Large Scale Analtics with the SimSQL/ Pliny Compute Platform

Chris Jermaine
Rice University

Current/Recent team: Jia Zou, Kia Teymourian, Matt Barnett, Carlos Monroy, Zhuhua Cai, Jacob Gao, Michael Gubanov, Shangyu Luo, Luis Perez
Also, Peter J. Haas at IBM Almaden

The Last Few Years…

- Have seen huge interest in large-scale data-processing systems
- OptiML, GraphLab, SystemML, MLBase, HBase, MongoDB, BigTable, Pig, Impala, ScalOps, Pregel, Giraph, Hadoop, TupleWare, Hama, Spark, Flink, Ricardo, Nyad, DradLinq, and **many** others...
- Many have had significant industrial impact...
- Why now?

Until the Mid-2000's

- If you had a large-scale data-processing problem, you:
    1. Rolled your own
    2. Bought an SQL-based database system
    3. Used a niche (often industry-specific) solution

Until the Mid-2000's

- If you had a large-scale data-processing problem, you:
  1. Rolled your own 
  2. Bought an SQL-based database system
  3. Used a niche (often industry-specific) solution 

Available to very few people

Until the Mid-2000's

- If you had a large-scale data-processing problem, you:
    1. Rolled your own
    2. **Bought an SQL-based database system**
    3. Used a niche (often industry-specific) solution

SQL Databases

- What were the complaints?
  - — Poor performance

SQL Databases

- What were the complaints?
  - — Poor performance  **This is misleading...**

SQL Databases

- What were the complaints?
  - — Poor price/performance (Teradata costs **a lot**)

SQL Databases

- What were the complaints?
  - — Poor price/performance (Teradata costs **a lot**)
  - — No open-source solution with good scale out

SQL Databases

- What were the complaints?
  - — Poor price/performance (Teradata costs **a lot**)
  - — No open-source solution with good scale out
  - **— Frustration with cost to load data into relations**

SQL Databases

- What were the complaints?
  - — Poor price/performance (Teradata costs **a lot**)
  - — No open-source solution with good scale out
  - **— Frustration with cost to load data into relations**
  - — SQL has never played nicely with other tools (software packages, other PLs)

SQL Databases

- What were the complaints?
  - — Poor price/performance (Teradata costs **a lot**)
  - — No open-source solution with good scale out
  - — **Frustration with cost to load data into relations**
  - — SQL has never played nicely with other tools (software packages, other PLs)

A Lot Of Pain in the Analytics Space

Then, Suddenly We Had Hadoop

- Our salvation! Right??

Then, Suddenly We Had Hadoop

```
import java.util.*;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;

public class WordCount {

  public static int main(String[] args) throws Exception {

    // if we got the wrong number of args, then exit
    if (args.length != 4 || !args[0].equals ("-r")) {
      System.out.println("usage: WordCount -r <num reducers> <input> <output>");
      return -1;
    }

    // Get the default configuration object
    Configuration conf = new Configuration ();

    // now create the MapReduce job
    Job job = new Job (conf);
    job.setJobName ("WordCount");

    // we'll output text/int pairs (since we have words as keys and counts as values)
    job.setMapOutputKeyClass (Text.class);
    job.setMapOutputValueClass (IntWritable.class);

    // again we'll output text/int pairs (since we have words as keys and counts as values)
    job.setOutputKeyClass (Text.class);
    job.setOutputValueClass (IntWritable.class);

    // tell Hadoop the mapper and the reducer to use
    job.setMapperClass (WordCountMapper.class);
    job.setCombinerClass (WordCountReducer.class);
    job.setReducerClass (WordCountReducer.class);

    // we'll be reading in a text file, so we can use Hadoop's built-in TextInputFormat
    job.setInputFormatClass (TextInputFormat.class);

    // we can use Hadoop's built-in TextOutputFormat for writing out the output text file
    job.setOutputFormatClass (TextOutputFormat.class);

    // set the input and output paths
    TextInputFormat.setInputPaths (job, args[2]);
    TextOutputFormat.setOutputPath (job, new Path (args[3]));

    // set the number of reduce paths
    try {
      job.setNumReduceTasks (Integer.parseInt (args[1]));
    } catch (Exception e) {
      System.out.println("usage: WordCount -r <num reducers> <input> <output>");
      return -1;
    }

    // force the mappers to handle one megabyte of input data each
    TextInputFormat.setMinInputSplitSize (job, 1024 * 1024);
    TextInputFormat.setMaxInputSplitSize (job, 1024 * 1024);

    // this tells Hadoop to ship around the jar file containing "WordCount.class" to all of
the different
    // nodes so that they can run the job
    job.setJarByClass(WordCount.class);

    // submit the job and wait for it to complete!
    int exitCode = job.waitForCompletion (true) ? 0 : 1;
    return exitCode;
```

Here's the main program for Word Count...

Then, Suddenly We Had Hadoop

```
import java.util.*;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;

public class WordCount {

  public static int main(String[] args) throws Exception {

    // if we got the wrong number of args, then exit
    if (args.length != 4 || !args[0].equals ("-r")) {
      System.out.println("usage: WordCount -r <num reducers> <input> <output>");
      return -1;
    }

    // Get the default configuration object
    Configuration conf = new Configuration ();

    // now create the MapReduce job
    Job job = new Job (conf);
    job.setJobName ("WordCount");

    // we'll output text/int pairs (since we have words as keys and counts as values)
    job.setMapOutputKeyClass (Text.class);
    job.setMapOutputValueClass (IntWritable.class);

    // again we'll output text/int pairs (since we have words as keys and counts as values)
    job.setOutputKeyClass (Text.class);
    job.setOutputValueClass (IntWritable.class);

    // tell Hadoop the mapper and the reducer to use
    job.setMapperClass (WordCountMapper.class);
    job.setCombinerClass (WordCountReducer.class);
    job.setReducerClass (WordCountReducer.class);

    // we'll be reading in a text file, so we can use Hadoop's built-in TextInputFormat
    job.setInputFormatClass (TextInputFormat.class);

    // we can use Hadoop's built-in TextOutputFormat for writing out the output text file
    job.setOutputFormatClass (TextOutputFormat.class);

    // set the input and output paths
    TextInputFormat.setInputPaths (job, args[2]);
    TextOutputFormat.setOutputPath (job, new Path (args[3]));

    // set the number of reduce paths
    try {
      job.setNumReduceTasks (Integer.parseInt (args[1]));
    } catch (Exception e) {
      System.out.println("usage: WordCount -r <num reducers> <input> <output>");
      return -1;
    }

    // force the mappers to handle one megabyte of input data each
    TextInputFormat.setMinInputSplitSize (job, 1024 * 1024);
    TextInputFormat.setMaxInputSplitSize (job, 1024 * 1024);

    // this tells Hadoop to ship around the jar file containing "WordCount.class" to all of
the different
    // nodes so that they can run the job
    job.setJarByClass(WordCount.class);

    // submit the job and wait for it to complete!
    int exitCode = job.waitForCompletion (true) ? 0 : 1;
    return exitCode;
```

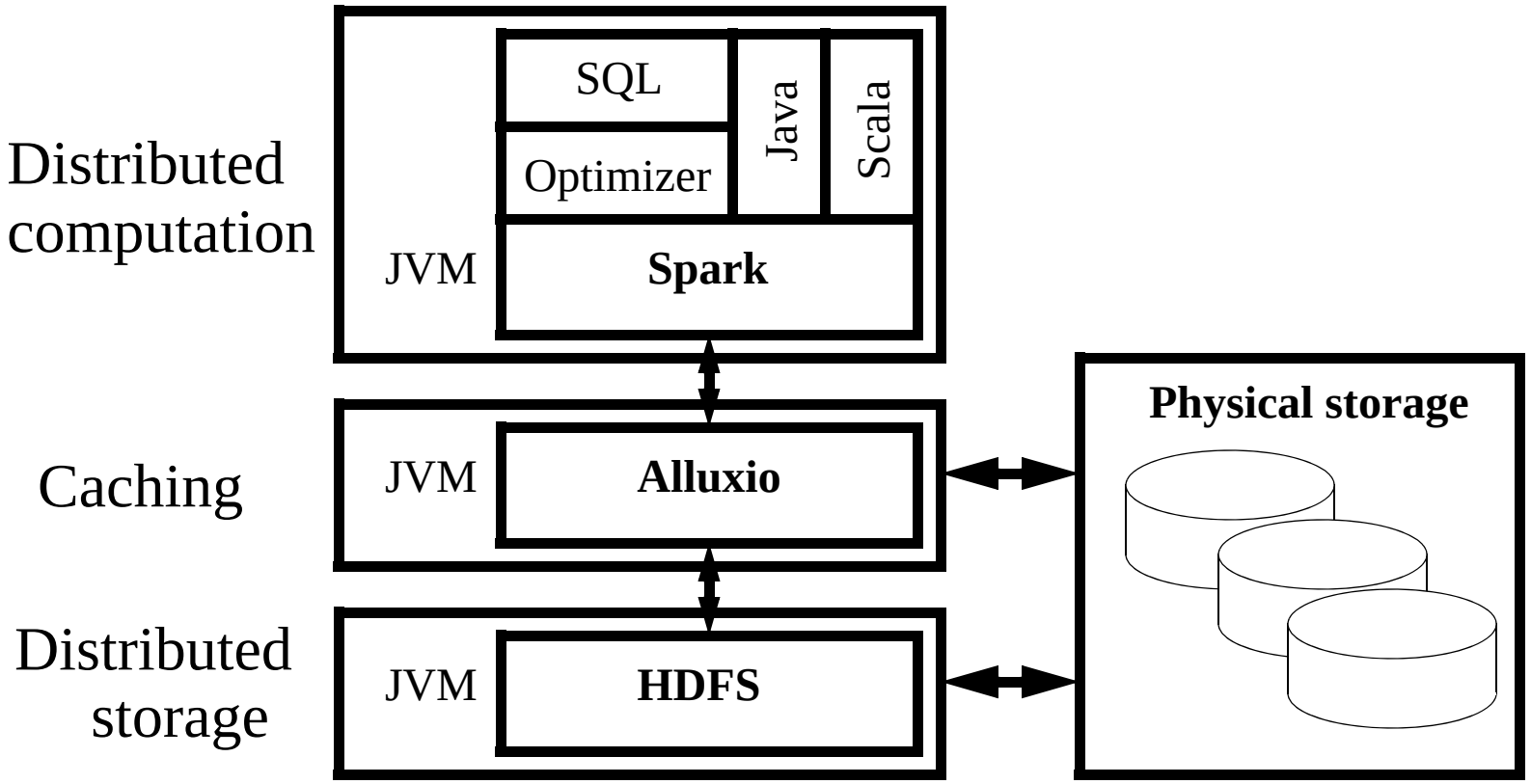
No pretty!

Plus, often not that fast...

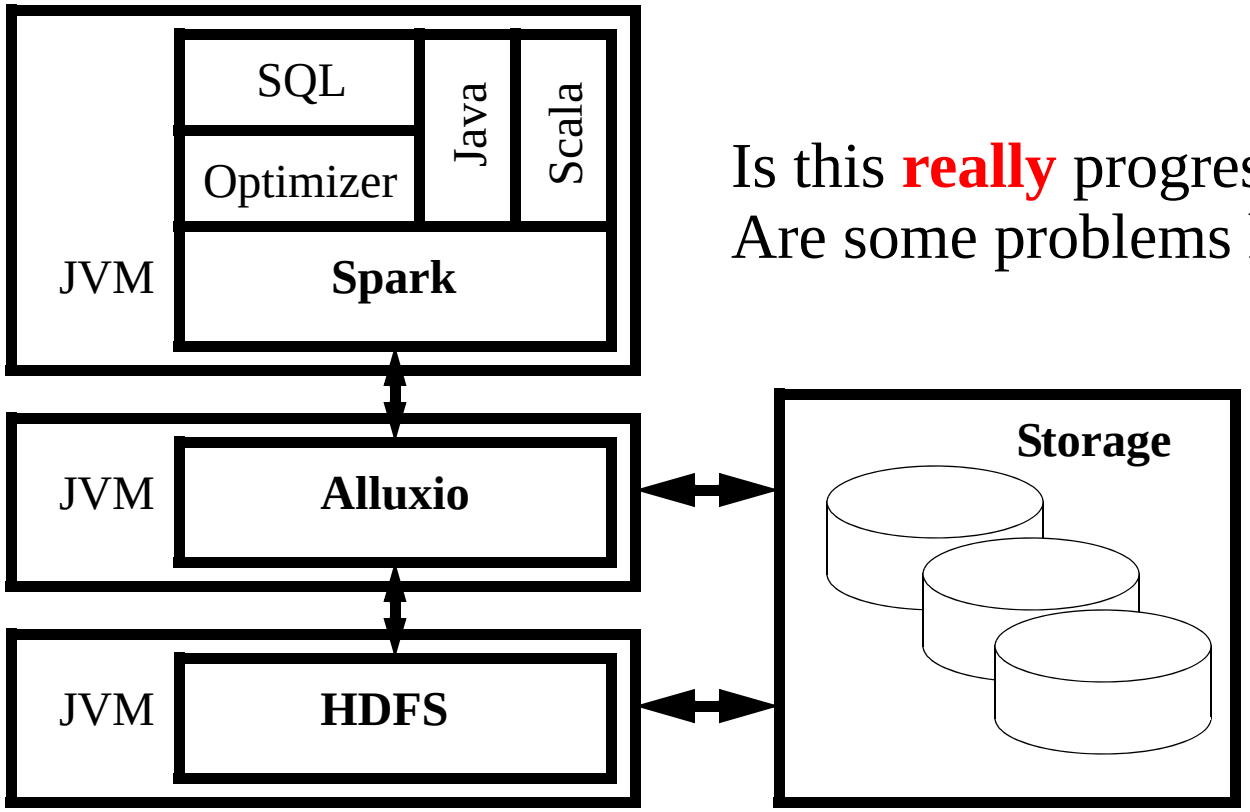
Next Generation “DataFlow” Platforms

- DryadLINQ, Spark, Flinq
  - — Better performance than Hadoop
  - — Human being strings together (simple) bulk ops to form a computation
  - — Much easier to program

The Emerging Analytics Ecosystem



The Emerging Analytics Ecosystem

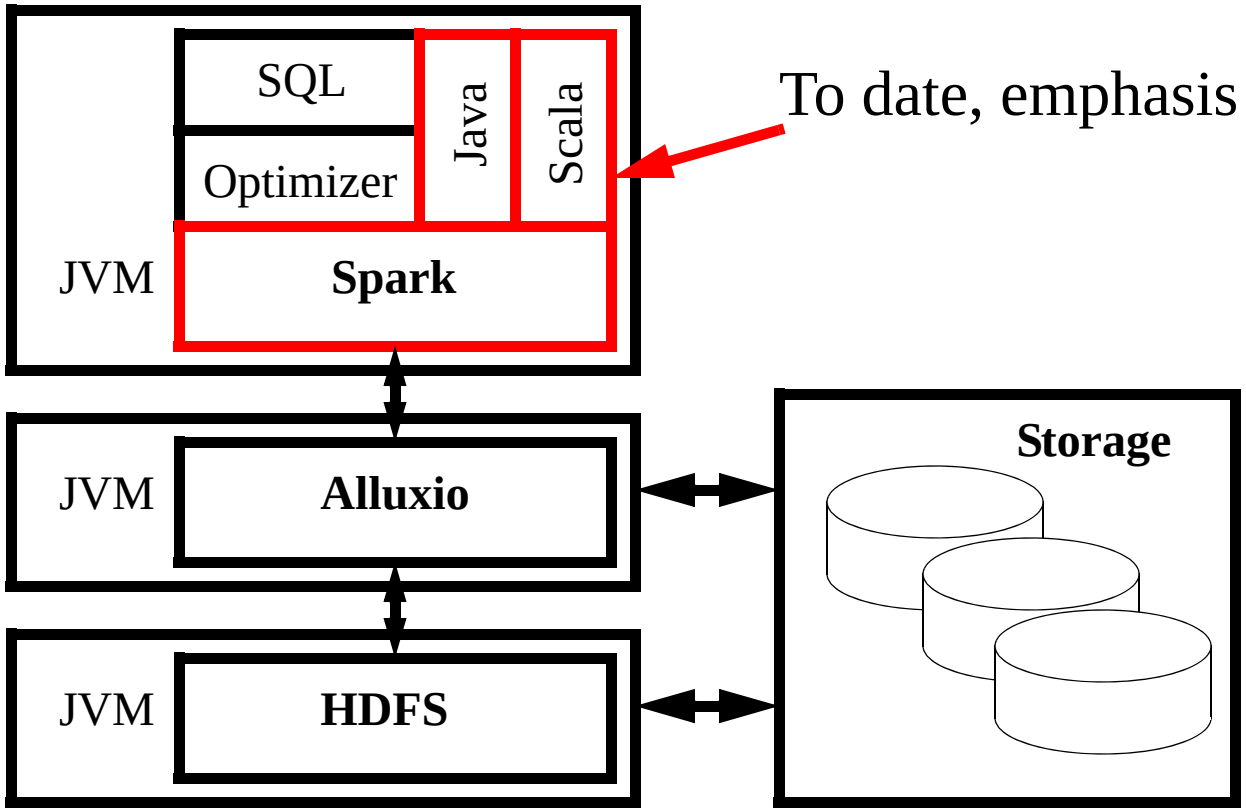


Is this **really** progress?
Are some problems here...

The Perils of Layering

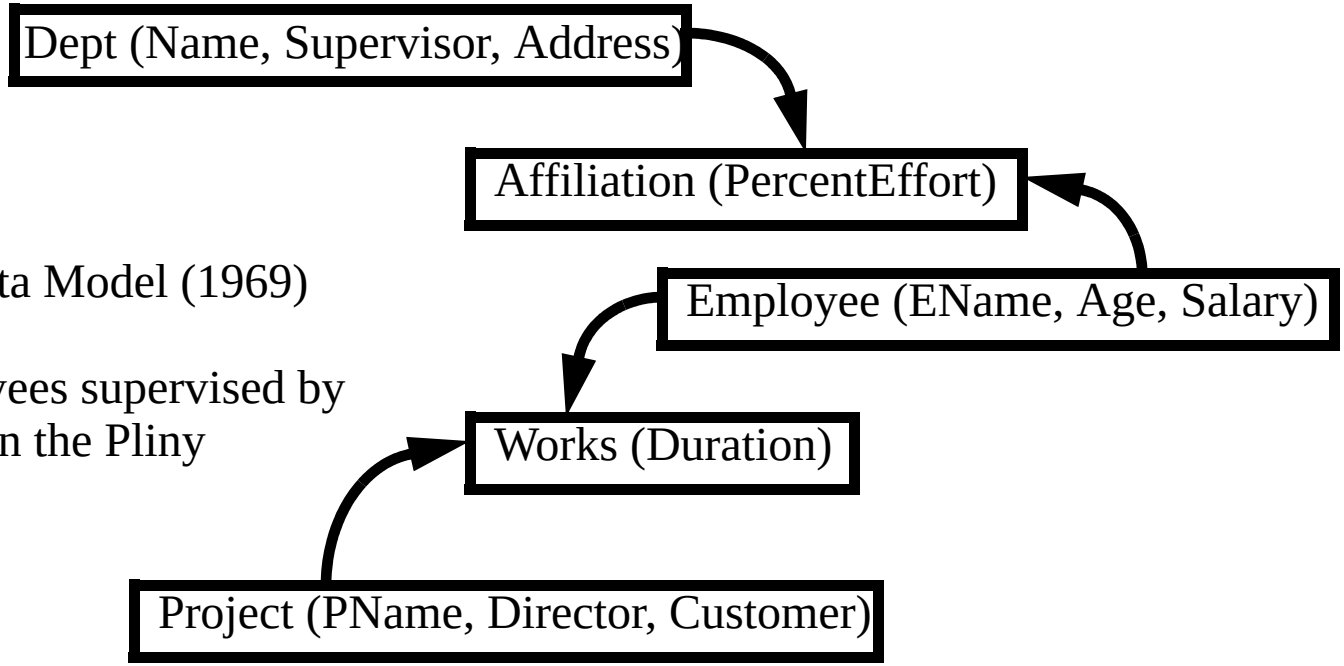
- How far does data travel? Consider a request served from disk
  1. Spark (in own JVM) asks Alluxio (in different JVM) for data
  2. Alluxio asks HDFS (in different JVM) for data
  3. (Worst case) HDFS goes out to network to find compute node with data
  4. HDFS serves Alluxio
  5. Alluxio serves Spark
- Our experience: up to 100X performance hit compared to a vertically integrated approach

Another Big Problem: Relying on Engineers to Be Smart



One Step Forward, Two Steps Back

- "Dataflow" platforms (such as Spark) are effectively RA engines
  - — Human being strings together (simple) bulk ops to form a computation
  - — But DB people have recognized imperative data access is bad for nearly 40 years

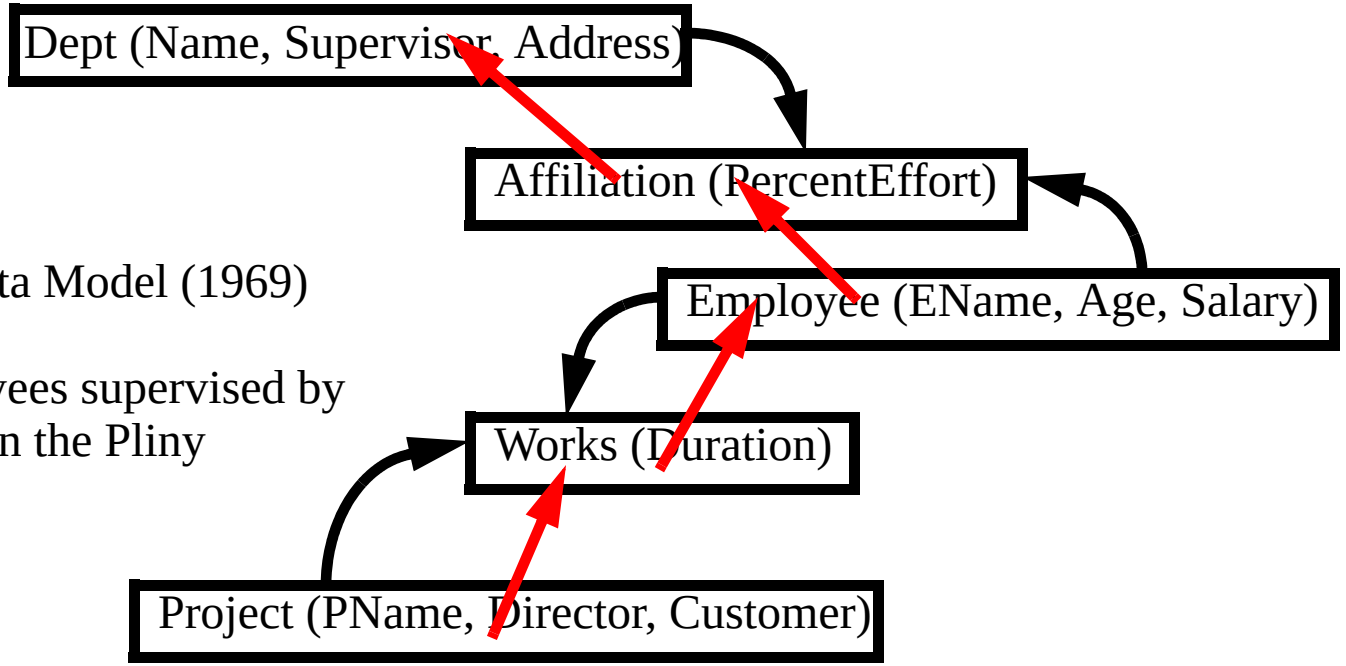


CODASYL Data Model (1969)

"Which employees supervised by Chris worked on the Pliny project?"

One Step Forward, Two Steps Back

- "Dataflow" platforms (such as Spark) are effectively RA engines
  - — Human being strings together (simple) bulk ops to form a computation
  - — But DB people have recognized imperative data access is bad for nearly 40 years



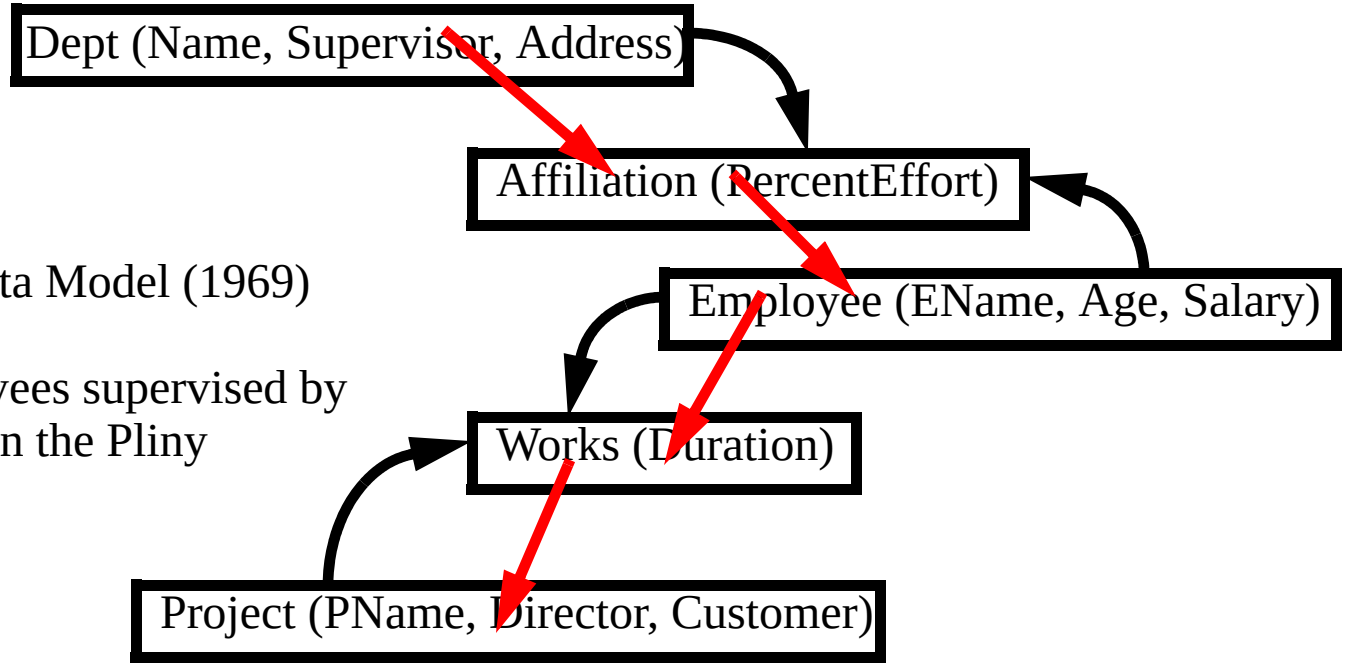
CODASYL Data Model (1969)

"Which employees supervised by Chris worked on the Pliny project?"

Good if few employees per project

One Step Forward, Two Steps Back

- "Dataflow" platforms (such as Spark) are effectively RA engines
  - — Human being strings together (simple) bulk ops to form a computation
  - — But DB people have recognized imperative data access is bad for nearly 40 years



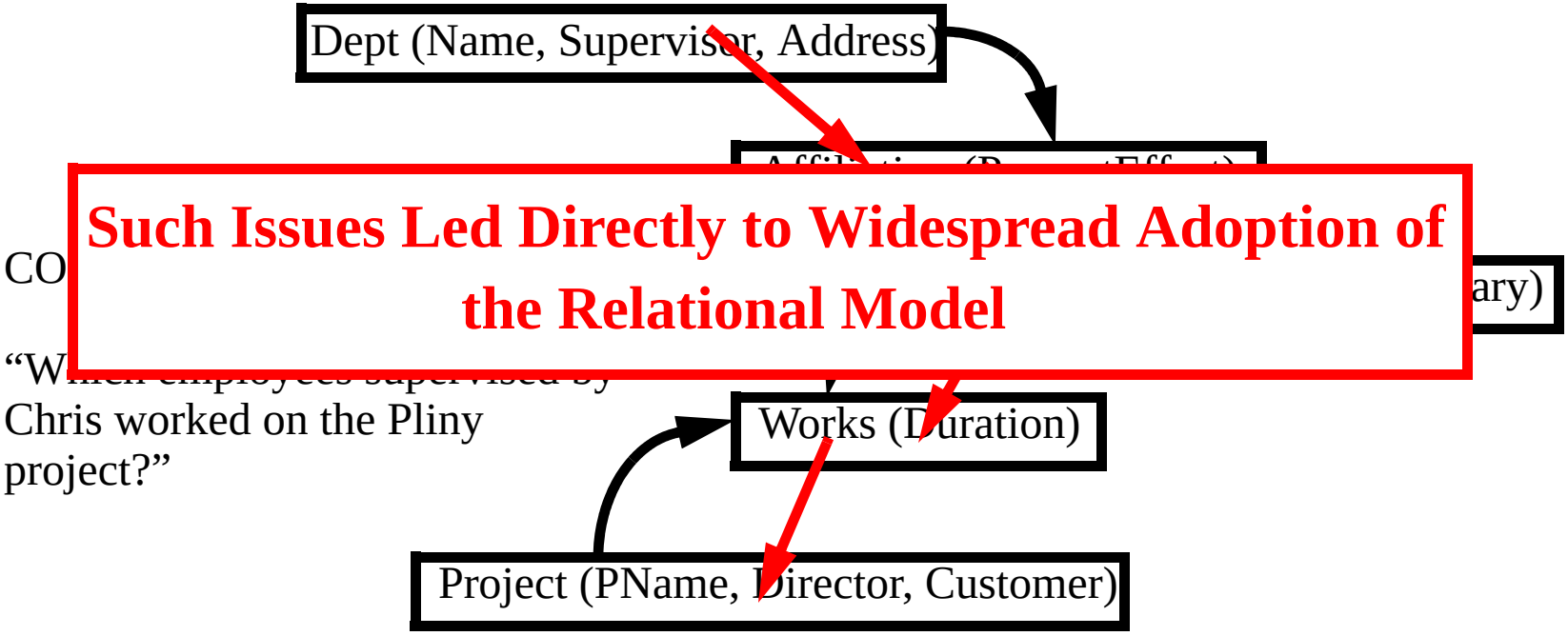
CODASYL Data Model (1969)

"Which employees supervised by Chris worked on the Pliny project?"

Good if few people supervised by Chris

One Step Forward, Two Steps Back

- "Dataflow" platforms (such as Spark) are effectively RA engines
  - — Human being strings together (simple) bulk ops to form a computation
  - — But DB people have recognized imperative data access is bad for nearly 40 years



Chris worked on the Pliny project?"

Good if few people supervised by Chris

Deja Vu, All Over Again

- I’m teaching a data science class this semester at Rice
  - — Assignment 3: Use Spark to analyze NYC taxi GPS data
  - — One task: find drop-off anomalies and geographic points-of-interest nearby
  - — “Prof. Chris: My last join hasn’t finished in two hours!!”

Deja Vu, All Over Again

```
#Map the above to CellDayHour, ratio numDrop/avgDrops, numDrops then
#take the top 20 ratios
output3 = output2.map(lambda p:(p[0], (p[1]/p[2], p[1]))).takeOrdered
   (20, lambda (key, value): -value[0])
#Turn back into RDD
output4 = sc.parallelize(output3)
```

Here's a fast solution

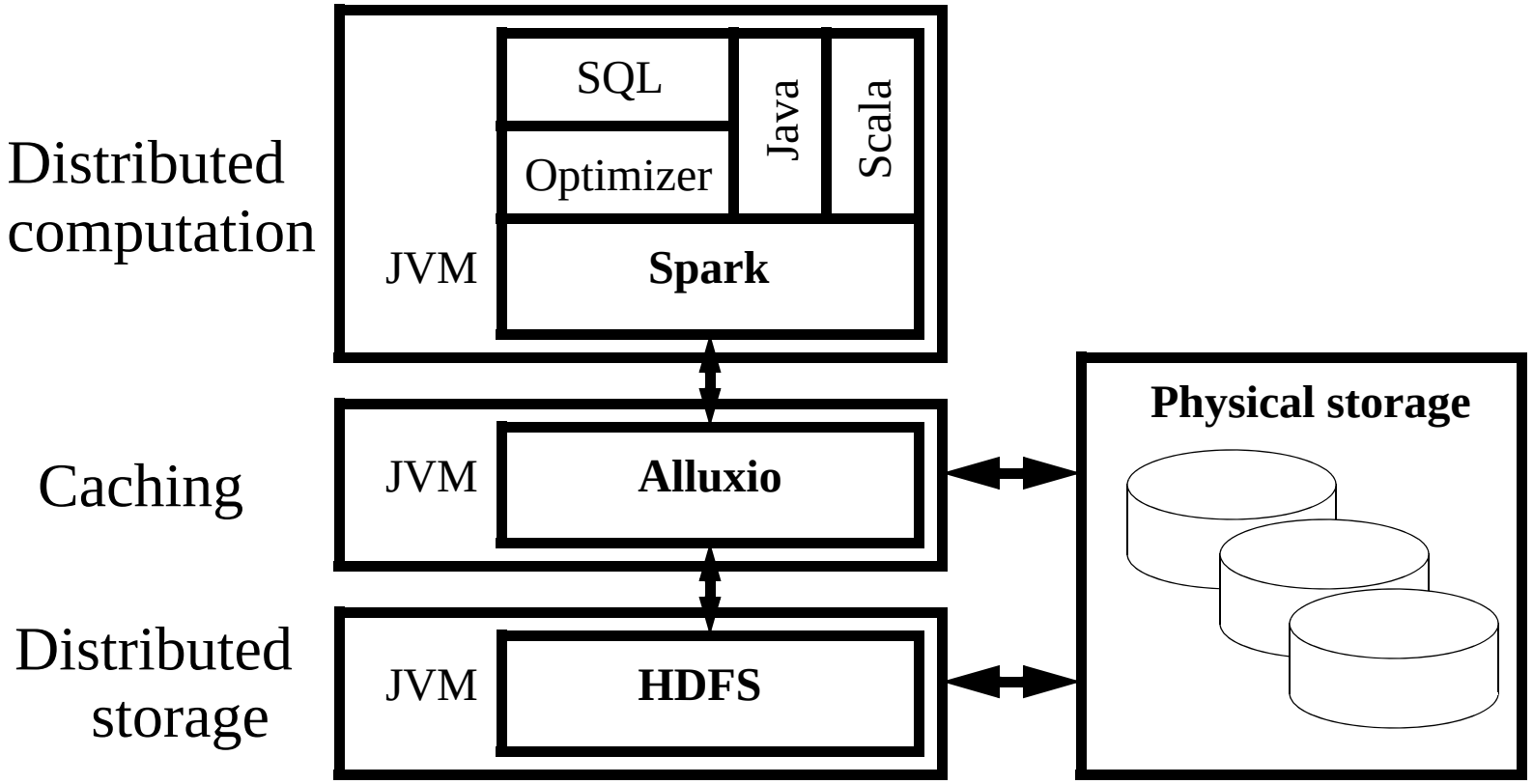
```
#Change formatting of the output
output5 = output4.map(lambda p: (str(p[0].split(":")[0]),
  [p[0].split(":")[1].split("")[1],p[0].split(":")[1].split(" ")[0],
  p[1]])).keyBy(lambda p: p[0])

#Map locations to grid cells
mappedData = mappings.map(lambda m: (str(changeValToCell
  (float(m.split("||")[1]))) + str(changeValToCell
  (float(m.split("||")[0]))), (m.split("||")[2]))).keyBy(lambda p:
  p[0]).reduceByKey(lambda t1, t2: (t1))

#Join the mappings to the data together
  final_output = output5.leftOuterJoin(mappedData).map(lambda p: (p[0],
  p[1][0][1][0], p[1][0][1][1],p[1][0][1][2][0] ,p[1][0][1][2][1]))
```

Deja Vu, All Over Again

```
# Now that it's sorted, put it back to (grid_cell,
# (hour, date, fraction, number))
grid_cell_combined = grid_cell_combined.map(lambda t: (t[1][0].split
  (":")[0], (t[1][0].split(":")[1], t[1][1],t[0], t[1][2])))

# now let's grab the POI data
lines = sc.textFile(sys.argv[2], 1)
poi_lines = lines.map(lambda x: x.split('||')).map(lambda l:
  (get_cell_id(l[0], l[1]), [l[2]])).reduceByKey(lambda list1,
  list2: list1 + list2)

grid_with_poi = grid_cell_combined.leftOuterJoin(poi_lines).map(
   lambda t: (t[1][0][2], (t[0], t[1][0][0], t[1][0][1], t[1][0][3],
   t[1][1]))).sortByKey(ascending=False).map(
   lambda t: (t[1][0], t[1][1], t[1][2], t[0], t[1][3], t[1][4]))

test_taxi_rows = sc.parallelize(grid_with_poi.take(20))

test_taxi_rows.saveAsTextFile(sys.argv[3])
```

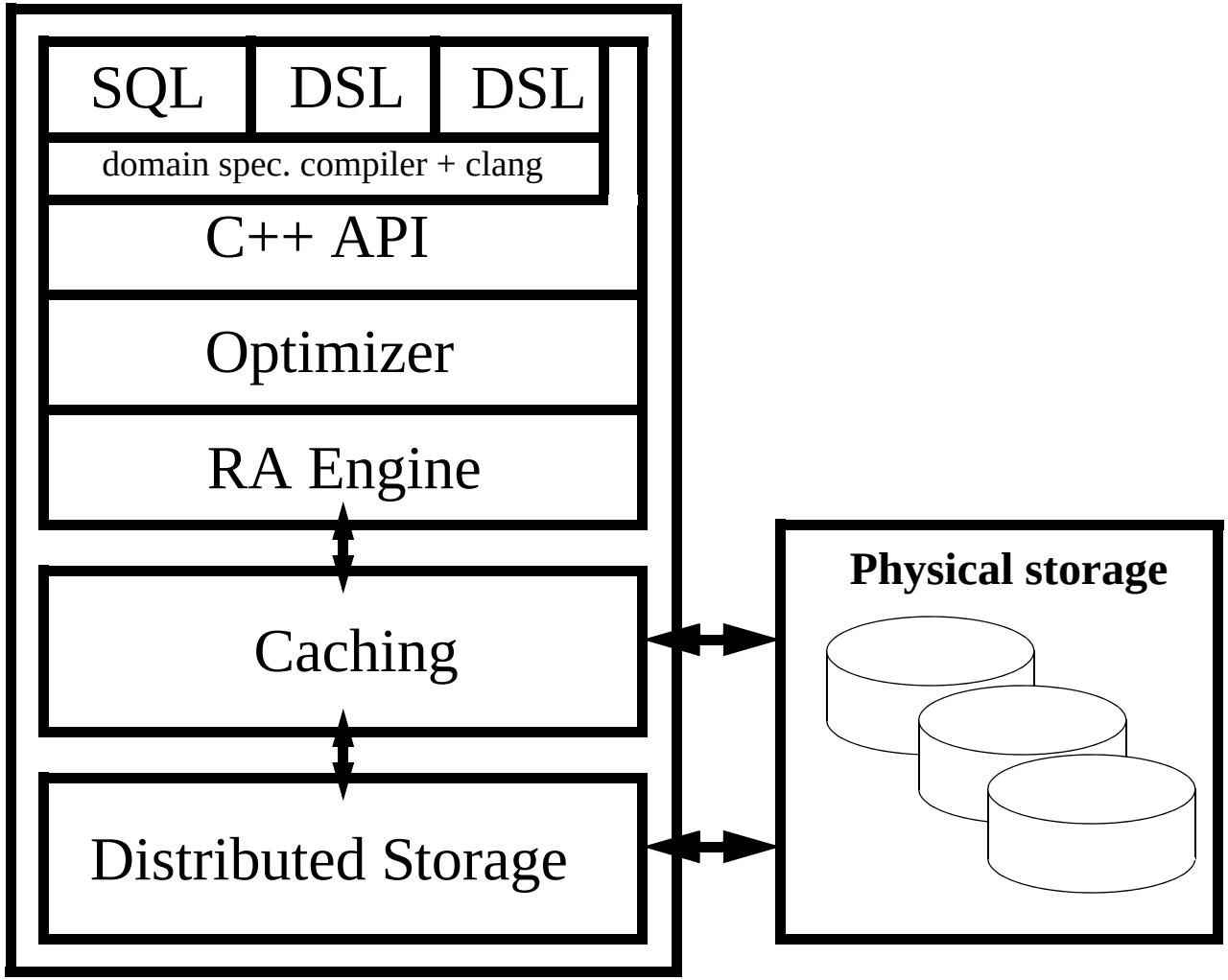
And here's a slow one

Is This Progress?



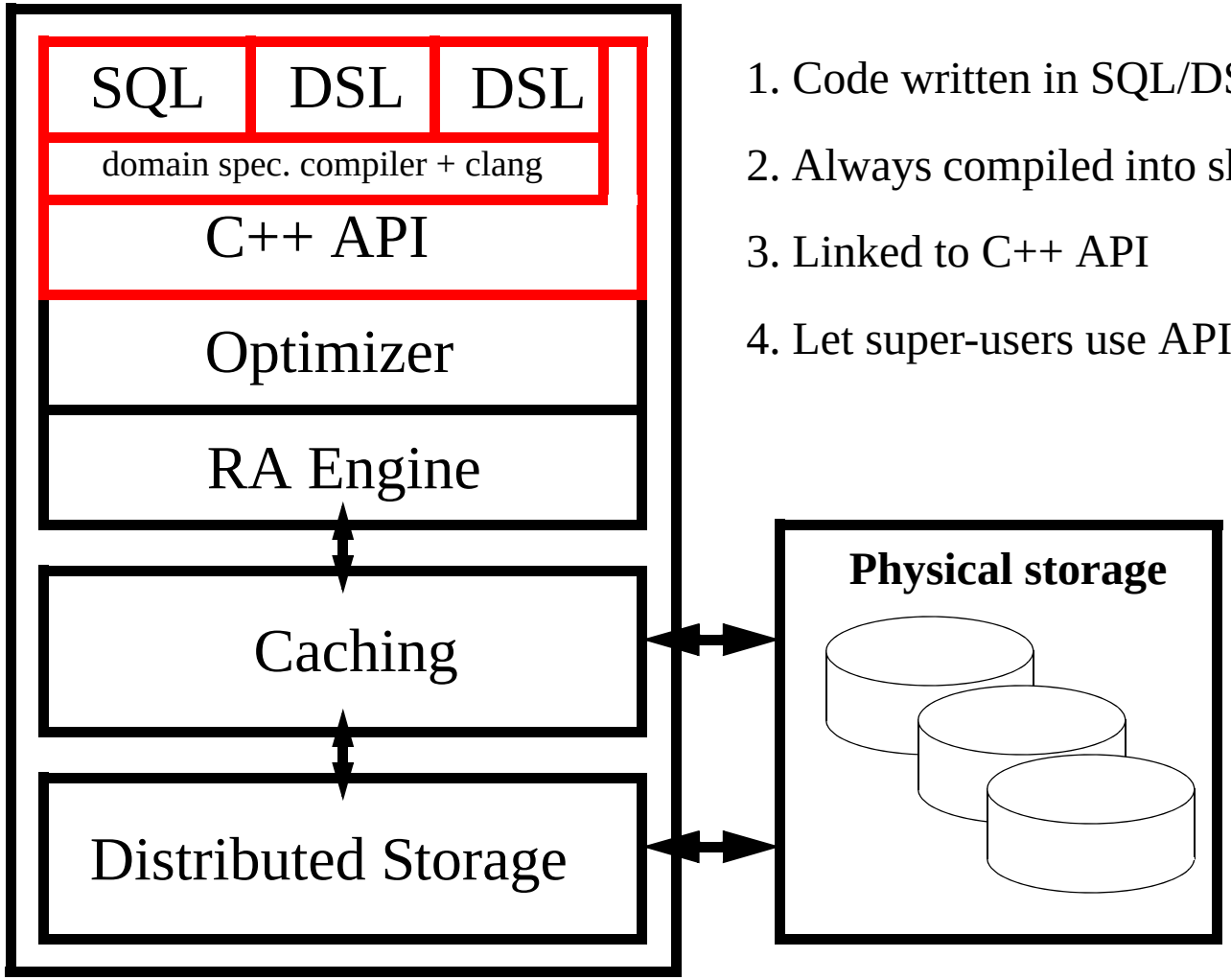
Our Vision: The Pliny Compute Platform

Pliny



Our Vision: The Pliny Compute Platform

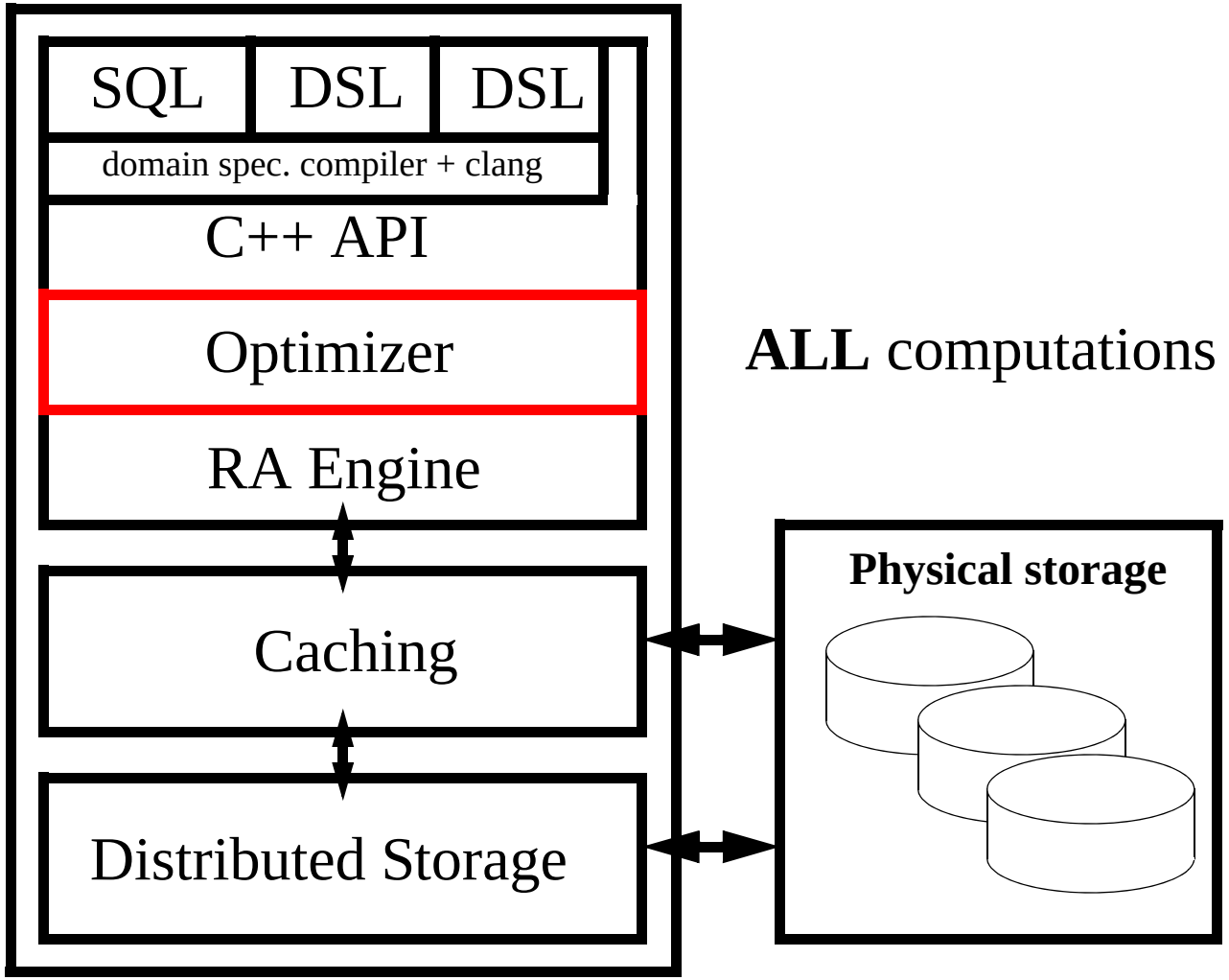
Pliny



1. Code written in SQL/DSL
2. Always compiled into shared libs
3. Linked to C++ API
4. Let super-users use API directly

Our Vision: The Pliny Compute Platform

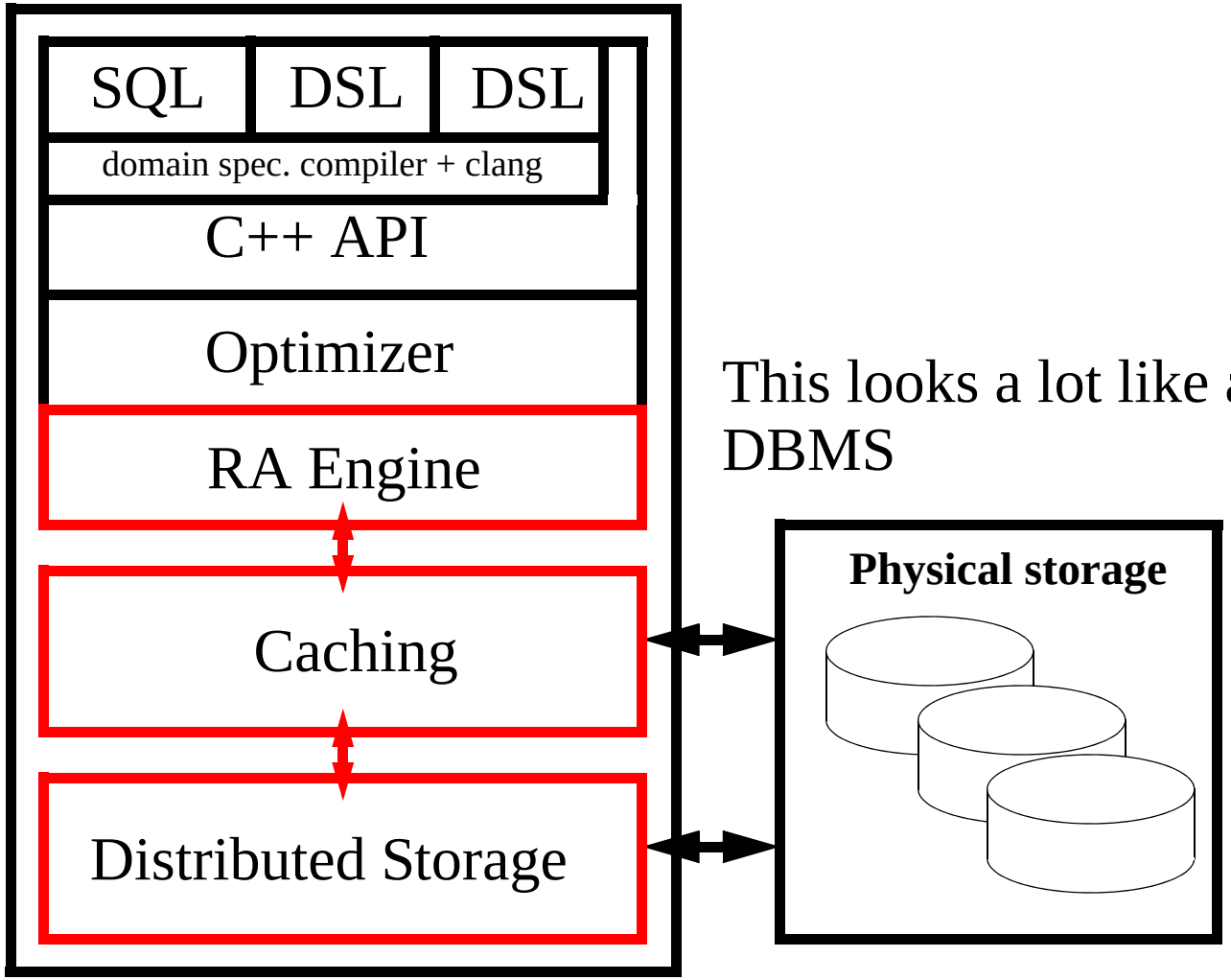
Pliny



ALL computations thru opt.

Our Vision: The Pliny Compute Platform

Pliny



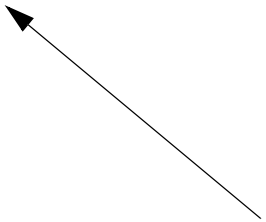
This looks a lot like a vectorized DBMS

Many Research Problems Here!

- How to make SQL a more suitable language for analytics?
    1. Fancier table functions (“VG functions”)
    2. Add native support for vectors/matrices (as att types)
    3. Support for executing huge “query” plans (1000’s of operations)
- What should new declarative DSLs for analytics look like?
- How does the optimizer change to support all of these front-end interfaces?
- How should the compute engine change?

Our Variant of SQL: SimSQL

- Most fundamental SQL addition is "VG Function" abstraction
- Called via a special `CREATE TABLE` statement
- Example; assuming:
  - `SBP(MEAN, STD, GENDER)`
  - `PATIENTS(NAME, GENDER)`
- To create a derived table, we might have:



We like randomized algs (MCMC) but not limited to that

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```

How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```

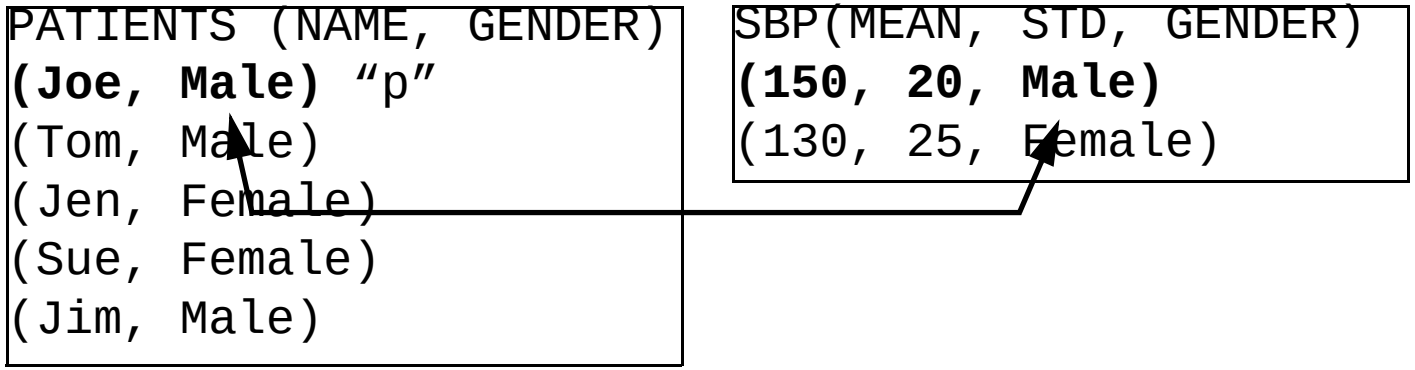
Loop through PATIENTS

PATIENTS (NAME, GENDER)
(Joe, Male) “p”
(Tom, Male)
(Jen, Female)
(Sue, Female)
(Jim, Male)

SBP(MEAN, STD, GENDER)
(150, 20, Male)
(130, 25, Female)

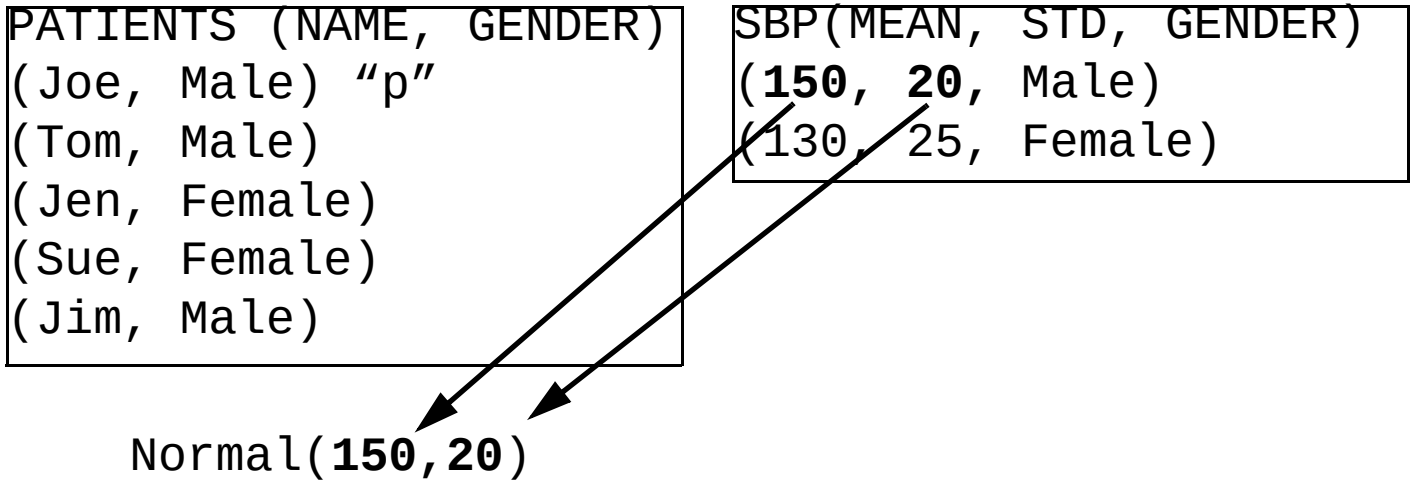
How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```

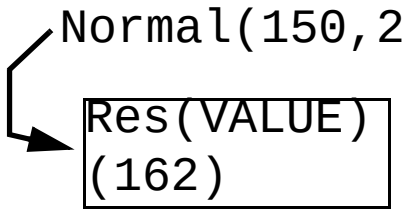


How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```

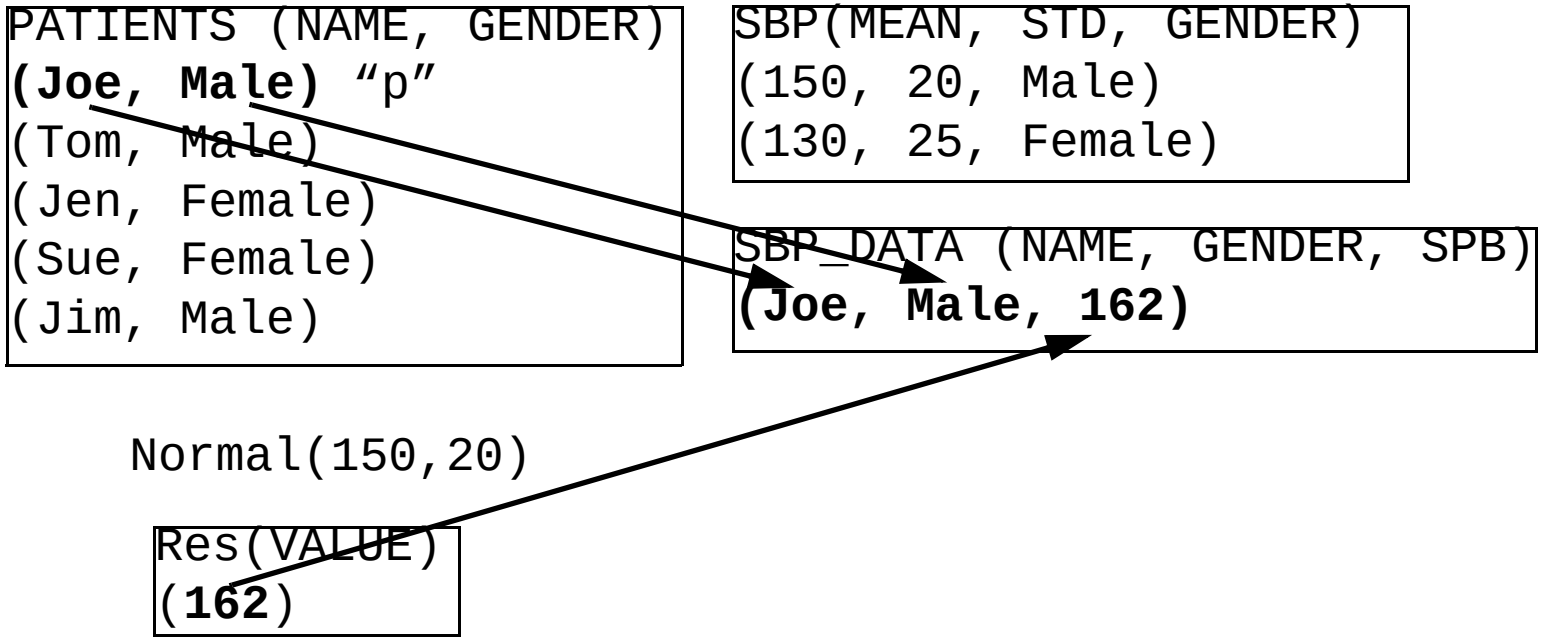
```
PATIENTS (NAME, GENDER)
(Joe, Male) “p”
(Tom, Male)
(Jen, Female)
(Sue, Female)
(Jim, Male)
```

```
SBP(MEAN, STD, GENDER)
(150, 20, Male)
(130, 25, Female)
```



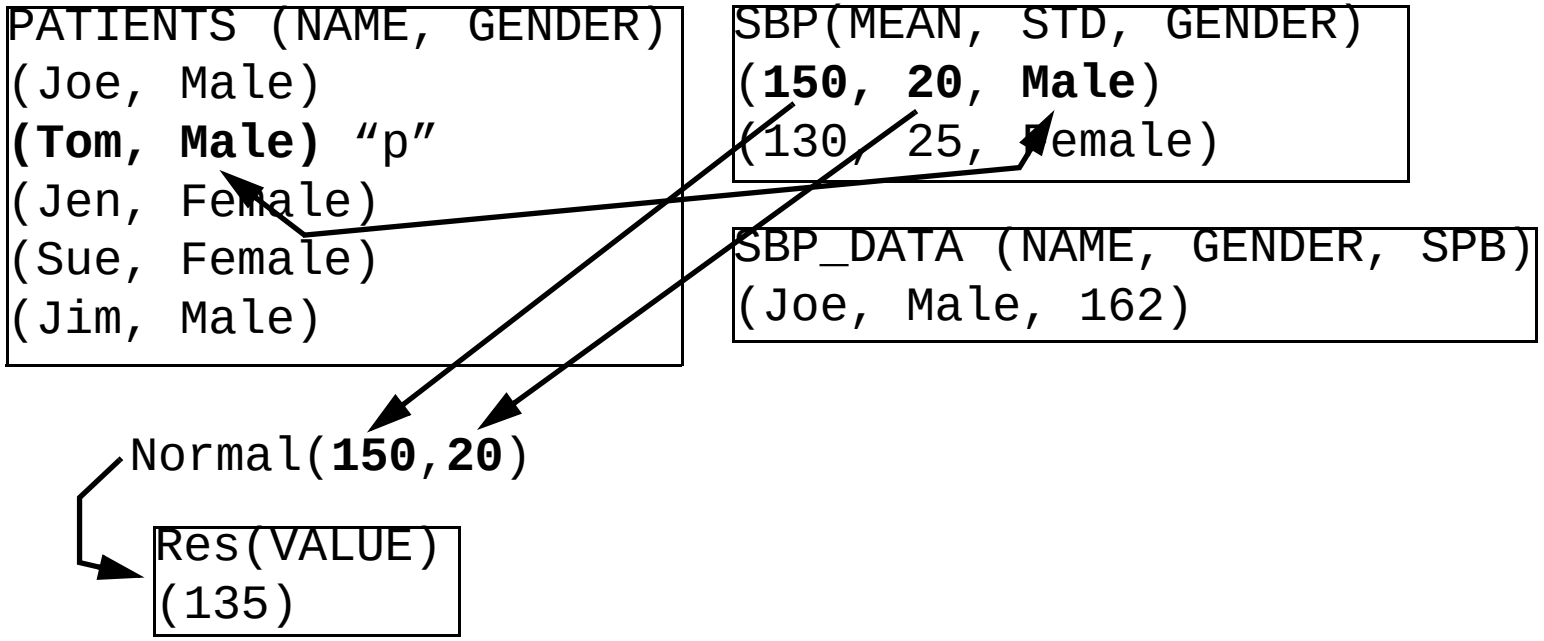
How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



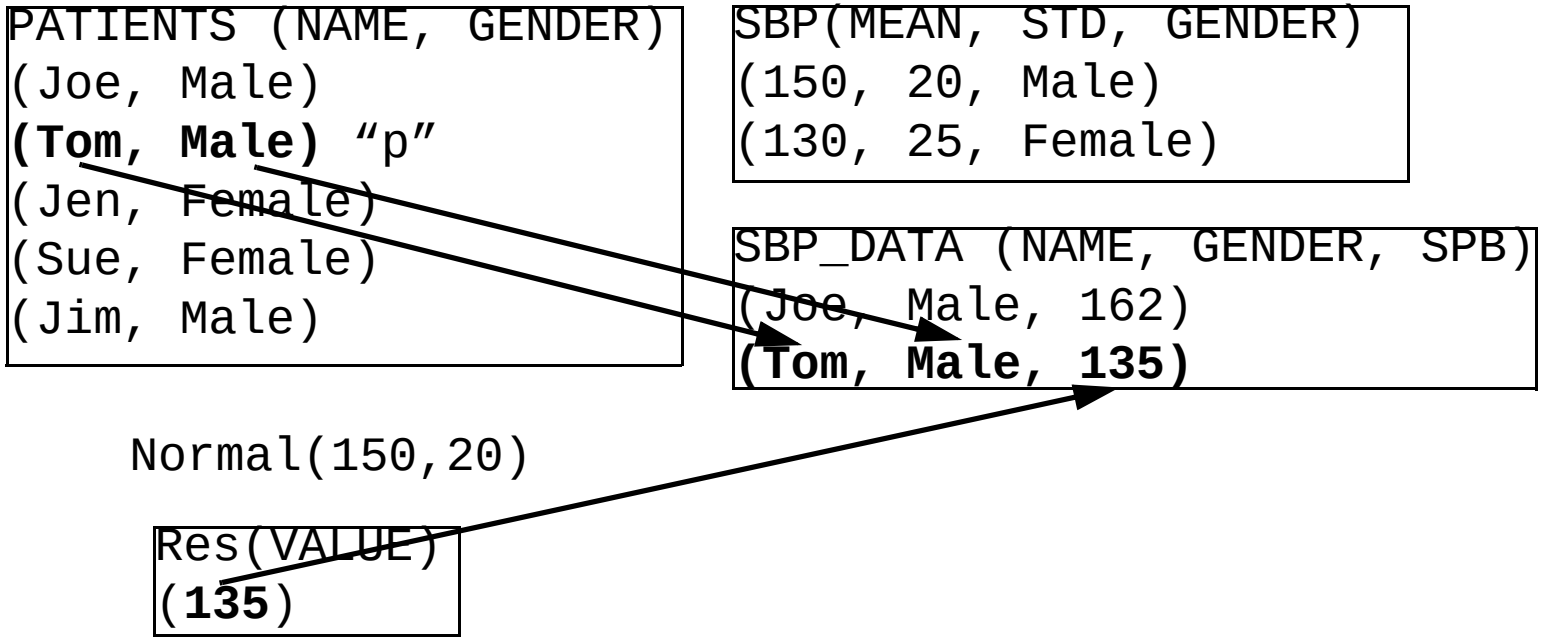
How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



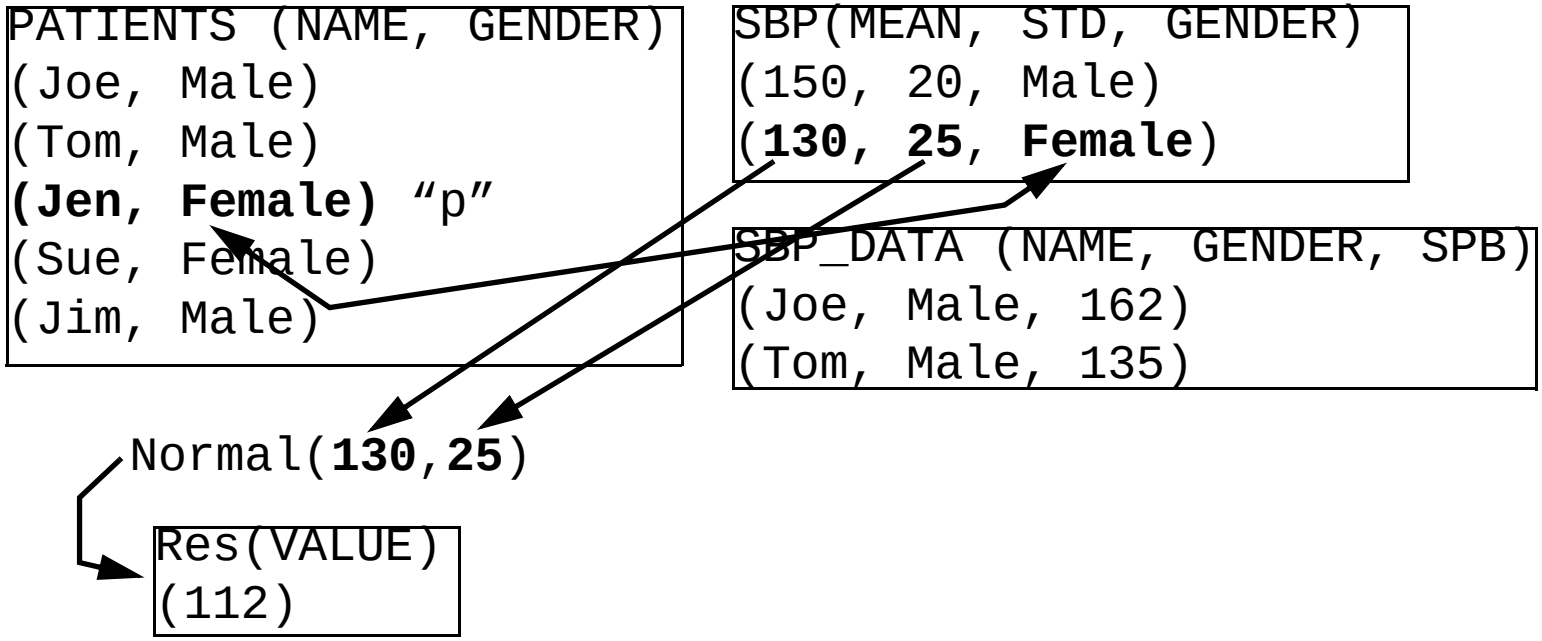
How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



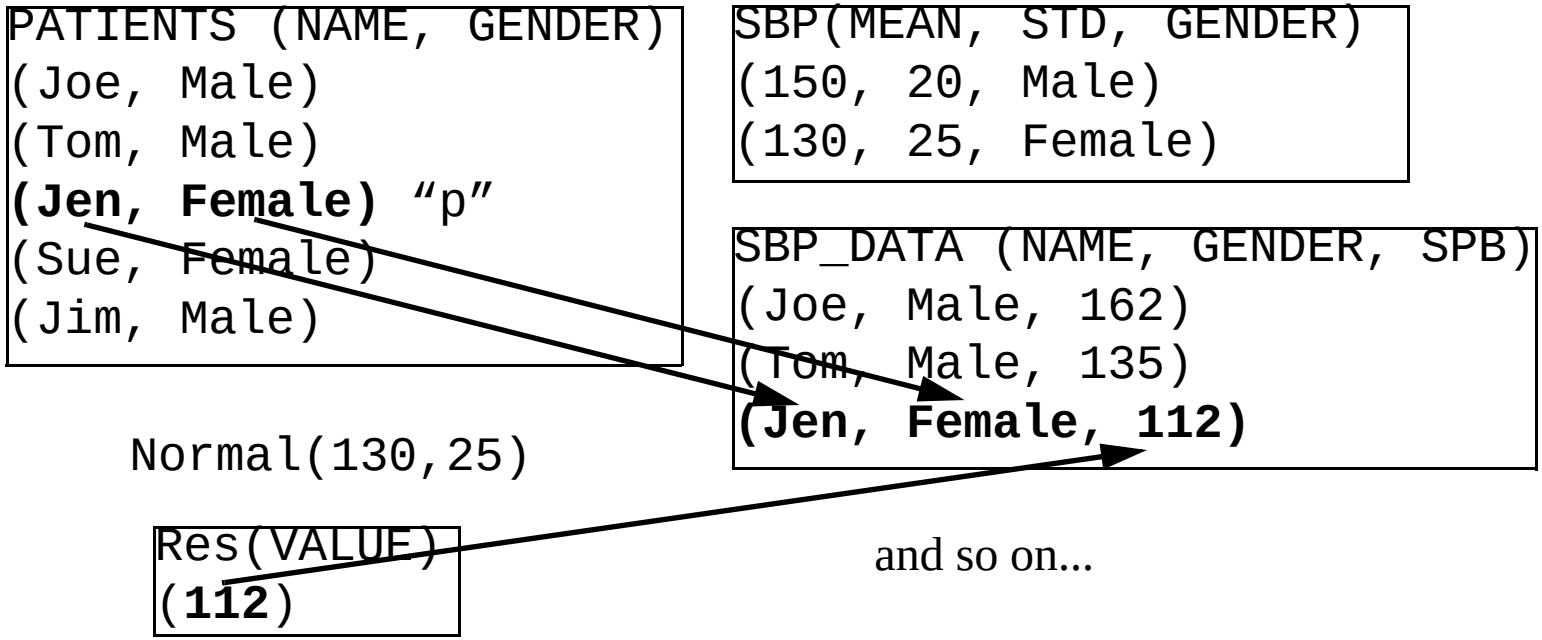
How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



How Does This Work?

```
CREATE TABLE SBP_DATA(NAME, GENDER, SBP) AS
FOR EACH p in PATIENTS
  WITH Res AS Normal (
    SELECT s.MEAN, s.STD
    FROM SPB s WHERE s.GENDER = p.GENDER)
  SELECT p.NAME, p.GENDER, r.VALUE
  FROM Res r
```



More Complicated Computations

- Previous allows (for example) table-valued RVs
- But Markov chains are easy in SimSQL, so Bayesian ML easy
- Here's a silly Markov chain. We have:
  - `PERSON (pname)`
  - `PATH (fromCity, toCity, prob)`
  - `RESTAURANT (city, rname, prob)`

Markov Chain Simulation

- To select an initial starting position for each person:

```
CREATE TABLE POSITION[0] (pname, city) AS
FOR EACH p IN PERSON
  WITH City AS DiscreteChoice (
    SELECT r DISTINCT toCity
    FROM PATH)
  SELECT p.pname, City.value
  FROM City
```

Markov Chain Simulation

- And then randomly select a restaurant:

```
CREATE TABLE VISITED[i] (pname, rname) AS
FOR EACH p IN PERSON
  WITH Visit AS Categorical (
    SELECT r.rname, r.prob
    FROM RESTAURANT r, POSITION[i] l
    WHERE r.city = l.city AND l.pname = p.pname)
  SELECT p.pname, Visit.val
  FROM Visit
```

Markov Chain Simulation

- And transition the person:

```
CREATE TABLE POSITION[i] (pname, city) AS
FOR EACH p IN PERSON
  WITH Next AS Categorical (
    SELECT PATH.tocity, PATH.prob
    FROM PATH, POSITION[i - 1] l
    WHERE PATH.fromcity = l.city AND l.pname = p.pname)
  SELECT p.pname, Next.val
  FROM Next
```

Markov Chain Simulation

- And transition the person:

```
CREATE TABLE POSITION[i] (pname, city) AS
FOR EACH p IN PERSON
  WITH Next AS Categorical (
    SELECT PATH.tocity, PATH.prob
    FROM PATH, POSITION[i - 1] l
    WHERE PATH.fromcity = l.city AND l.pname = p.pname)
  SELECT p.pname, Next.val
  FROM Next
```

- Fully spec’ed a distributed Markov chain!

Native Vector and Matrix Support

- Vectors and matrices fundamental to analytics

Native Vector and Matrix Support

- Vectors and matrices fundamental to analytics
- Can be difficult to write code/expensive without it

$$d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}') = (\mathbf{x}_i - \mathbf{x}')^T \mathbf{A} (\mathbf{x}_i - \mathbf{x}')$$

Native Vector and Matrix Support

- Vectors and matrices fundamental to analytics
- Can be difficult to write code/expensive without it

$$d_{\mathbf{A}}^2(\mathbf{x}_i, \mathbf{x}') = (\mathbf{x}_i - \mathbf{x}')^T \mathbf{A} (\mathbf{x}_i - \mathbf{x}')$$

Classical SQL

```
data (pointID INTEGER, dimID INTEGER, value DOUBLE)
matrixA (row INTEGER, col INTEGER, value DOUBLE)

CREATE VIEW xDiff (pointID, value) AS
  SELECT x2.pointID, x1.value - x2.value
  FROM data AS x1, data AS x2
  WHERE x1.pointID = i and x1.dim = x2.dim

SELECT x.pointID, SUM (firstPart.value * x.value)
FROM (SELECT x.pointID, a.colID, SUM (a.value, x.value) AS value
      FROM xDiff as X, matrixA AS a
      WHERE x.dimID = a.rowID
      GROUP BY x.pointID, a.colID) AS firstPart, xDiff AS x
WHERE firstPart.colID = x.dimID AND firstPart.pointID = x.pointID
GROUP BY x.pointID
```

Native Vector and Matrix Support

- Vectors and matrices fundamental to analytics
- Can be difficult to write code/expensive without it

$$d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}') = (\mathbf{x}_i - \mathbf{x}')^T \mathbf{A} (\mathbf{x}_i - \mathbf{x}')$$

SimSQL

```
data (pointID INTEGER, val VECTOR [])
matrixA (val MATRIX [][])

SELECT x2.pointID, inner_product (
        matrix_vector_multiply (
          x1.val - x2.val, a.val), x1.val - x2.val) AS value
FROM data AS x1, data AS x2, matrixA AS a
WHERE x1.pointID = i
```

Native Vector and Matrix Support

- Another example: linear regression
- Goal is to compute

$$\hat{\beta} = \left(\sum_i \mathbf{x}_i \mathbf{x}_i^T \right)^{-1} \left(\sum_i \mathbf{x}_i y_i \right)$$

Native Vector and Matrix Support

- Another example: linear regression
- Goal is to compute

$$\hat{\beta} = \left(\sum_i \mathbf{x}_i \mathbf{x}_i^T \right)^{-1} \left(\sum_i \mathbf{x}_i y_i \right)$$

```
CREATE TABLE X (
  i INTEGER,
  x_i VECTOR []);

CREATE TABLE y (
  i INTEGER,
  y_i DOUBLE);

SELECT matrix_vector_multiply (matrix_inverse (
  SUM (outer_product (X.x_i, X.x_i))),
  SUM (X.x_i * y_i))
FROM X, y
WHERE X.i = y.i
```

Easy in SimSQL!

Promotion and Demotion

- Start with pure, tuple-based encoding

```
mat (row INTEGER, col INTEGER, value DOUBLE)
```

Promotion and Demotion

- Start with pure, tuple-based encoding
- Move to a set of (vector, rowID) pairs

```
mat (row INTEGER, col INTEGER, value DOUBLE)

CREATE VIEW vecs (vec, row) AS
  SELECT VECTORIZE (label_scalar (val, col)) AS vec, row
  FROM mat
  GROUP BY row
```

Promotion and Demotion

- Start with pure, tuple-based encoding
- Move to a set of (vector, rowID) pairs
- Then create a matrix...

```
mat (row INTEGER, col INTEGER, value DOUBLE)

CREATE VIEW vecs (vec, row) AS
  SELECT VECTORIZE (label_scalar (val, col)) AS vec, row
  FROM mat
  GROUP BY row

SELECT ROWMATRIX (label_vector (vec, row))
FROM vecs
```

Promotion and Demotion

- Start with pure, tuple-based encoding
- Move to a set of (vector, rowID) pairs
- Then create a matrix... or move back to tuples

```
mat (row INTEGER, col INTEGER, value DOUBLE)

CREATE VIEW vecs (vec, row) AS
  SELECT VECTORIZE (label_scalar (val, col)) AS vec, row
  FROM mat
  GROUP BY row

SELECT ROWMATRIX (label_vector (vec, row))
FROM vecs

SELECT v.row, c.cnt AS col, get_scalar (v.vec, c.cnt) AS v.value
FROM vecs AS v, counts AS c
```

Additional DSLs: BUDS

- SQL might not be most natural declarative interface for analytics

Additional DSLs: BUDS

- SQL might not be most natural declarative interface for analytics
- Math for the Bayesian Lasso, lifted from original paper

1. $r \sim \text{Normal}(\mathbf{A}^{-1}\mathbf{X}^T\tilde{y}, \sigma^2\mathbf{A}^{-1})$

2. $\sigma^2 \sim \text{InvGamma}\left(\frac{(n-1)+p}{2}, \frac{(\tilde{y}-\mathbf{X}r)^T(\tilde{y}-\mathbf{X}r)^T + r^T\mathbf{D}^{-1}r}{2}\right)$

3. $\tau_j^{-2} \sim \text{InvGaussian}\left(\frac{\lambda\sigma}{r_j}, \lambda^2\right)$

— where $\mathbf{A} = \mathbf{X}^T\mathbf{X} + \mathbf{D}^{-1}$, $\mathbf{D}^{-1} = diag(\tau_1^{-2}, \tau_2^{-2}, \ldots)$

SQL Is Not Trivial

```
CREATE TABLE response(
       respID INTEGER,
       respValue DOUBLE,
       PRIMARY KEY (respID)
);
CREATE TABLE regressor(
       respID INTEGER,
       regValue VECTOR[],
       PRIMARY KEY (respID)
);
CREATE TABLE prior(
       numResponses INTEGER,
       numRegressors INTEGER,
       lambdaValue DOUBLE,
       invGammaShape DOUBLE,
       invGammaScale DOUBLE,
       invGaussianMean VECTOR[],
       invGaussianShape DOUBLE
);

CREATE VIEW centeredResponse(respID, respValue) AS
  SELECT r1.respID, (r1.respValue - m.meanRespValue)
  FROM response r1,
       (SELECT AVG(r2.respValue) AS meanRespValue
        FROM response r2) AS m;

CREATE VIEW regressorGram(val) AS
  SELECT SUM(OUTER_PRODUCT(r.regValue, r.regValue))
  FROM regressor r;

CREATE VIEW regressorSum(sumValue) AS
  SELECT SUM(reg.regValue * res.respValue)
  FROM regressor reg, centeredResponse res
  WHERE reg.respID = res.respID;


CREATE TABLE sigma[0](sigmaValue) AS
  WITH g AS InvGamma (SELECT invGammaShape, invGammaScale FROM prior)
  SELECT g.outValue
  FROM g;
```

```
CREATE TABLE tau[0](tauValue) AS
  WITH ig AS InvGaussian_VM(SELECT invGaussianMean, invGaussianShape FROM prior)
  SELECT ig.outValue
  FROM ig;

CREATE TABLE A[i](AValue) AS
  SELECT MATRIX_INVERSE(xtx.val + DIAG_MATRIX(t.tauValue))
  FROM tau[i] t, regressorGram xtx;

CREATE TABLE beta[i](betaValue) AS
  WITH mvn AS MultiNormal_VM(
     (SELECT MATRIX_VECTOR_MULTIPLY(a.AValue, rs.sumValue), (a.AValue * s.sigmaValue)
      FROM A[i] a, regressorSum rs, sigma[i] s))
  SELECT mvn.out_mean
  FROM mvn;

CREATE TABLE sigma[i](sigmaValue) AS
  WITH g as InvGamma(
    (SELECT (pr.numResponses - 1)/2.0 + (pr.numRegressors/2.0)
     FROM prior pr),
    (SELECT sb.sumBetas + st.sumTaus FROM
    (SELECT SUM(((res.respValue - xb.sumValue) * (res.respValue - xb.sumValue)) / 2.0) AS sumBetas
      FROM centeredResponse res,
         (SELECT reg.respID as respID, INNER_PRODUCT(reg.regValue, b1.betaValue) as sumValue
            FROM regressor reg, beta[i-1] b1) AS xb
      WHERE res.respID = xb.respID) AS sb,
    (SELECT (INNER_PRODUCT(b2.betaValue * b2.betaValue, t.tauValue) / 2.0) AS sumTaus
      FROM tau[i-1] t, beta[i-1] b2) AS st))
  SELECT g.outValue
  FROM g;

CREATE TABLE tau[i](tauValue) AS
    WITH ig AS InvGaussian_VM(
     (SELECT SQRT_VECTOR((pr1.lambdaValue * pr1.lambdaValue * s.sigmaValue) /
      (b.betaValue * b.betaValue))
        FROM prior pr1, sigma[i] s, beta[i-1] b),
       (SELECT (pr2.lambdaValue * pr2.lambdaValue)
        FROM prior pr2)),
  SELECT ig.outValue
  FROM ig;
```

BUDS Is Much Simpler!

- Write code that looks just like the math...

```
data {
  n: range (responses); p: range (regressors);
  X: array[n, p] of real; y: array[n] of real;
  lam: real
}

var {
  sig: real;
  r, t: array[p] of real; yy, Z: array[n] of real;
}

A <- inv(X ‘* X + diag(t));
yy <- (y[i] - mean(y) | i in 1:n);
Z <- yy - X * r;

init {
  sig ~ InvGamma (1, 1);
  t ~ (InvGauss (1, lam) | j in 1:p);
}

r ~ Normal (A *’ X * yy, sig * A);
sig ~ InvGamma(((n-1) + p)/2,
   (Z ‘* Z + (r * diag(t) ‘* r)) / 2);
for (j in 1:p) {
  t[j] ~ InvGauss (sqrt((lam * sig) / r[j]), lam);
}
```

Truly declarative;
Programmer only lists dependencies

BUDS Is Much Simpler!

- Write code that looks just like the math...

```
data {
  n: range (responses); p: range (regressors);
  X: array[n, p] of real; y: array[n] of real;
  lam: real
}

var {
  sig: real;
  r, t: array[p] of real; yy, Z: array[n] of real;
}

A <- inv(X '* X + diag(t));
yy <- (y[i] - mean(y) | i in 1:n);
Z <- yy - X * r;

init {
  sig ~ InvGamma (1, 1);
  t ~ (InvGauss (1, lam) | j in 1:p);
}

r ~ Normal (A *' X * yy, sig * A);
sig ~ InvGamma(((n-1) + p)/2,
   (Z '* Z + (r * diag(t) '* r)) / 2);
for (j in 1:p) {
  t[j] ~ InvGauss (sqrt((lam * sig) / r[j]), lam);
}
```

$$\mathbf{A} = \mathbf{X}^T\mathbf{X} + \mathbf{D}^{-1}$$

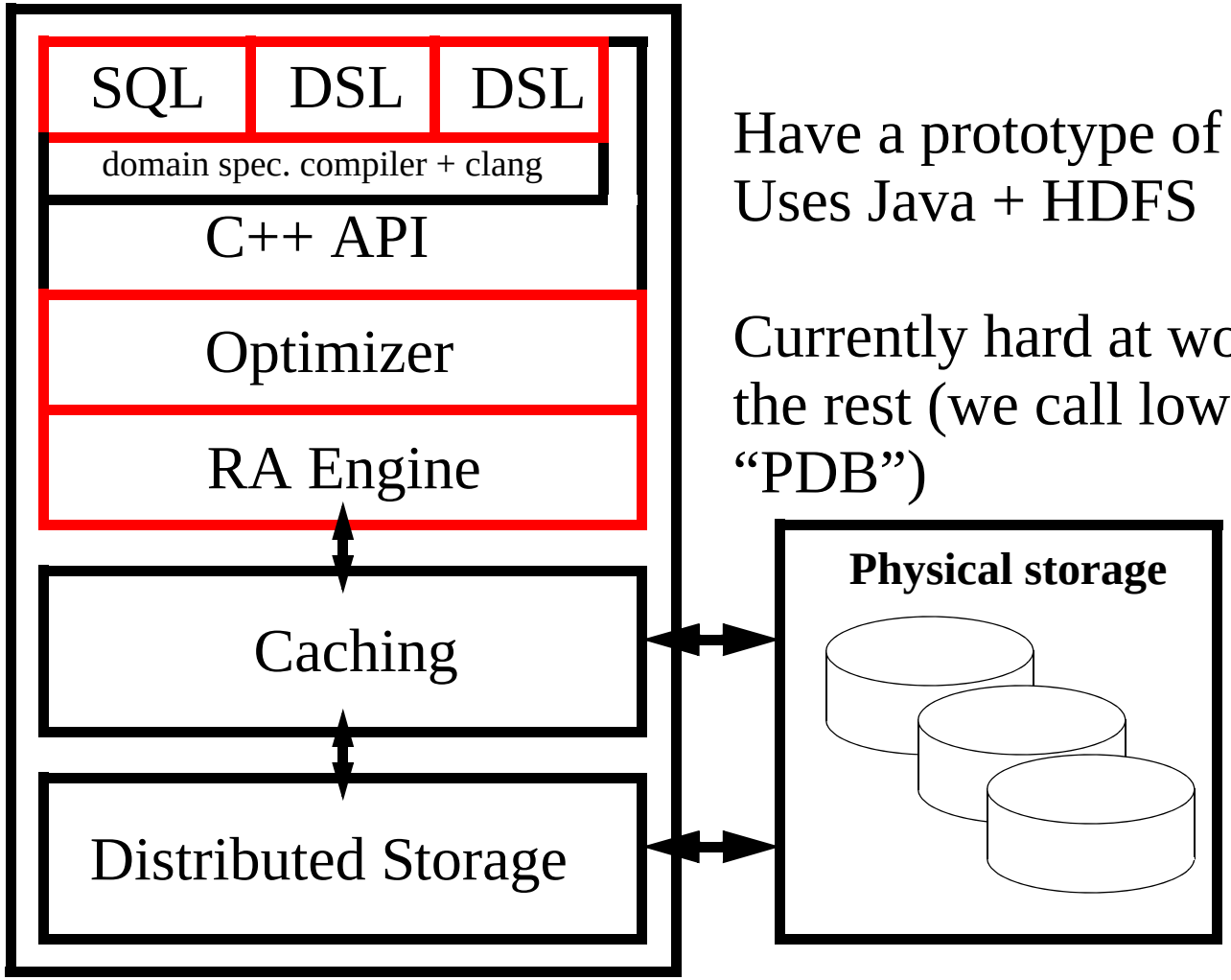
$$r \sim \text{Normal}(\mathbf{A}^{-1}\mathbf{X}^T\tilde{y}, \sigma^2\mathbf{A}^{-1})$$

$$\sigma^2 \sim \text{InvGamma}\left(\frac{(n-1)+p}{2}, \frac{(\tilde{y}-\mathbf{X}r)^T(\tilde{y}-\mathbf{X}r)^T + r^T\mathbf{D}^{-1}r}{2}\right)$$

$$\tau_j^{-2} \sim \text{InvGaussian}\left(\frac{\lambda\sigma}{r_j}, \lambda^2\right)$$

Project Status

Pliny



Have a prototype of **this**
Uses Java + HDFS

Currently hard at work on the rest (we call lower layers "PDB")

How Well Does All of This Work?

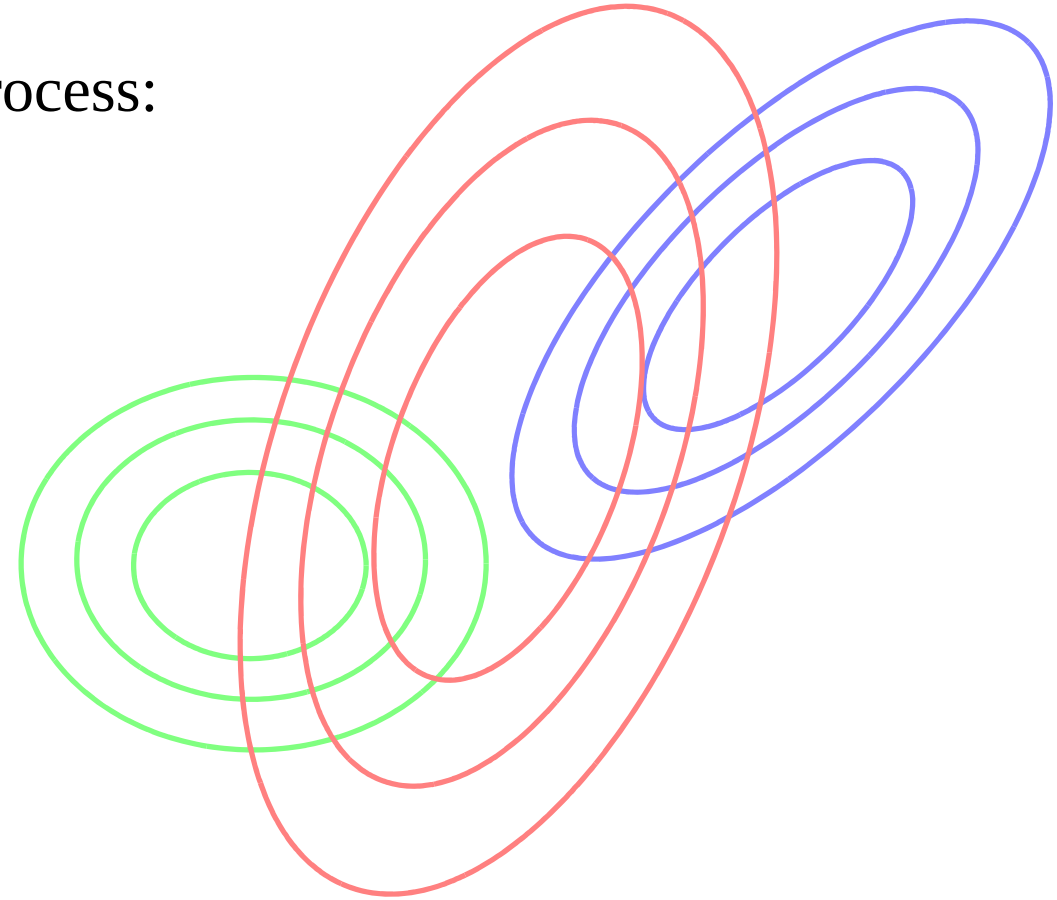
- Example: how does our SimSQL prototype compete with other "Big Data" platforms?
  - — After all, are many that count ML as the primary (or a motivating) application
  - — OptiML, GraphLab, SystemML, MLBase, ScalOps, Pregel, Giraph, Hama, Spark, Ricardo, Nyad, DradLinq
  - — How might those compare?

How Well Does All of This Work?

- We've done a **LOT** of comparisons with other mature platforms
  - — Specifically, GraphLab, Giraph, Spark
  - — More than 70,000 hours of Amazon EC2 time ($100,000 @on-demand price)
  - — I'd wager that few groups have a better understanding of how well these platforms work in practice!

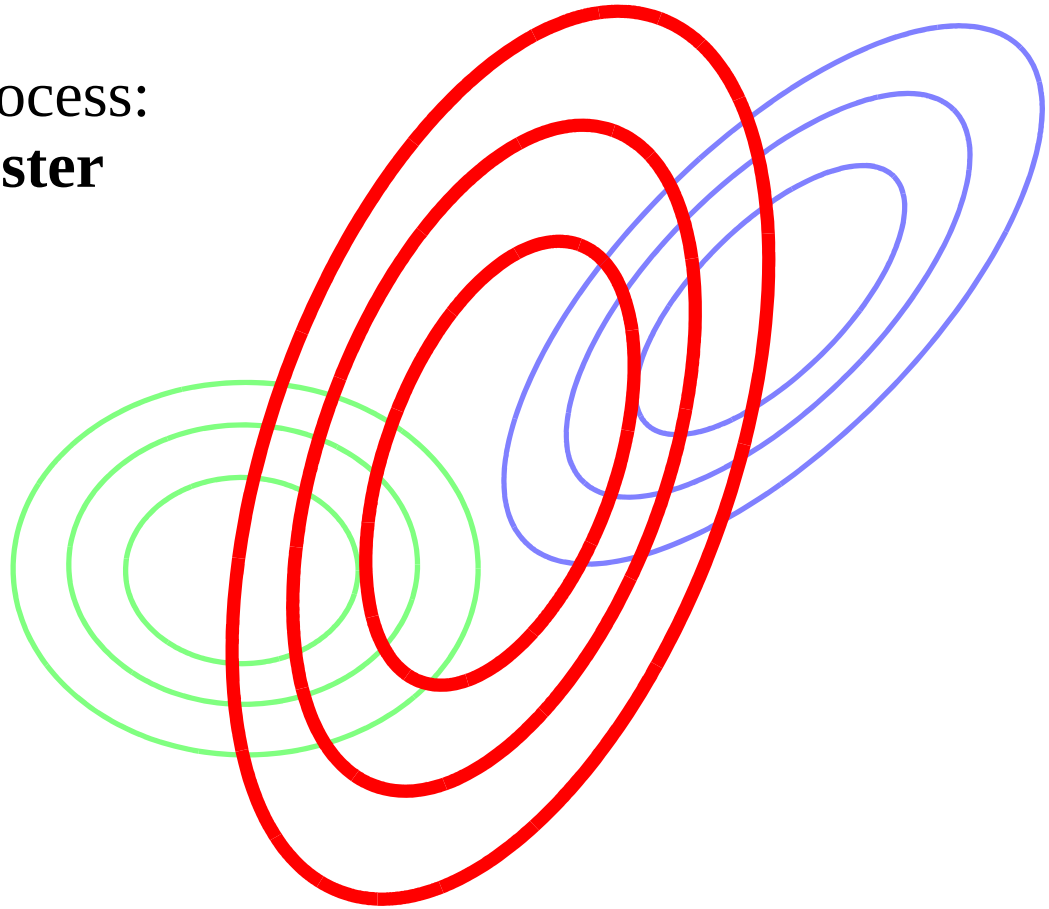
Example One: Bayesian GMM

Generative process:



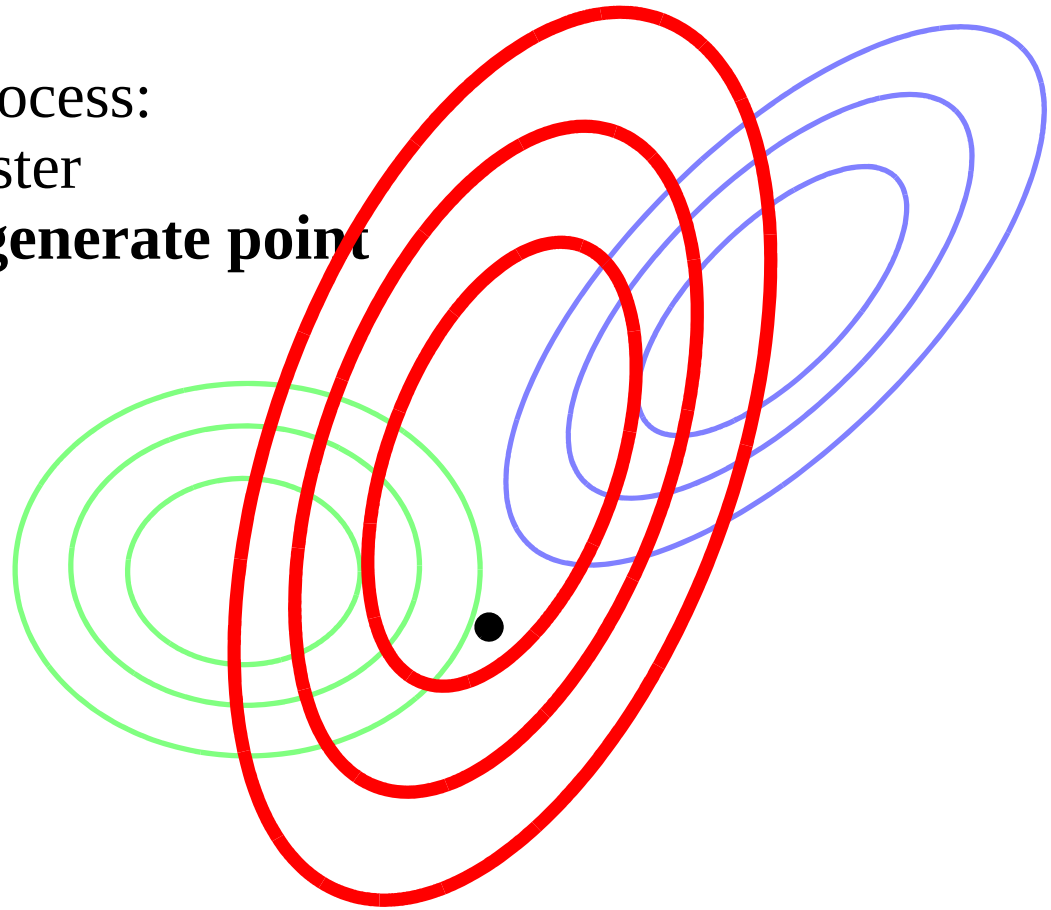
Example One: Bayesian GMM

Generative process:
(1) **Pick a cluster**



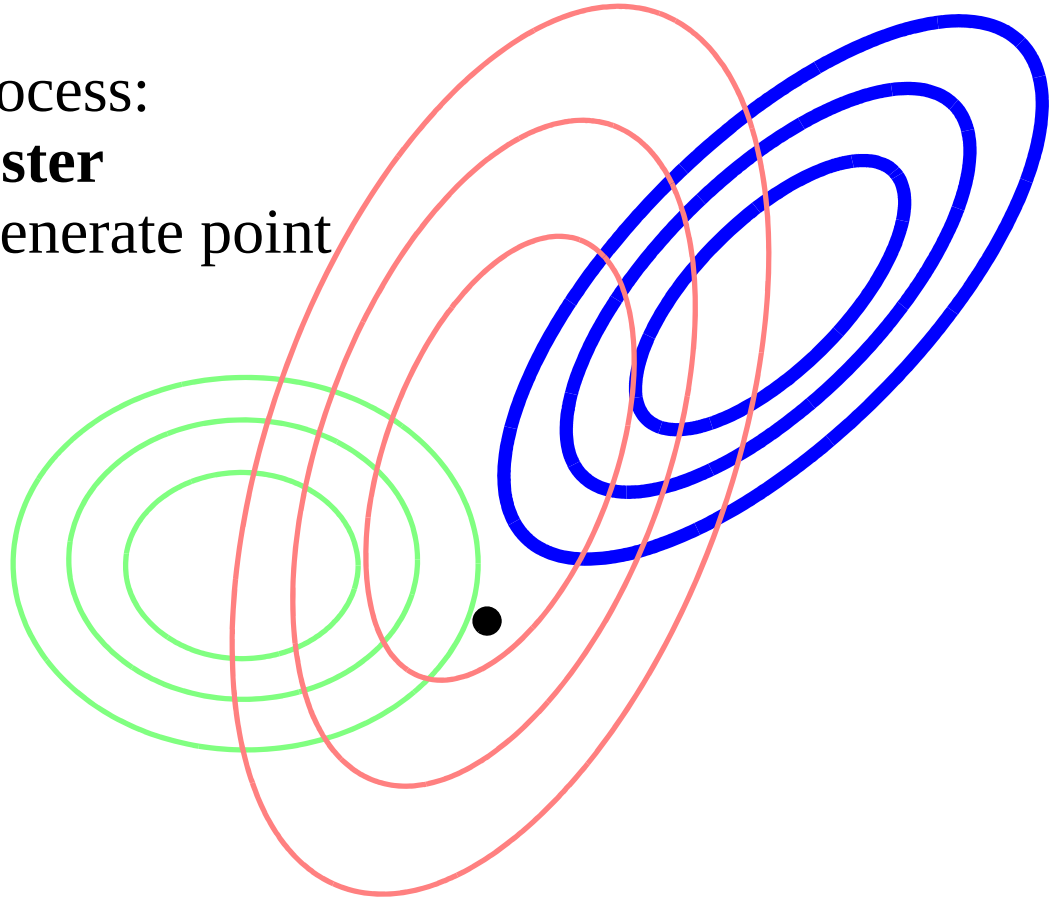
Example One: Bayesian GMM

Generative process:
(1) Pick a cluster
(2) **Use it to generate point**



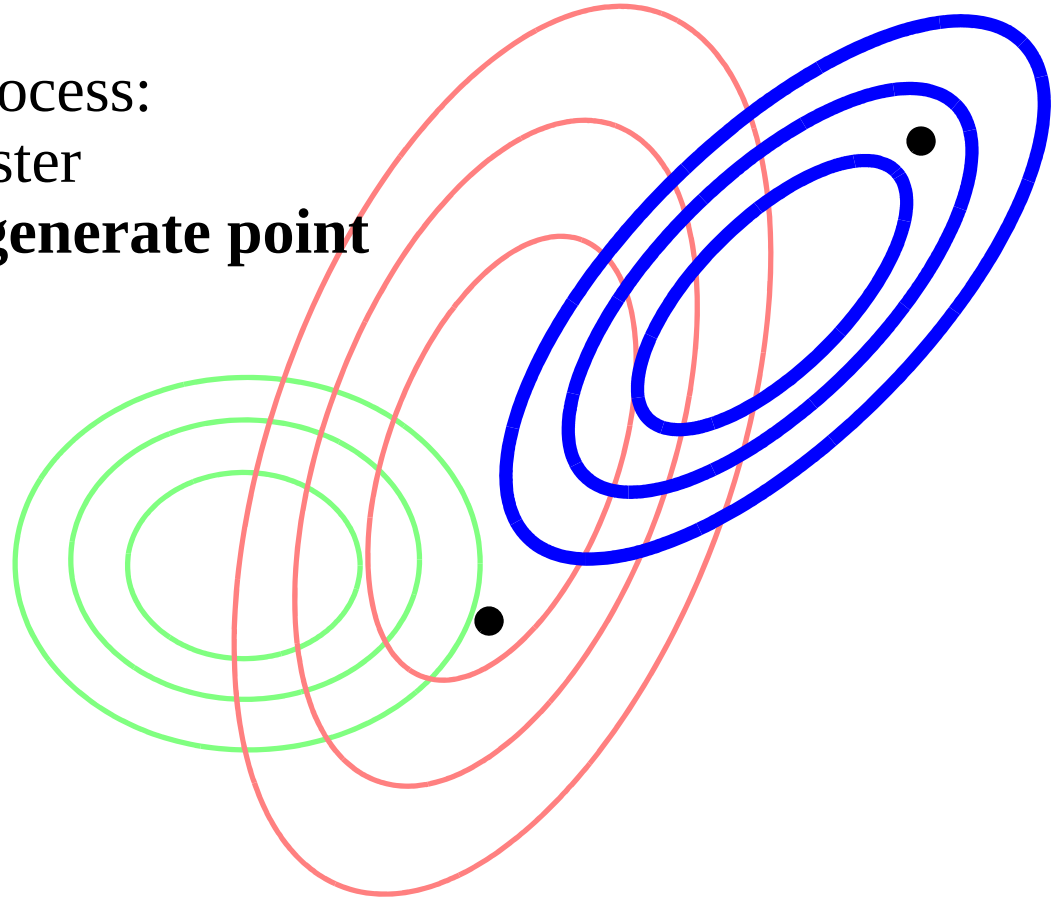
Example One: Bayesian GMM

Generative process:
(1) **Pick a cluster**
(2) Use it to generate point



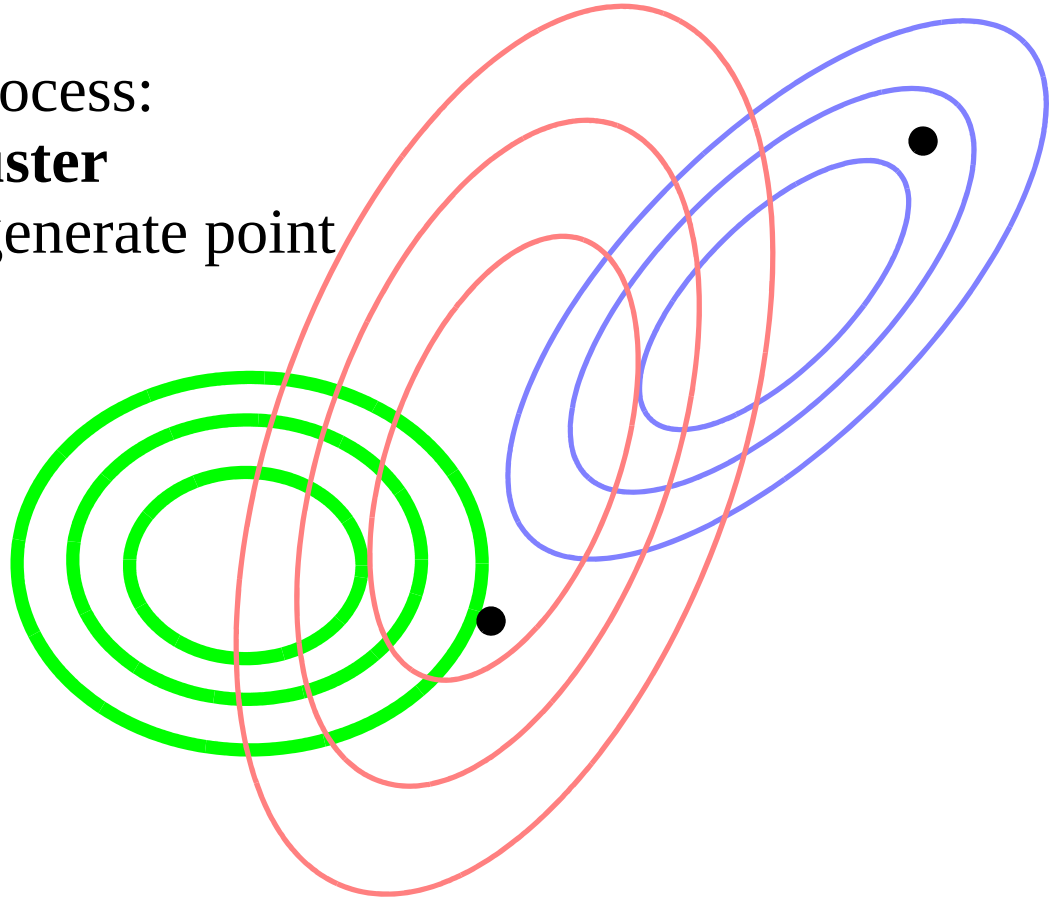
Example One: Bayesian GMM

Generative process:
(1) Pick a cluster
(2) **Use it to generate point**



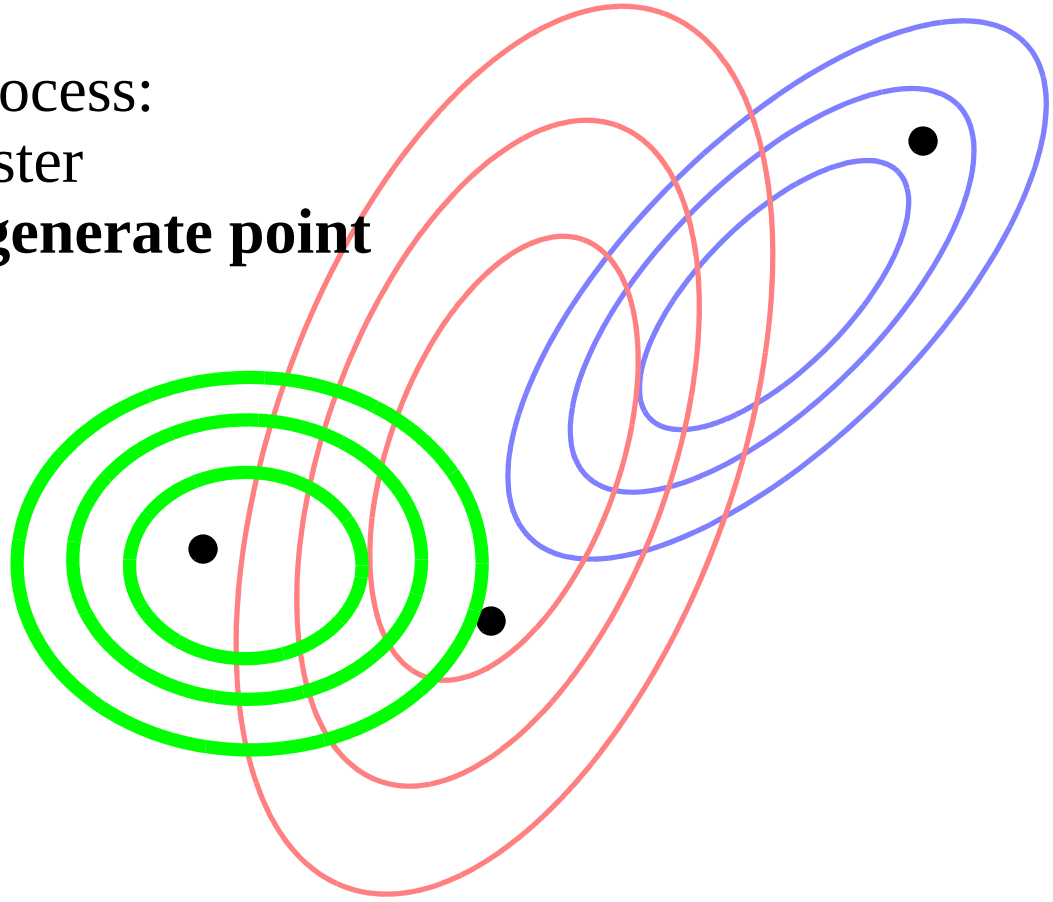
Example One: Bayesian GMM

Generative process:
(1) **Pick a cluster**
(2) Use it to generate point



Example One: Bayesian GMM

Generative process:
(1) Pick a cluster
(2) **Use it to generate point**

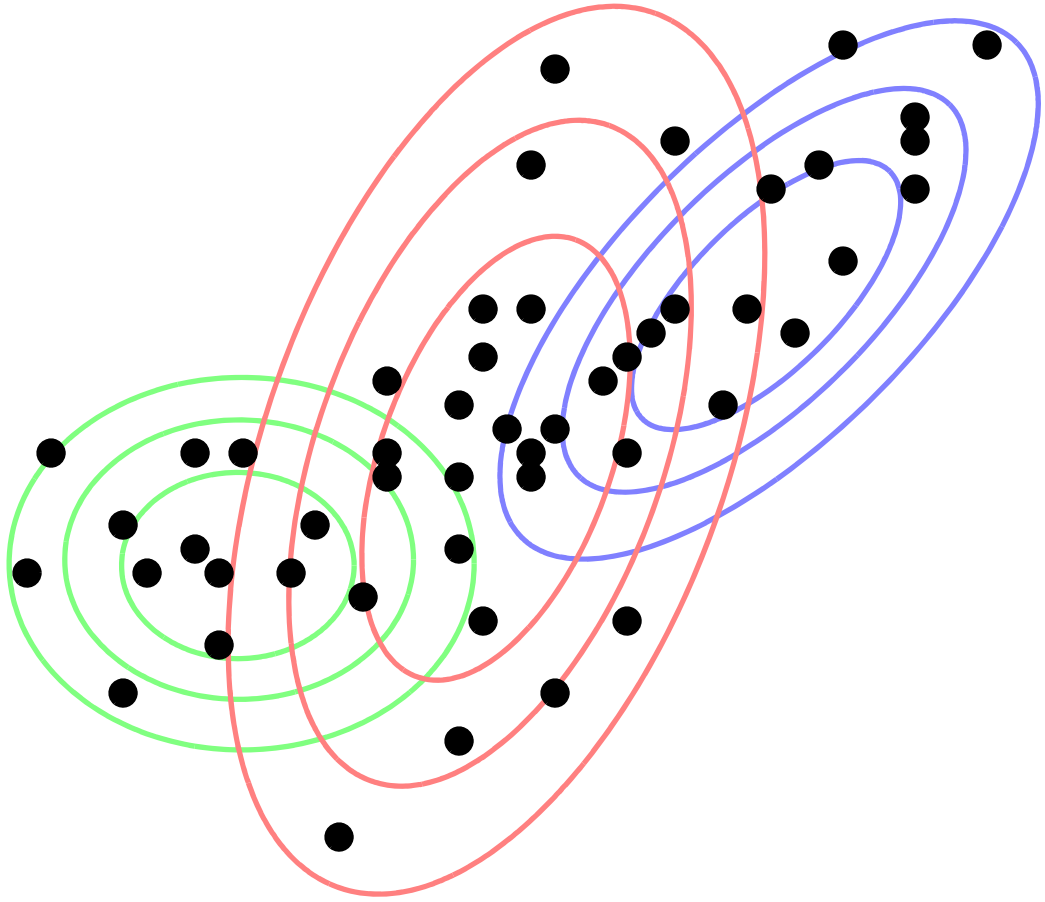


Example One: Bayesian GMM

Then given **this**

Example One: Bayesian GMM

Infer **this**



Example One: Bayesian GMM

- Implemented relevant MCMC simulation on all four platforms
  - — SimSQL, GraphLab, Spark, Giraph

Example One: Bayesian GMM

- Implemented relevant MCMC simulation on all four platforms
  - — SimSQL, GraphLab, Spark, Giraph
- Philosophy: be true to the platform
  - — Ex: avoid “Hadoop abuse” [Smola & Narayanamurthy, VLDB 2010]

Example One: Bayesian GMM

- Implemented relevant MCMC simulation on all four platforms
  - — SimSQL, GraphLab, Spark, Giraph
- Philosophy: be true to the platform
  - — Ex: avoid “Hadoop abuse” [Smola & Narayanamurthy, VLDB 2010]
- Ran on 10 dimensional data, 10 clusters, 10M points per machine
  - — Full (non-diagonal) covariance matrix
  - — Also on 100 dimensional data, 1M points per machine

Example One: Bayesian GMM

GMM: Initial Implementations					
		10 dimensions			100 dimensions
	lines of code	5 machines	20 machines	100 machines	5 machines
SimSQL	197	27:55 (13:55)	28:55 (14:38)	35:54 (18:58)	1:51:12 (36:08)
GraphLab	661	Fail	Fail	Fail	Fail
Spark (Python)	236	26:04 (4:10)	37:34 (2:27)	38:09 (2:00)	47:40 (0:52)
Giraph	2131	25:21 (0:18)	30:26 (0:15)	Fail	Fail

- Some notes:
  - — Times are HH:MM:SS per iteration (time in parens is startup/initialization)
  - — Amount of data is kept constant per machine in all tests
  - — “Fail” means that even with much effort and tuning, it crashed

Example One: Bayesian GMM

GMM: Initial Implementations					
		10 dimensions			100 dimensions
	lines of code	5 machines	20 machines	100 machines	5 machines
SimSQL	197	27:55 (13:55)	28:55 (14:38)	35:54 (18:58)	1:51:12 (36:08)
GraphLab	661	Fail	Fail	Fail	Fail
Spark (Python)	236	26:04 (4:10)	37:34 (2:27)	38:09 (2:00)	47:40 (0:52)
Giraph	2131	25:21 (0:18)	30:26 (0:15)	Fail	Fail

- Not much difference!
  - — But SimSQL was slower in 100 dims. Why?
    - -Experiments didn’t use support for vectors/matrices

Example One: Bayesian GMM

GMM: Initial Implementations					
		10 dimensions			100 dimensions
	lines of code	5 machines	20 machines	100 machines	5 machines
SimSQL	197	27:55 (13:55)	28:55 (14:38)	35:54 (18:58)	1:51:12 (36:08)
GraphLab	661	Fail	Fail	Fail	Fail
Spark (Python)	236	26:04 (4:10)	37:34 (2:27)	38:09 (2:00)	47:40 (0:52)
Giraph	2131	25:21 (0:18)	30:26 (0:15)	Fail	Fail

- Spark is surprisingly slow
  - — Is Spark slower due to Python vs. Java?

GMM: Alternative Implementations					
		10 dimensions			100 dimensions
	lines of code	5 machines	20 machines	100 machines	5 machines
Spark (Java)	737	12:30 (2:01)	12:25 (2:03)	18:11 (2:26)	6:25:04 (36:08)

Example One: Bayesian GMM

GMM: Initial Implementations					
		10 dimensions			100 dimensions
	lines of code	5 machines	20 machines	100 machines	5 machines
SimSQL	197	27:55 (13:55)	28:55 (14:38)	35:54 (18:58)	1:51:12 (36:08)
GraphLab	661	Fail	Fail	Fail	Fail
Spark (Python)	236	26:04 (4:10)	37:34 (2:27)	38:09 (2:00)	47:40 (0:52)
Giraph	2131	25:21 (0:18)	30:26 (0:15)	Fail	Fail

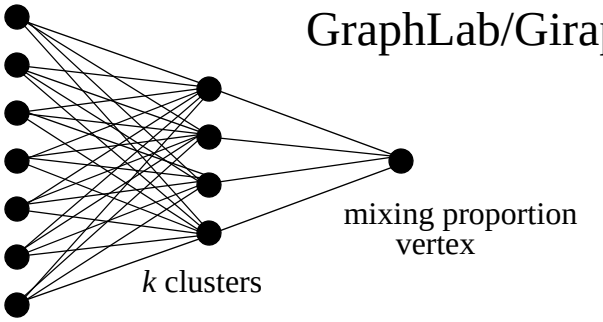
- What about GraphLab?
  - — GraphLab failed every time. Why?

Example One: Bayesian GMM

GMM: Initial Implementations					
		10 dimensions			100 dimensions
	lines of code	5 machines	20 machines	100 machines	5 machines
SimSQL	197	27:55 (13:55)	28:55 (14:38)	35:54 (18:58)	1:51:12 (36:08)
GraphLab	661	Fail	Fail	Fail	Fail
Spark (Python)	236	26:04 (4:10)	37:34 (2:27)	38:09 (2:00)	47:40 (0:52)
Giraph	2131	25:21 (0:18)	30:26 (0:15)	Fail	Fail

- What about GraphLab?
  - — GraphLab failed every time. Why?

GraphLab/Giraph graph model



1 billion data points by
10 clusters by
1KB = 10TB RAM (6TB RAM in 100-machine cluster)

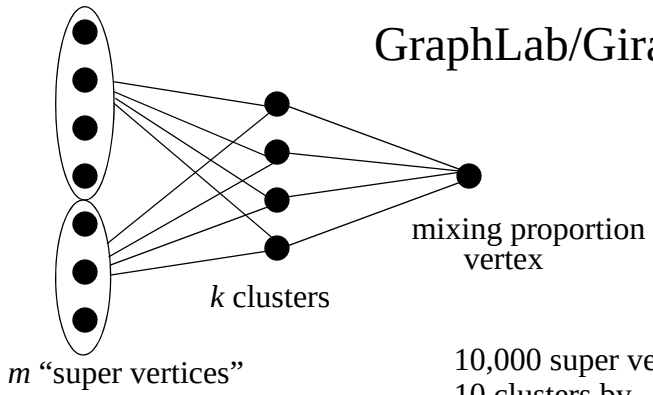
Example One: Bayesian GMM

GMM: Initial Implementations					
		10 dimensions			100 dimensions
	lines of code	5 machines	20 machines	100 machines	5 machines
SimSQL	197	27:55 (13:55)	28:55 (14:38)	35:54 (18:58)	1:51:12 (36:08)
GraphLab	661	Fail	Fail	Fail	Fail
Spark (Python)	236	26:04 (4:10)	37:34 (2:27)	38:09 (2:00)	47:40 (0:52)
Giraph	2131	25:21 (0:18)	30:26 (0:15)	Fail	Fail

- What about GraphLab?
  - — GraphLab failed every time. Why?

To Fix...

GraphLab/Giraph graph model



10,000 super vertices
10 clusters by
1KB = 100 MB RAM (insignificant!)

Example One: Bayesian GMM

GMM: Alternative Implementations					
		10 dimensions			100 dimensions
	lines of code	5 machines	20 machines	100 machines	5 machines
GraphLab (Super Vertex)	681	6:13 (1:13)	4:36 (2:47)	6:09 (1:21)*	33:32 (0:42)

- Super vertex results
  - — GraphLab super vertex screams!

Example One: Bayesian GMM

GMM: Alternative Implementations					
		10 dimensions			100 dimensions
	lines of code	5 machines	20 machines	100 machines	5 machines
GraphLab (Super Vertex)	681	6:13 (1:13)	4:36 (2:47)	6:09 (1:21)*	33:32 (0:42)

- Super vertex results
  - — GraphLab super vertex screams!
  - — But to be fair, others can benefit from super vertices as well...

.

GMM: Super Vertex Implementations				
	10 dimensions, 5 machines		100 dimensions, 5 machines	
	w/o super vertex	with super vertex	w/o super vertex	with super vertex
SimSQL	27:55 (13:55)	6:20 (12:33)	1:51:12 (36:08)	7:22 (14:07)
GraphLab	Fail	6:13 (1:13)	Fail	33:32 (0:42)
Spark (Python)	26:04 (4:10)	29:12 (4:01)	47:40 (0:52)	47:03 (2:17)
Giraph	25:21 (0:18)	13:48 (0:03)	Fail	6:17:32 (0:03)

Questions?