

# Software for cluster computing

Steve Simon, P.Mean Consulting

# Abstract

This presentation will cover some of the software systems for cluster computing with an emphasis on R libraries that can work with these systems.

# Common features of software systems for cluster computing

All of software systems described in this presentation are

- distributed under open source licenses,
- have lots of free documentation, and
- run on any reasonable Unix based system.

# Message Passing Interface (MPI)

[Tutorials](#) | [Exercises](#) | [Abstracts](#) | [LC Workshops](#) | [Comments](#) | [Privacy & Legal Notice](#)

## Message Passing Interface (MPI)

*Author: Blaise Barney, Lawrence Livermore National Laboratory*

### Table of Contents

1. [Abstract](#)
2. [What is MPI?](#)
3. [LLNL MPI Implementations and Compilers](#)
4. [Getting Started](#)
5. [Environment Management Routines](#)
6. [Exercise 1](#)
7. [Point to Point Communication Routines](#)
  1. [General Concepts](#)
  2. [MPI Message Passing Routine Arguments](#)
  3. [Blocking Message Passing Routines](#)
  4. [Non-blocking Message Passing Routines](#)
8. [Exercise 2](#)
9. [Collective Communication Routines](#)
10. [Derived Data Types](#)
11. [Group and Communicator Management Routines](#)
12. [Virtual Topologies](#)
13. [A Brief Word on MPI-2 and MPI-3](#)
14. [Exercise 3](#)
15. [References and More Information](#)
16. [Appendix A: MPI-1 Routine Index](#)

Screenshot from Blaise Barney's MPI tutorial

# MPI is a routine library

## MPI Routines used for communicating between nodes

The Message Passing Interface is a standard used for message passing. Typically it is used in conjunction with a C or C++ program to farm out computation to the nodes of a cluster. The implementation of MPI used in this project was the open-source MPICH library. Two types of MPI operations were used in this project, collective and non-collective operations. Only two non-collective operations were used. *MPI\_Send* is used to send data from one node to another, *MPI\_Recv* is used to receive data from a particular node. Both these operations are blocking, meaning that the node which calls the operation pauses until the operation is complete.

```
MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag,
        MPIComm comm)
MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag,
        MPIComm comm, MPIStatus *status)
```

The other operations used are all collective. The *MPI\_Bcast* operation broadcasts a message from the root node to all other nodes/processes in the specified group. This is used to broadcast the dimension of the matrix to all nodes, and also to broadcast an "exit" matrix to each node.

```
MPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root,
        MPIComm comm)
```

When the group of nodes that are to work on the matrix-vector multiplication has been set up, the root node must give out a portion of the matrix to each node. This can be achieved with *MPI\_Send*, but it is much more efficient to use the *MPI\_Scatter* operation. This operation farms out pieces of an array to different nodes. Thus, the decomposition of the matrix can be achieved in just one command!

```
MPI_Scatter(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf,
            int recvcnt, MPI_Datatype recvtype, int root, MPIComm comm)
```

There is also a function called *MPI\_Gather* that implements the opposite function of *MPI\_Scatter*. When called on the root node, it gathers in data of a fixed size from all the nodes in the specified group, into an array. This is used to gather in the newly calculated qubit vector from the nodes, when the calculation is finished.

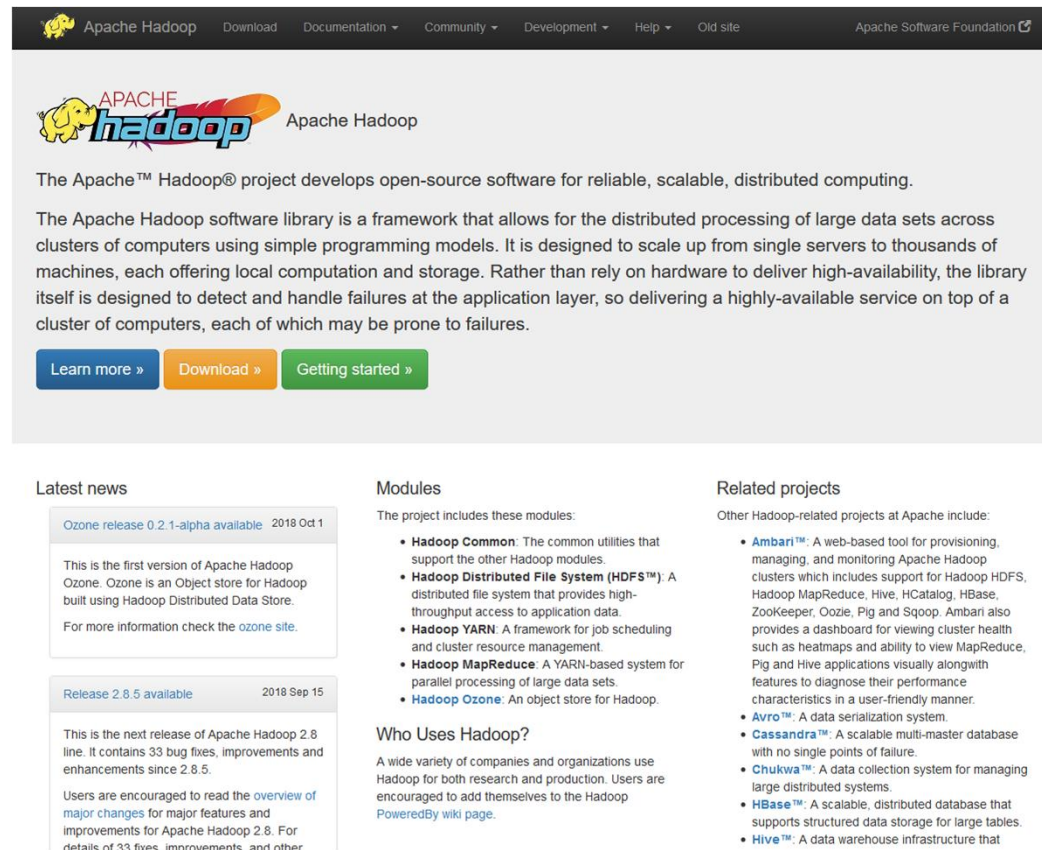
```
MPI_Gather(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf,
            int recvcnt, MPI_Datatype recvtype, int root, MPIComm comm)
```

None of the collective operations detailed above are blocking, even though they must operate at the same time on each node. To synchronize all the nodes, the *MPI\_Barrier* operation is called after a collective function. This ensures that all nodes in the group are operating in the correct place.

```
MPI_Barrier(MPIComm comm)
```

Colm Ó hÉigeartaigh's website listing of MPI routines

# Hadoop



The screenshot shows the Apache Hadoop website. At the top is a dark navigation bar with links for Apache Hadoop, Download, Documentation, Community, Development, Help, and Old site, along with the Apache Software Foundation logo. Below the navigation bar is the Apache Hadoop logo and the text "Apache Hadoop". The main content area features a paragraph about the project's goal of developing open-source software for reliable, scalable, distributed computing. It describes the software library as a framework for distributed processing of large data sets across clusters of computers. Below this text are three buttons: "Learn more", "Download", and "Getting started". The bottom section is divided into three columns: "Latest news" with two news items about Ozone and Hadoop 2.8.5 releases; "Modules" listing Hadoop Common, HDFS, YARN, MapReduce, and Ozone; and "Related projects" listing Ambari, Avro, Cassandra, Chukwa, HBase, and Hive.

Apache Hadoop

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

[Learn more »](#) [Download »](#) [Getting started »](#)

### Latest news

**Ozone release 0.2.1-alpha available** 2018 Oct 1

This is the first version of Apache Hadoop Ozone. Ozone is an Object store for Hadoop built using Hadoop Distributed Data Store. For more information check the [ozone site](#).

**Release 2.8.5 available** 2018 Sep 15

This is the next release of Apache Hadoop 2.8 line. It contains 33 bug fixes, improvements and enhancements since 2.8.5. Users are encouraged to read the [overview of major changes](#) for major features and improvements for Apache Hadoop 2.8. For details of 33 fixes, improvements, and other

### Modules

The project includes these modules:

- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.
- **Hadoop Ozone**: An object store for Hadoop.

### Who Uses Hadoop?

A wide variety of companies and organizations use Hadoop for both research and production. Users are encouraged to add themselves to the Hadoop [PoweredBy wiki page](#).

### Related projects

Other Hadoop-related projects at Apache include:

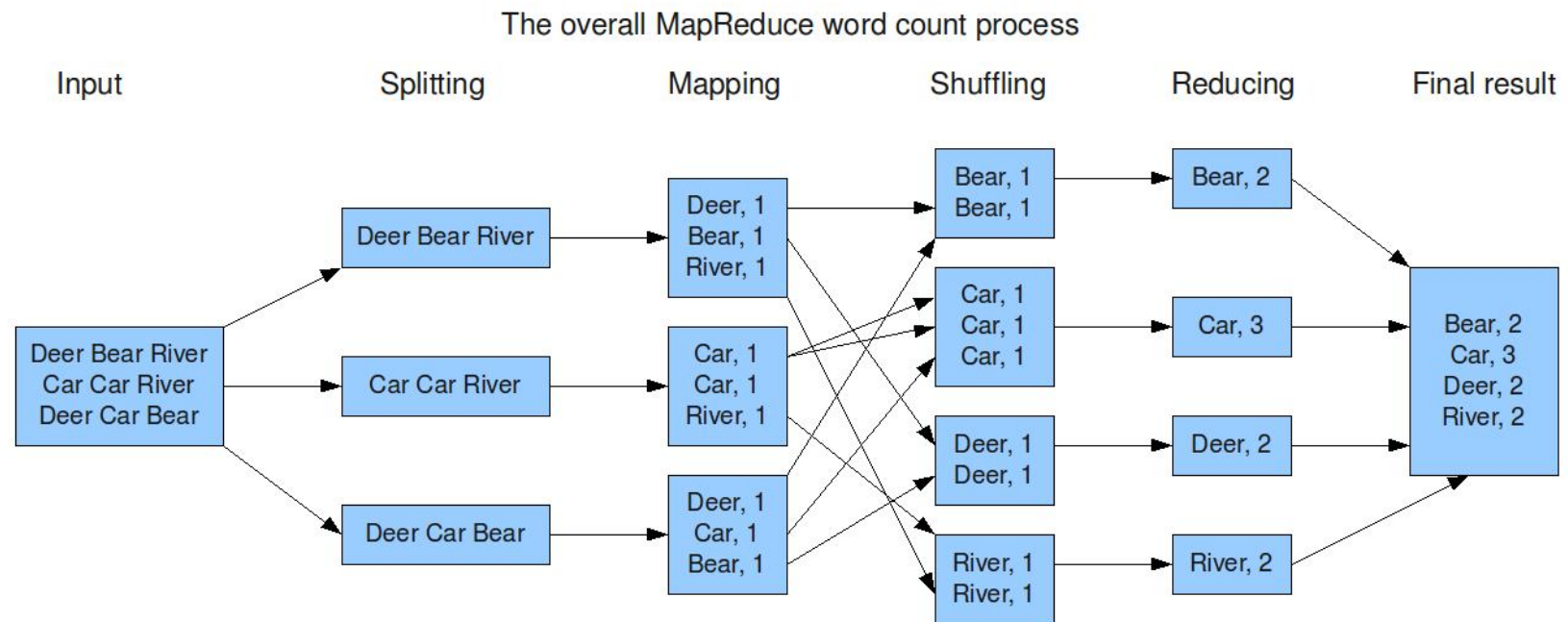
- **Ambari™**: A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop. Ambari also provides a dashboard for viewing cluster health such as heatmaps and ability to view MapReduce, Pig and Hive applications visually alongwith features to diagnose their performance characteristics in a user-friendly manner.
- **Avro™**: A data serialization system.
- **Cassandra™**: A scalable multi-master database with no single points of failure.
- **Chukwa™**: A data collection system for managing large distributed systems.
- **HBase™**: A scalable, distributed database that supports structured data storage for large tables.
- **Hive™**: A data warehouse infrastructure that

Screenshot from Apache Hadoop website

# Components of Hadoop

- MapReduce
- Hadoop Distributed File System (HDFS)
- Hive
- Pig

# MapReduce



Mapreduce applied to a simple word count example



# Hadoop Distributed File System (HDFS)

HDFS Architecture

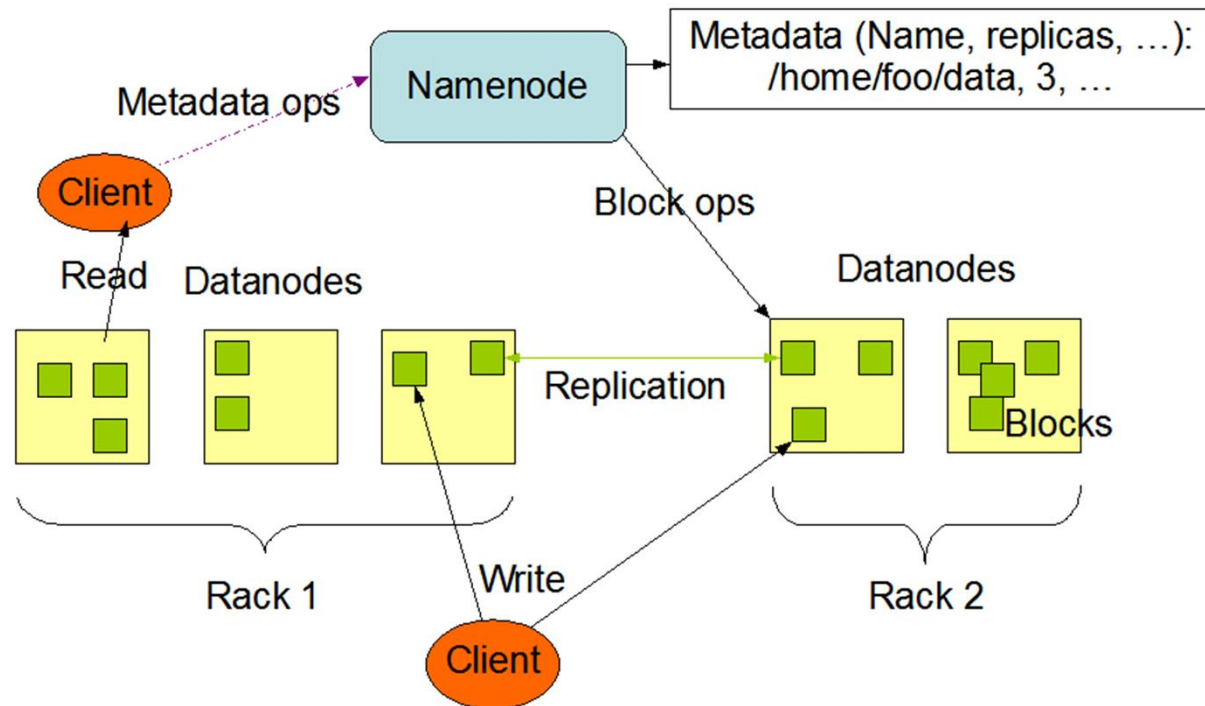


Illustration of HDFS architecture

Pig

# Hive

# Spark

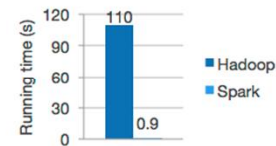
[Download](#)[Libraries ▾](#)[Documentation ▾](#)[Examples](#)[Community ▾](#)[Developers ▾](#)[Apache Software Foundation ▾](#)

Apache Spark™ is a unified analytics engine for large-scale data processing.

## Speed

Run workloads 100x faster.

Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.



Logistic regression in Hadoop and Spark

## Ease of Use

Write applications quickly in Java, Scala, Python, R, and SQL.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala, Python, R, and SQL shells.

```
df = spark.read.json("logs.json")
df.where("age > 21")
  .select("name.first").show()
```

Spark's Python DataFrame API  
Read JSON files with automatic schema inference

### Latest News

[Spark 2.4.0 released](#) (Nov 02, 2018)

[Spark 2.3.2 released](#) (Sep 24, 2018)

[Spark+AI Summit](#) (October 2-4th, 2018, London) agenda posted (Jul 24, 2018)

[Spark 2.2.2 released](#) (Jul 02, 2018)

[Archive](#)

[Download Spark](#)

### Built-in Libraries:

[SQL and DataFrames](#)[Spark Streaming](#)[MLlib \(machine learning\)](#)[GraphX \(graph\)](#)[Third-Party Projects](#)

Screenshot of Apache Spark main web page

# Conclusion

This talk has covered

- Message Passing Interface
- Hadoop
  - MapReduce
  - HDFS
  - Pig
  - Hive
- Spark