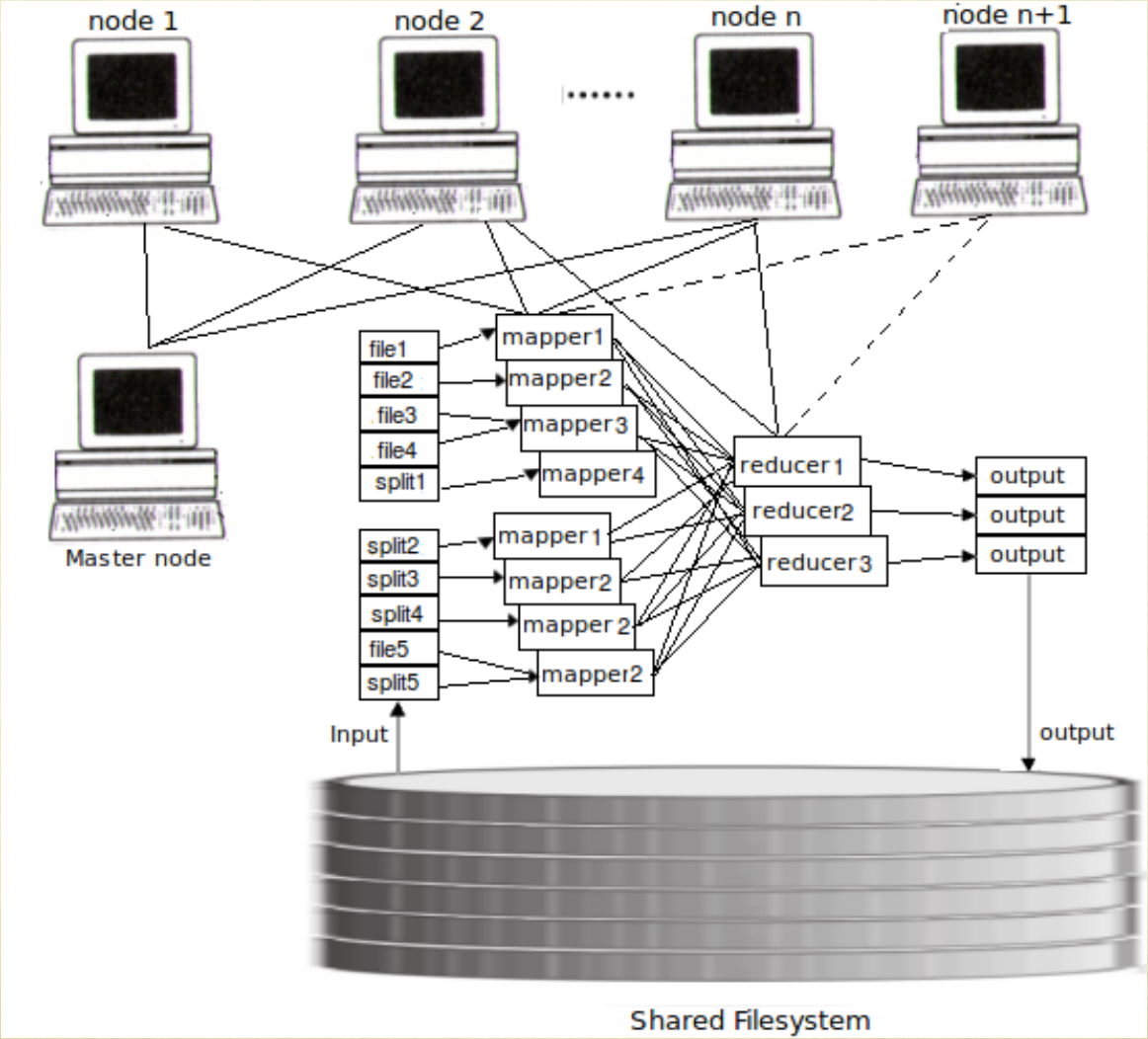


Hadoop vs Hadoop Streaming (Cont.)

- This graph shows processing of 1.6TB of Wikipedia data
- Filtering to isolate embedded indices using 75 nodes in both cases
- Streaming performance as not a factor of I/O speeds.
- In fact 'C" streaming performs faster I/O than Hadoop native.
- The performance losses occur in processing.



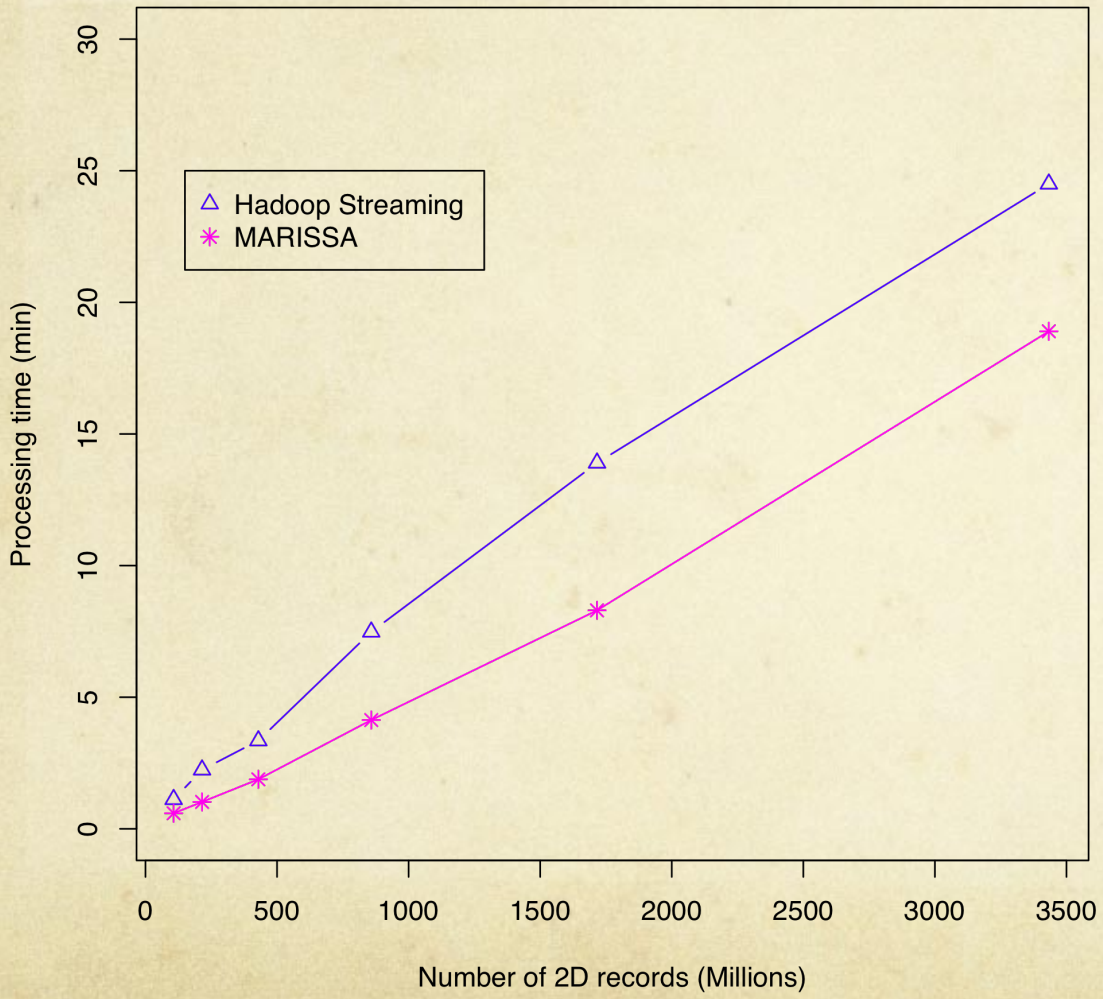
MARISSA: MApReduce Implementation for Streaming Science Applications



MARISSA: Architecture and Design

- MARISSA allows the use of the shared-disk file systems
  - The framework leverages the data visibility offered by the shared-disk file system
  - No need of extra layer like HDFS
- Providing input to applications through standard I/O and through file access
  - STDIN/STDOUT is not a must
  - Scheduling each node to spawn a process tied to the specified input file or dataset
  - Model permits redundant task execution
- Each worker can execute different binaries
  - Same binary can work with different arguments
- A node specific fault-tolerance mechanism, rather than an input specific one
  - Node failure is inherently decoupled from data availability
  - Re-execution simply consists of task re-assignment

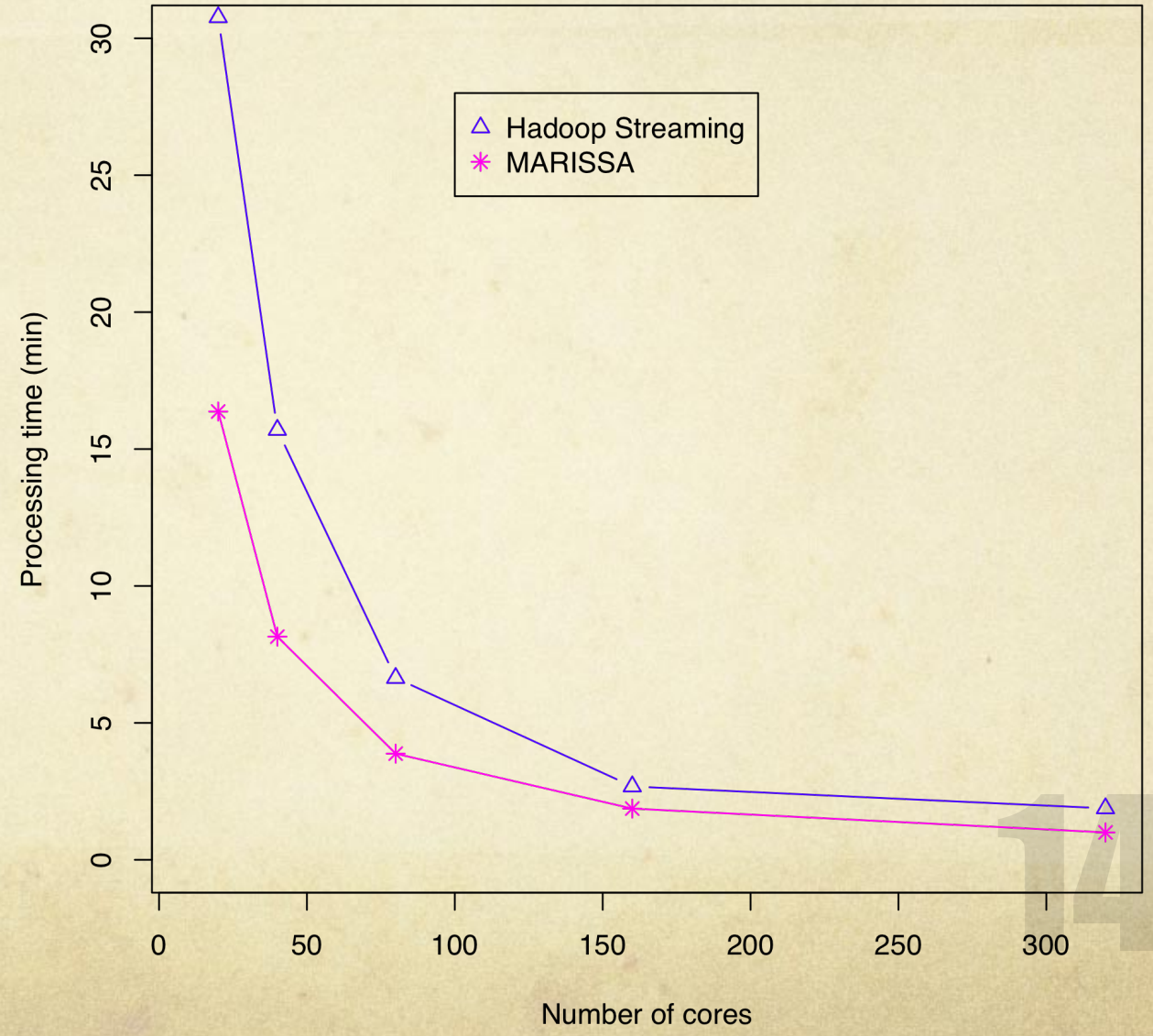
Hadoop Streaming vs MARISSA



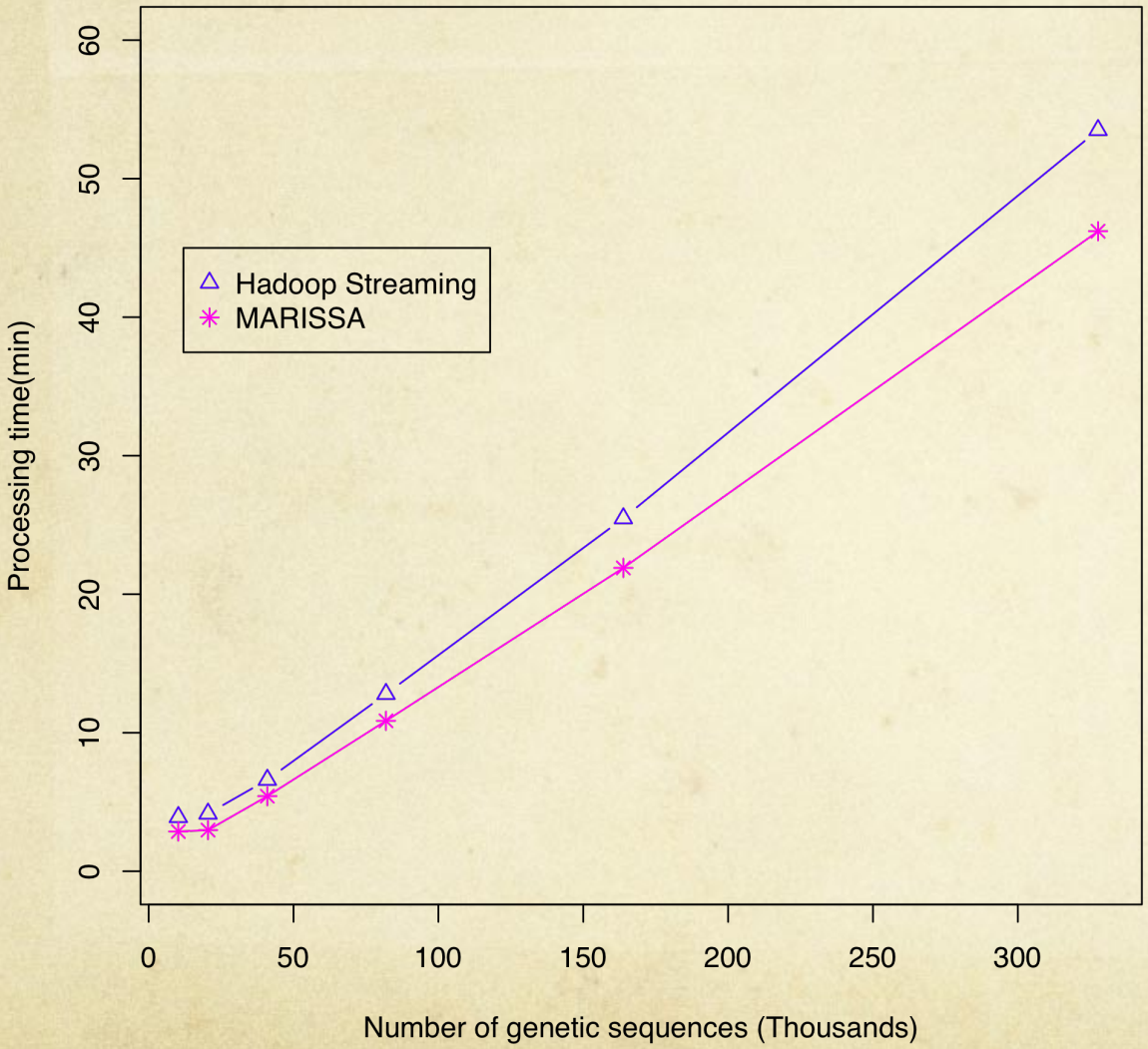
- 80-core Hadoop and MARISSA clusters
- K-means clustering on 0.1 Billion to 3.4 Billion 2D-records
- MARISSA here, given the CPU-intensive nature of the application performs up to 47% faster

Hadoop Streaming vs MARISSA (Cont.)

- K-means clustering
- 858 million 2D-records cluster sizes ranging from 20 to 320 cores
- MARISSA performs up to 46.8% faster than with 20 cores, and up to 47% with 400 cores.



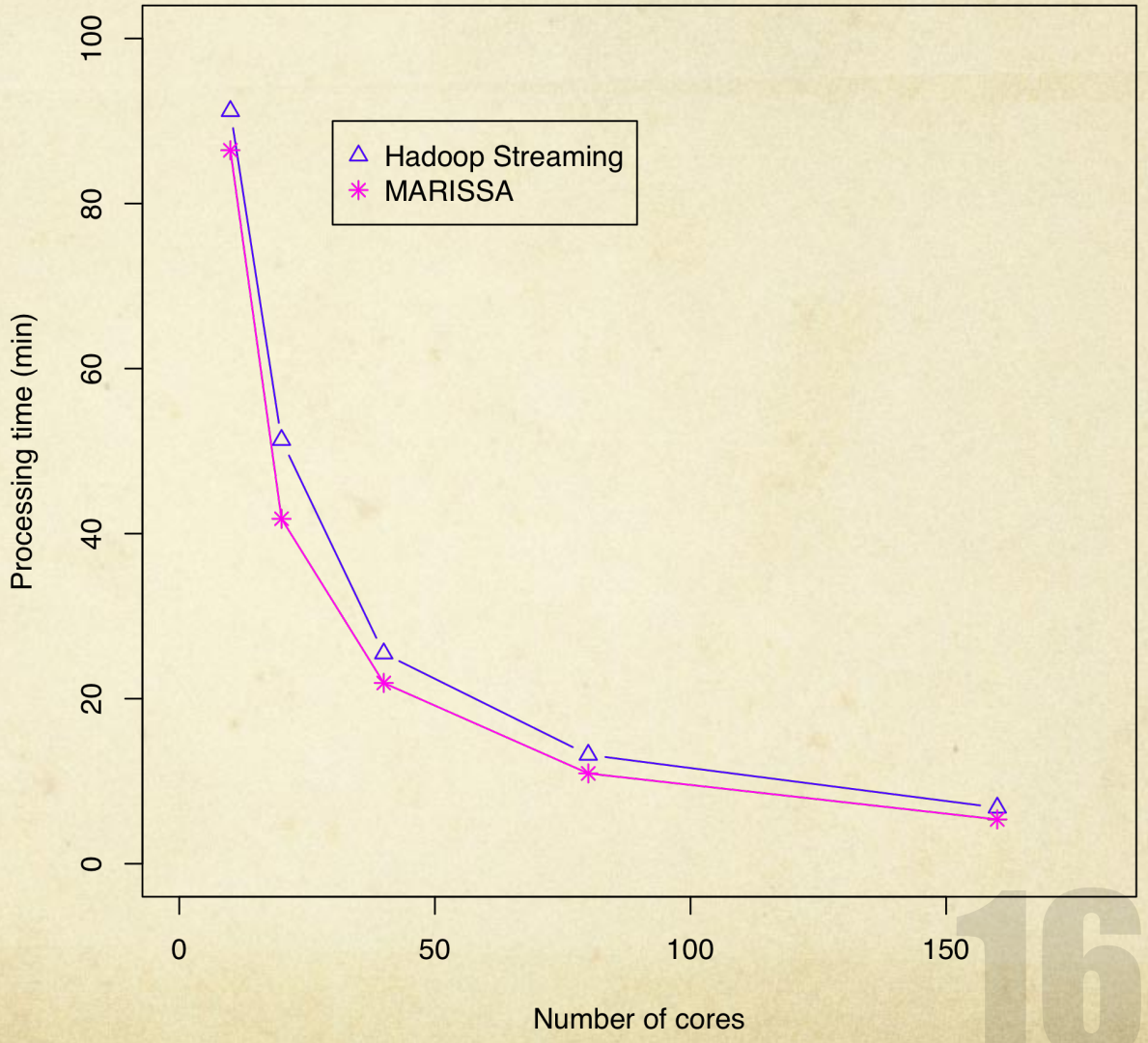
Hadoop Streaming vs MARISSA (Cont.)



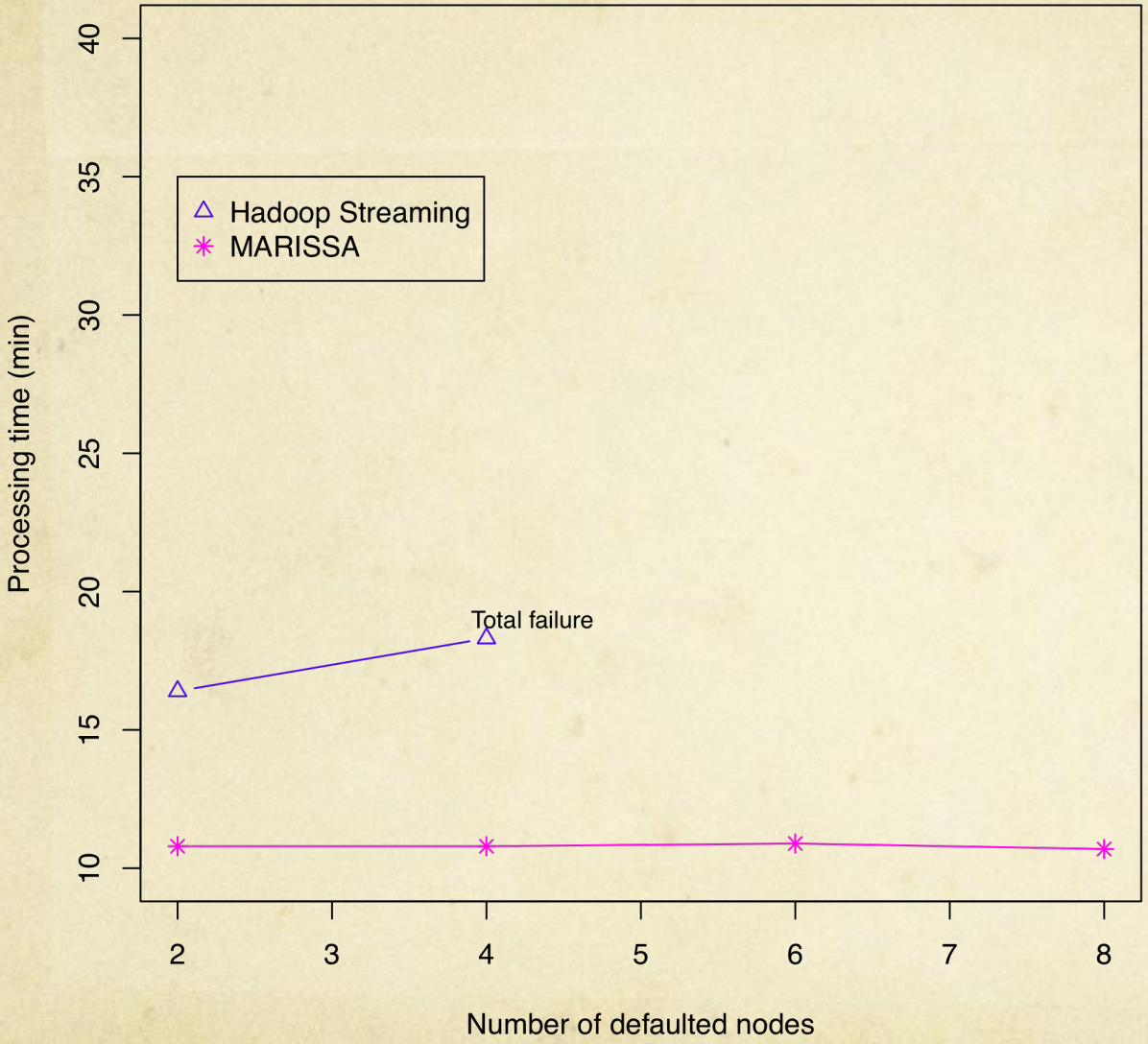
- 160-core Hadoop Streaming and MARISSA clusters
- Performing gene sequencing and similarity comparisons using BLAST
- Each framework processes from 10,240 to 327,680 genetic sequences, using 320 cores.

Hadoop Streaming vs MARISSA (Cont.)

- MARISSA and Hadoop clusters running BLAST
- Performing similarity comparisons on 40,960 genetic sequences
- The number of cores used here varies from 10 to 160 cores.

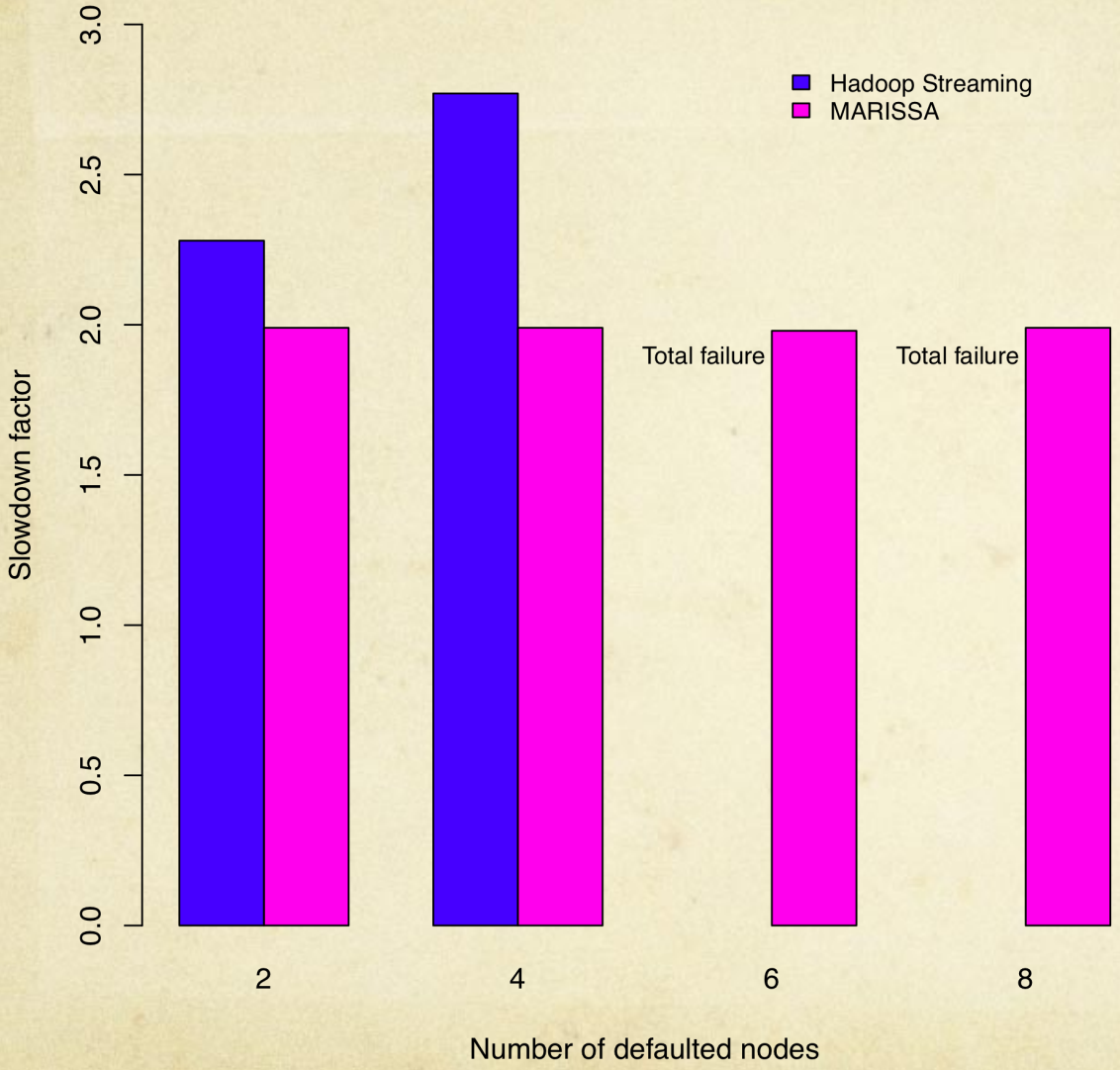


Hadoop Streaming vs MARISSA (Cont.)



- Hadoop Streaming and MARISSA faced with defaulting or dying nodes while running BLAST
- Processing 10240 genetic sequences
- Both clusters start with 20 nodes, and progressively, in separate runs lose 2, 4, and 6 nodes.
- The Hadoop experiences a total failure beyond a 4 node loss
- This occurs as worker nodes also hold vital data for processing.

Hadoop Streaming vs MARISSA (Cont.)



- Hadoop streaming and MARISSA slowdown given node losses while running BLAST.
- Both frameworks start with 20 nodes and process 10240 genetic sequences.
- Where comparable, Hadoop performs up to three times as slow, while MARISSA only slows down by a factor of 2.

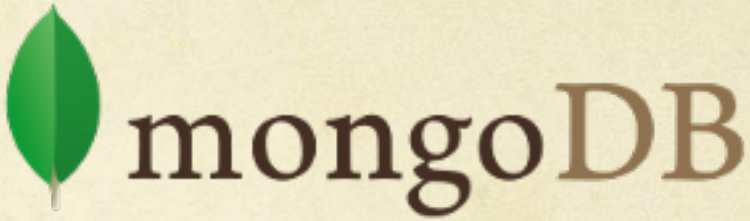
Conclusions

MARISSA allows for:

- The ability to work with existing file systems
- The ability to work with no STDIN/STDOUT applications
- The ability to run different executables on each node of the cluster
- The ability to run a same executables with different arguments
- The ability to run different input datasets on different nodes.
- The ability for all or a subset of the nodes to run duplicates of the same task with same input without need of data replication

Future Work


hadoop


mongoDB


ØMQ


RabbitMQ
Open Source Enterprise Messaging


HYPERTABLE


Cassandra


HBASE

Thanks.

Questions?

Contact:

Madhu Govindaraju

mgovinda@binghamton.edu

Binghamton University, State University of New York (SUNY)

NY 13902