Back to Hadoop

What is Hadoop?

- Hadoop is an ecosystem of tools for processing “Big Data”.
- Hadoop is an open source project.

A family of tools

MapReduce	Distributed computation framework (data processing model and execution environment)
HDFS	Distributed file system
HBase	Distributed, column-oriented database
Hive	Distributed data warehouse
Pig	Higher-level data flow language and parallel execution framework
ZooKeeper	Distributed coordination service
Avro	Data serialization system (RPC and persistent data storage)
Sqoop	Tool for bulk data transfer between structured data stores (e.g., RDBMS) and HDFS
Oozie	Complex job workflow service
Chukwa	System for collecting management data
Mahout	Machine learning and data mining library
BigTop	Packaging and testing

DBMS/SQL

```
SELECT * FROM Customers
WHERE Country='Mexico';

SELECT column1, column2....columnN
FROM   table_name
WHERE  CONDITION
ORDER BY column_name {ASC|DESC};
```

Hypothetical Relational Database Model

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7733903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4859223	O'Reilly Press	77 Boston Ave, Cambridge
03-3920886	City Lights Books	99 Market, San Francisco

AuthorID	AuthorName	AuthorBDay
345-28-2938	Haile Selassie	14-Aug-92
392-48-9965	Joe Blow	14-Mar-15
454-22-4012	Sally Hemmings	12-Sept-70
663-59-1254	Hannah Arendt	12-Mar-06

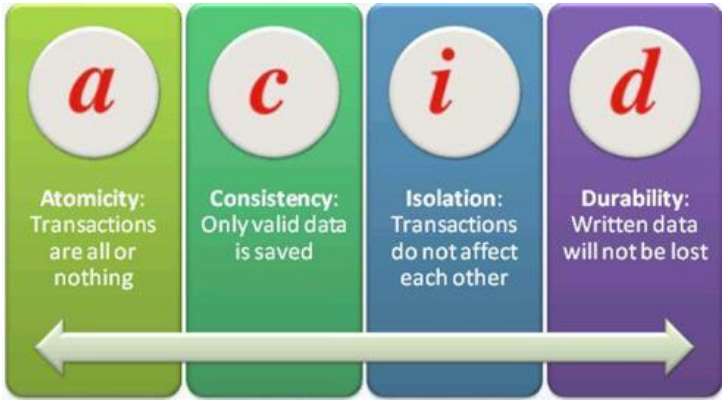
ISBN	AuthorID	PubID	Date	Title
1-34532-482-1	345-28-2938	03-4472822	1990	Cold Fusion for Dummies
1-38482-995-1	392-48-9965	04-7733903	1985	Macrame and Straw Tying
2-35921-499-4	454-22-4012	03-4859223	1952	Fluid Dynamics of Aquaducts
1-38278-293-4	663-59-1254	03-3920886	1967	Beads, Baskets & Revolution

artist_id	artist_name
1	Bono
2	Cher
3	Nuno Bettencourt

Link Broken

artist_id	album_id	album_name
3	1	Schizophonic
4	2	Eat the rich
3	3	Crave (single)

Referential Integrity



MapReduce vs. Traditional RDBMS

	MapReduce	Traditional RDBMS
Data size	Petabytes	Gigabytes
Access	Batch	Interactive and batch
Updates	Write once, read many times	Read and write many times
Structure	Dynamic schema	Static schema
Integrity	Low	High (normalized data)
Scaling	Linear	Non-linear (general SQL)

Message Passing Interface (MPI)

- Communication protocol for programming parallel computers.
  - Point-to-point and collective communication
  - *”a message-passing application programmer interface, together with protocol and semantic specifications for how its features must behave in any implementation.”*
- Goals: high performance, scalability, and portability.
- Dominant model used in high-performance computing

- Virtual topologies
  - Predefined Naming schemas
  - Graph, Cartesian (e.g., refer to other processes by coordinates)

0 (0,0)	1 (0,1)	2 (0,2)	3 (0,3)
4 (1,0)	5 (1,1)	6 (1,2)	7 (1,3)
8 (2,0)	9 (2,1)	10 (2,2)	11 (2,3)
12 (3,0)	13 (3,1)	14 (3,2)	15 (3,3)

Message Passing Interface (MPI)

- User writes a single program that runs on all computers

```
if ( I am processor A ) then
      add a bunch of numbers
else if ( I am processor B ) then
      multiply a matrix times a vector
end
```

- Data on a computer is separate from data on others
  - Explicit data transfer
  - Sync points

```
if ( I am processor A ) then
      call MPI_Send ( X )
    else if ( I am processor B ) then
      call MPI_Recv ( X )
    end
```

```
# include <mpi.h>
# include <stdlib.h>
# include <stdio.h>
# include <time.h>

int main ( int argc, char *argv[] );
void timestamp ( );

int main ( int argc, char *argv[] ){
  int count;
  float data[100];
  int dest;
  int i;
  int ierr;
  int num_procs;
  int rank;
  int source;
  MPI_Status status;
  int tag;
  float value[200];
/*
  Initialize MPI.
*/
  ierr = MPI_Init ( &argc, &argv );
/*
  Determine this process's rank.
*/
  ierr = MPI_Comm_rank ( MPI_COMM_WORLD,
/*
  Determine the number of available processes.
*/
  ierr = MPI_Comm_size ( MPI_COMM_WORLD, &num_procs );
/*
  Have Process 0 say hello.
*/
```

```
  if ( rank == 0 ){
    timestamp ( );
    printf ( "\n" );
    printf ( "BONES:\n" );
    printf ( "  C version\n" );
    printf ( "  An MPI example program.\n" );
    printf ( "  The number of processes available is %d\n", num_procs );
  }
/*
  Process 0 expects up to 200 real values, from any source.
*/
  if ( rank == 0 ) {
    source = 1;
    tag = 55;

    ierr = MPI_Recv ( value, 200, MPI_FLOAT, MPI_ANY_SOU
      MPI_COMM_WORLD, &status );
    ierr = MPI_Get_count ( &status, MPI_FLOAT, &count );
    printf ( "P:%d Got %d elements.\n", rank, count );
    printf ( "P:%d value[5] = %f\n", rank, value[5] );
  }
/*
  Process 1 sends 100 real values to process 0.
*/
  else if ( rank == 1 ){
    printf ( "\n" );
    printf ( "P:%d - setting up data to send to process 0.\n",

    for ( i = 0; i < 100; i++ ) {
      data[i] = i;
    }

    dest = 0;
    tag = 55;
    ierr = MPI_Send ( data, 100, MPI_FLOAT, dest, tag, MPI_COMM_WORLD );
  }
```

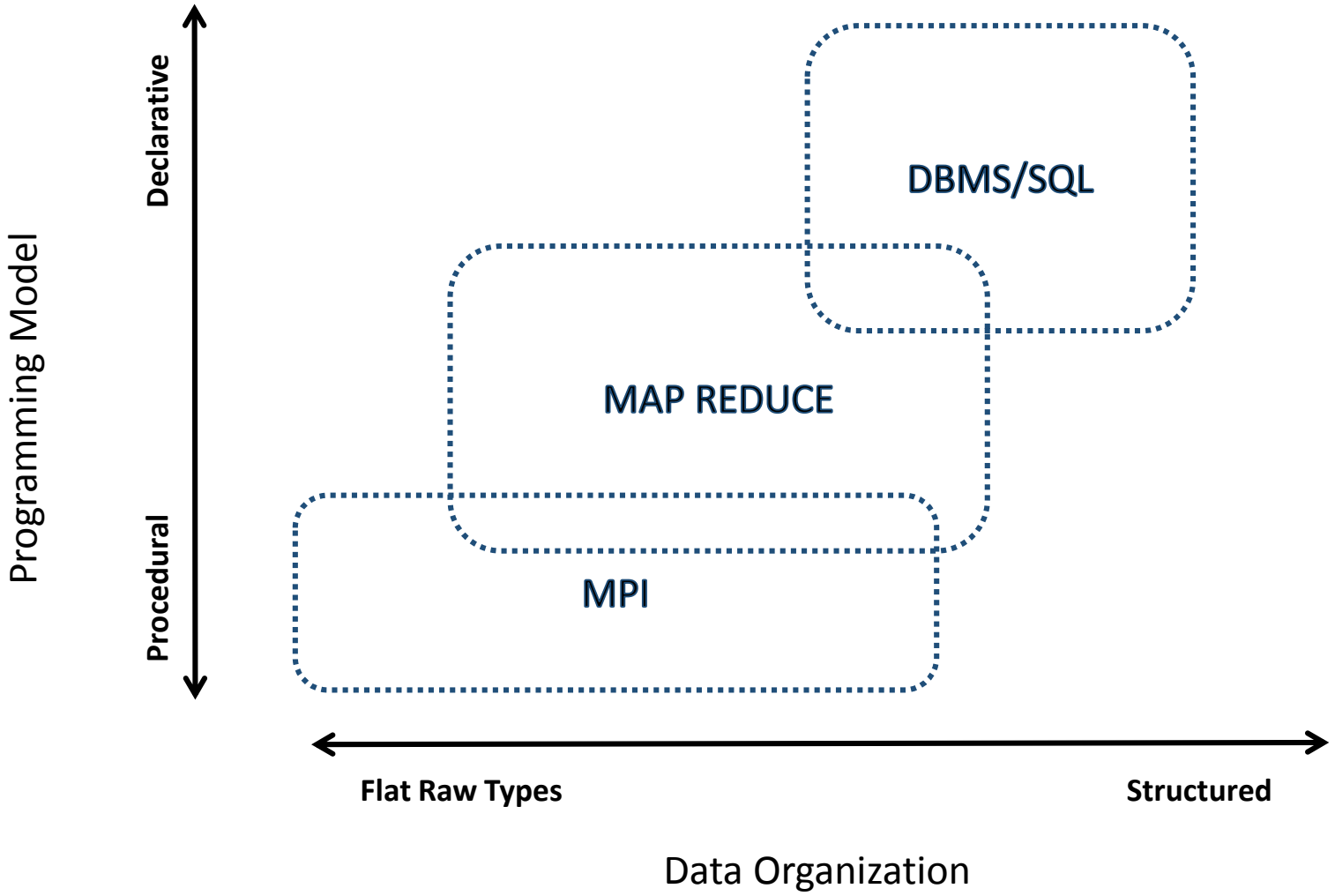
```
/*
  Any other process is idle.
*/
  else {
    printf ( "\n" );
    printf ( "P:%d - MPI has no work for me!\n", rank );
  }
/*
  Terminate MPI.
*/
  ierr = MPI_Finalize ( );
/*
  Terminate.
*/
  if ( rank == 0 ) {
    printf ( "\n" );
    printf ( "BONES:\n" );
    printf ( "  Normal end of execution.\n" );
    printf ( "\n" );
    timestamp ( );
  }
  return 0;
}

  ...
```

A demonstration of the use of MPI by a C program.
This program should be run on at least two processes.
Any processes beyond the first two will not be given any work.

http://people.sc.fsu.edu/~jburkardt/c_src/mpi/bones_mpi.c

A Summary

	MPI	MapReduce	DBMS/SQL
What they are	A general parrellel programming paradigm	A programming paradigm and its associated execution system	A system to store, manipulate and serve data.
Programming Model	Messages passing between nodes	Restricted to Map/Reduce operations	Declarative on data query/retrieving; Stored procedures
Data organization	No assumption	"files" can be sharded	Organized datastructures
Data to be manipulated	Any	k,v pairs: string	Tables with rich types
Execution model	Nodes are independent	Map/Shuffle/Reduce Checkpointing/Backup Physical data locality	Transaction Query/operation optimization Materialized view
Usability	Steep learning curve*; difficult to debug	Simple concept Could be hard to optimize	Declarative interface; Could be hard to debug in runtime
Key selling point	Flexible to accommodate various applications	Plow through large amount of data with commodity hardware	Interactive querying the data; Maintain a consistent view across clients


Programming Model
Declarative
Procedural
DBMS/SQL
MAP REDUCE
MPI
Flat Raw Types
Structured
Data Organization

Questions?

Sources & References

MPI Examples

- http://people.sc.fsu.edu/~jburkardt/c_src/mpi/mpi.html

Tool echosystem

- www.dbs.ifi.lmu.de/.../Big**Data**...16/Chapter-3_DFS_MapReduce_Hadoop_part2.pdf