Large Scale Machine Learning with the SimSQL System

Chris Jermaine
Rice University

Many Current and Past Rice Team Members
Also, Peter J. Haas at IBM Almaden

This Talk Is About

- Programming environments/execution platforms for big ML

First, an Admission!!!

- I’m a database guy™

First, an Admission…

- I’m a database guy™
- What does that mean?

First, an Admission…

- I’m a database guy™
- What does that mean?
- To me, means that I worship at the church of **data independence**
  - — Now what in the heck does ***that*** mean?

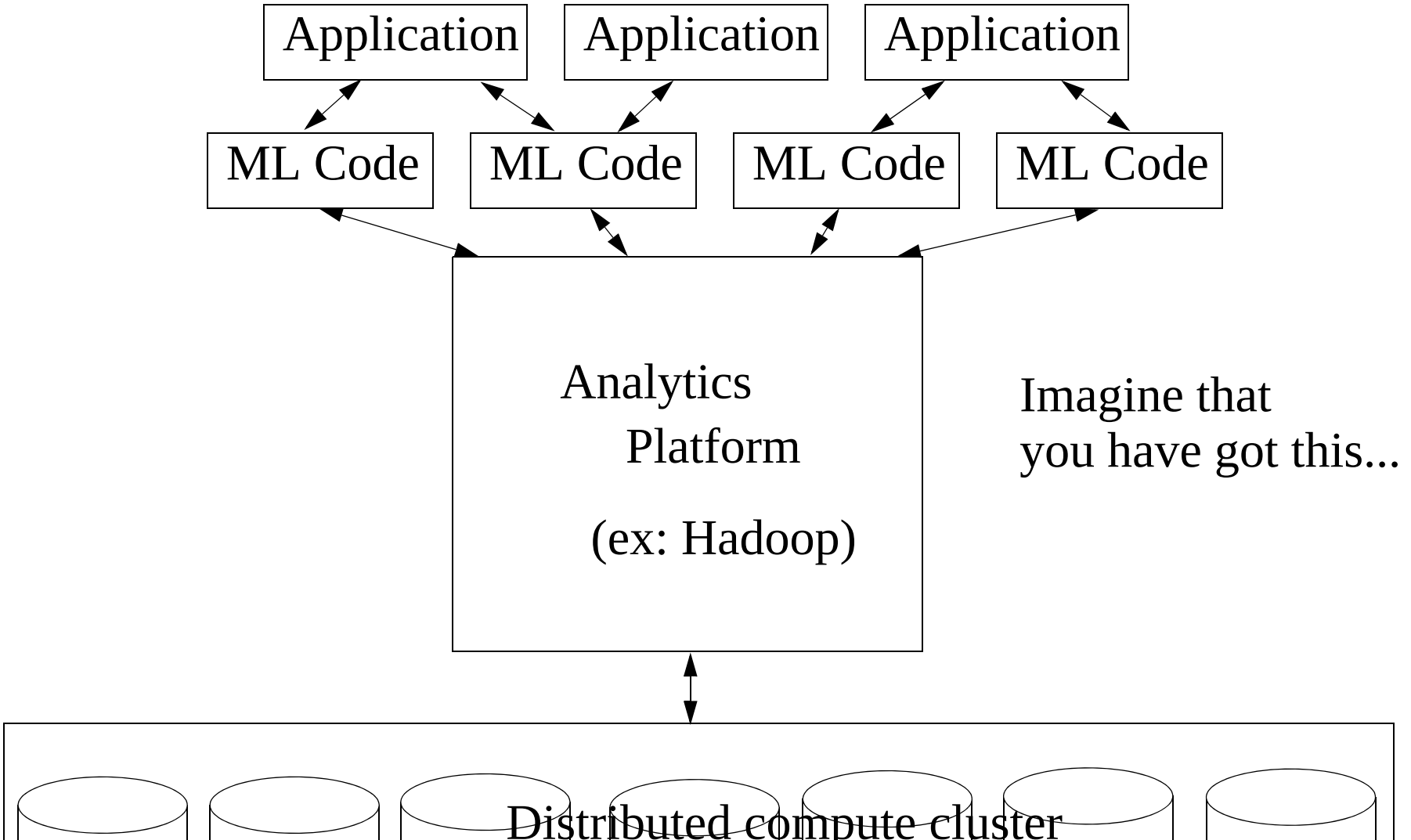
First, an Admission...

- I'm a database guy™
- What does that mean?
- To me, means that I worship at the church of **data independence**
  - — Now what in the heck does ***that*** mean?
- Means that when one designs a data-processing system...
- It should strive for the following ideal:
  - — Coder specifies **what** the computation result should be, not **how** to get there
  - — System itself figures out the **how** (the "declarative" paradigm)
  - — Means code can be independent of data format, size, schema, processing hardware
  - — Same code runs on one box with a GPU and on a 1000-machine cluster

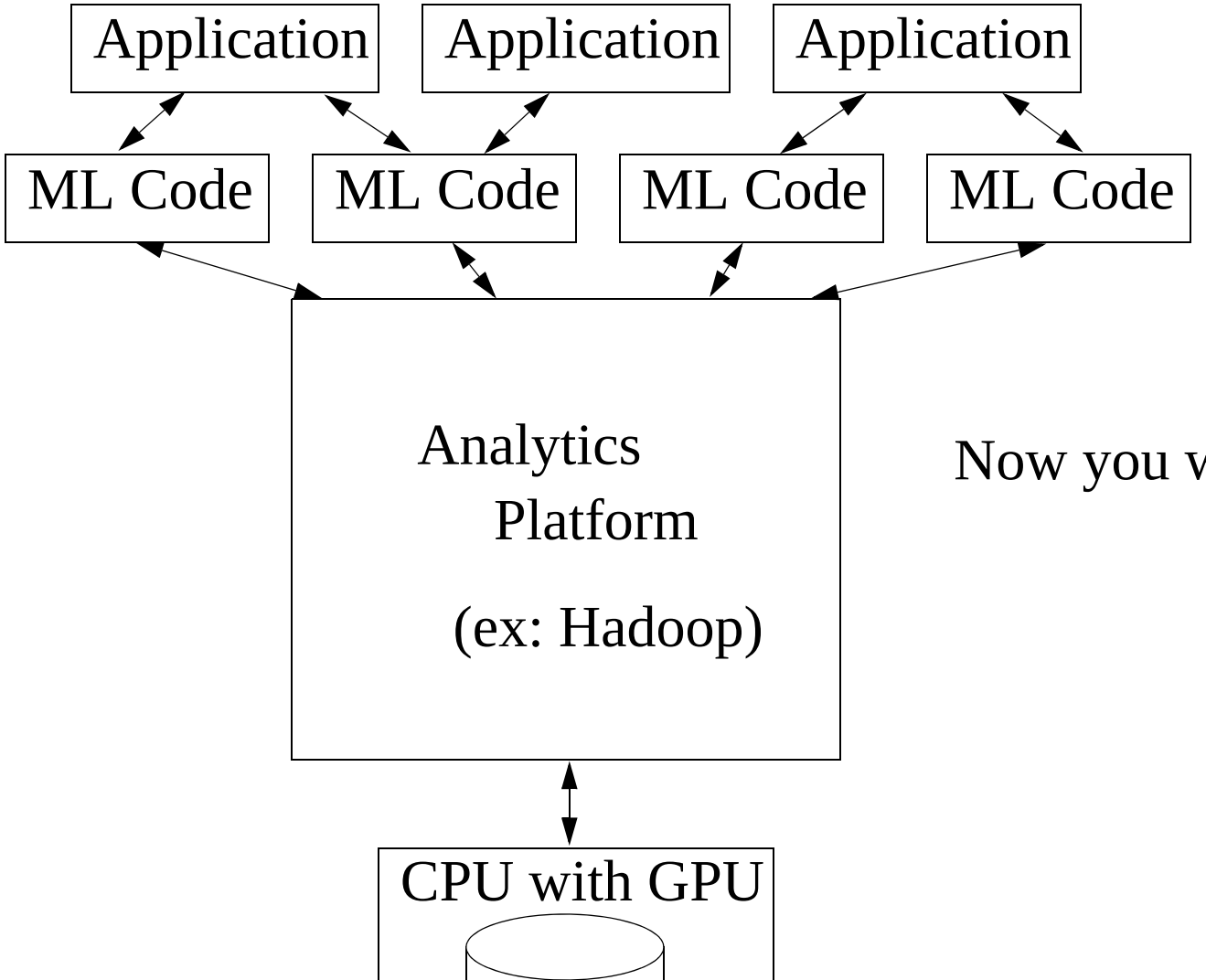
Why Are Declarative and Data That Good?

Declarative Arg If: One Code, Many Backends

Declarative Arg 1: One Code, Many Backends

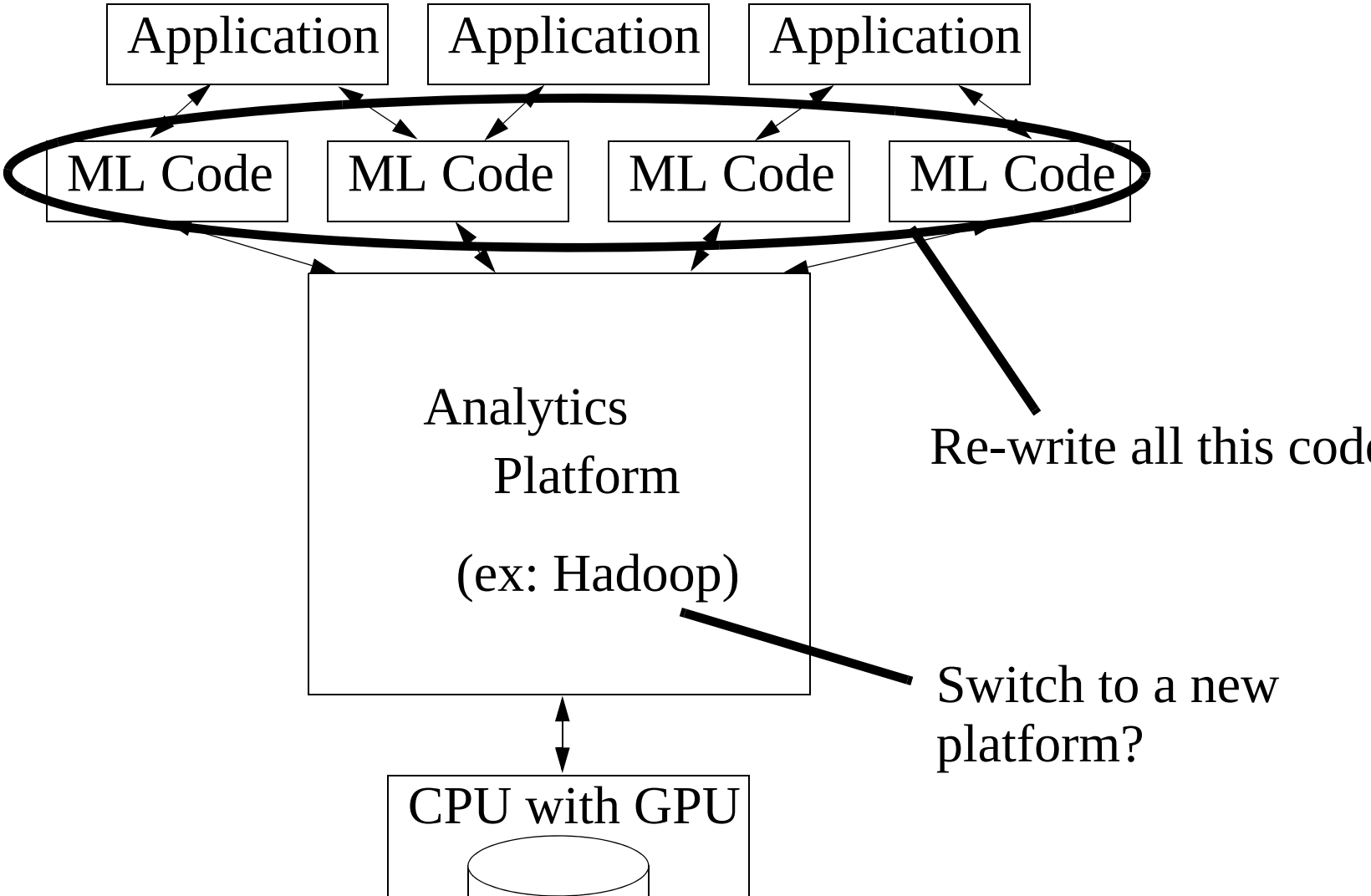


Declarative Arg 1: One Code, Many Backends

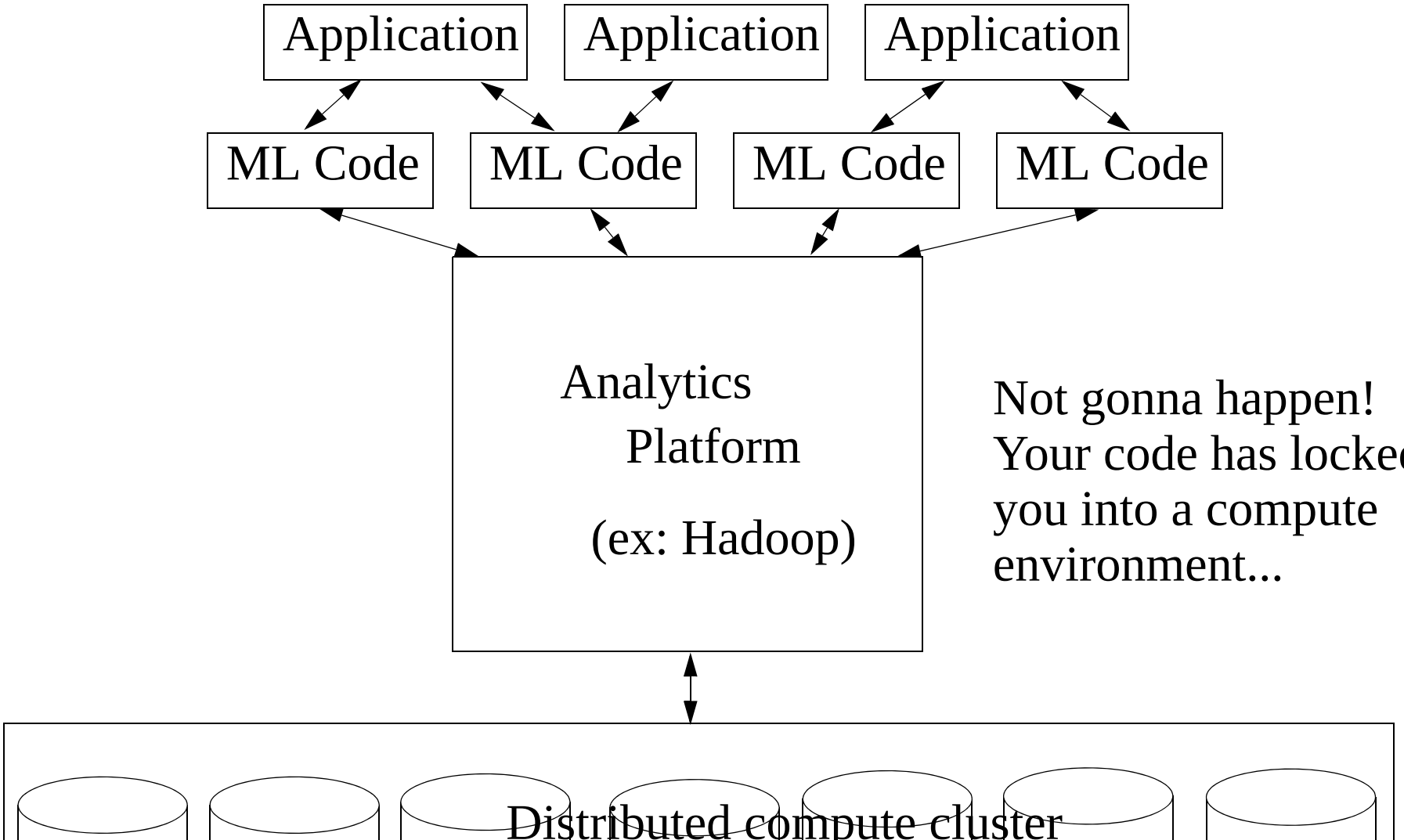


Now you want this...

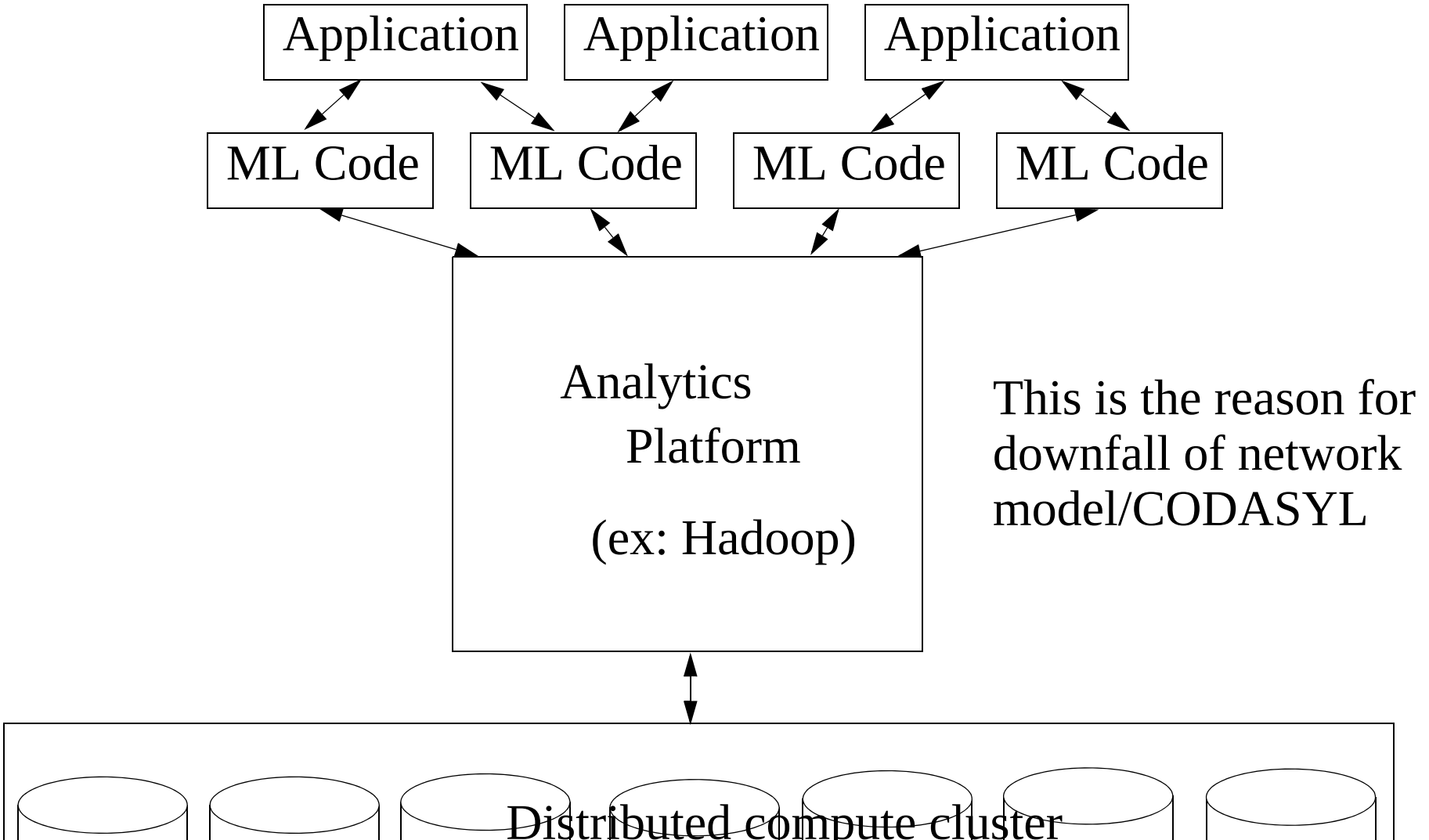
Declarative Arg: One Code, Many Backends



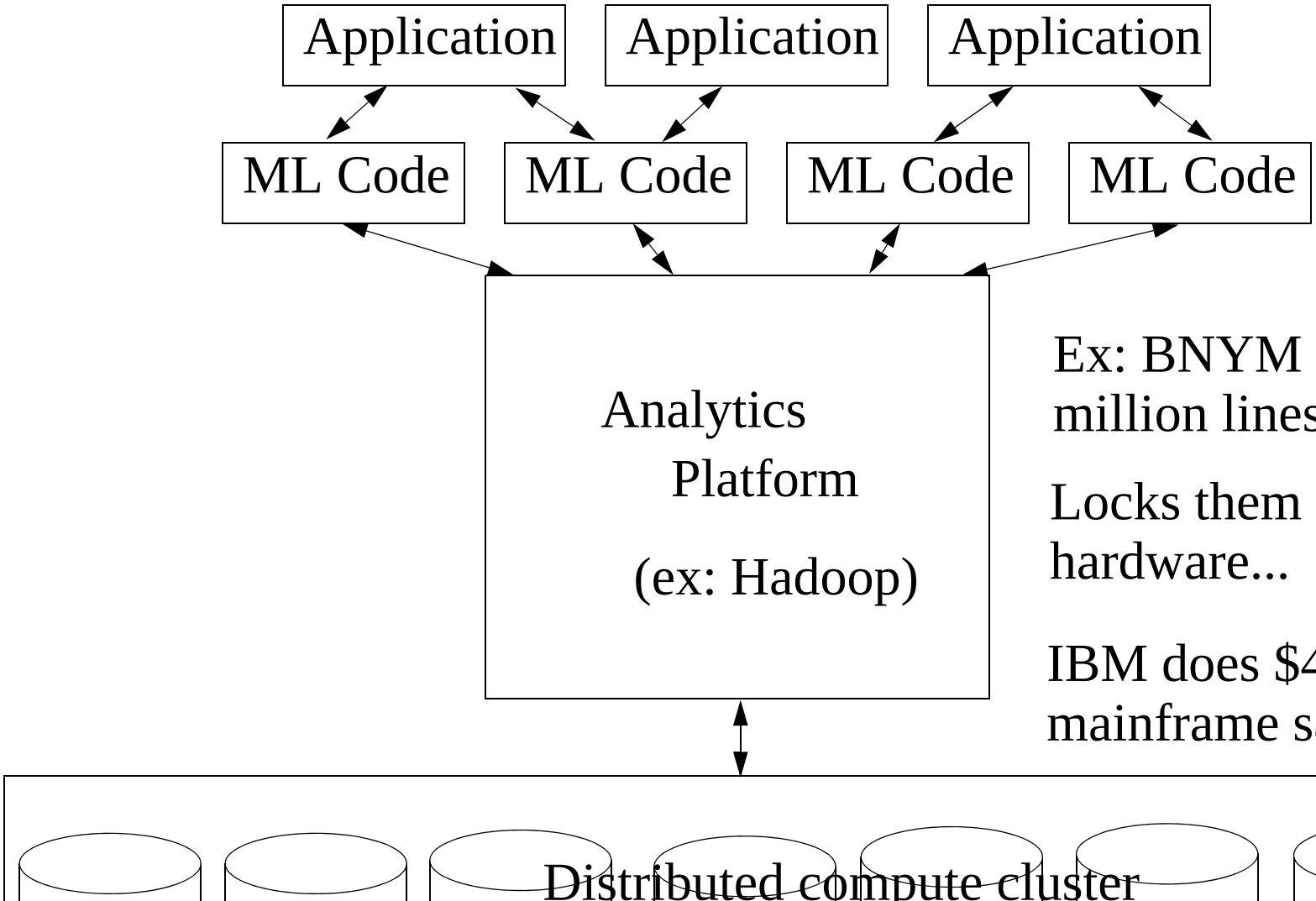
Declarative Arg 1: One Code, Many Backends



Declarative Arg 1: One Code, Many Backends



Declarative Arg 1: One Code, Many Backends



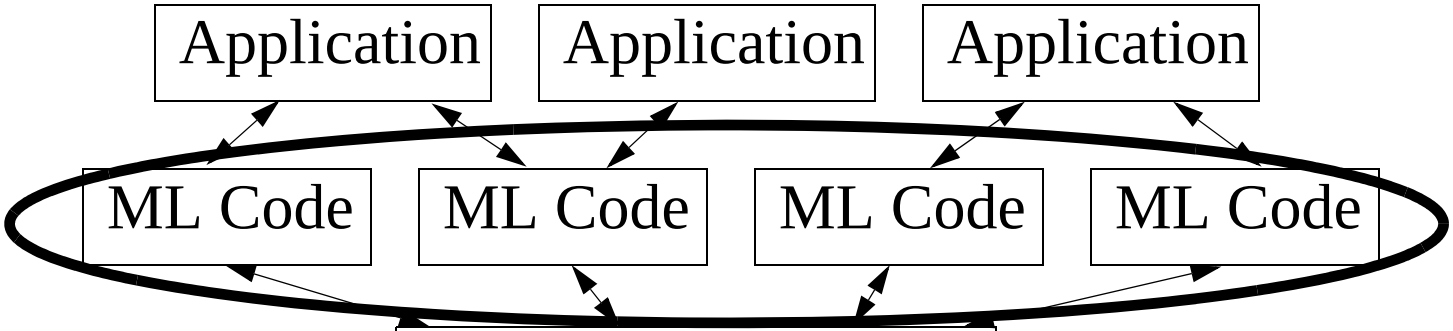
Ex: BNYM runs 343 million lines of COBOL

Locks them into 60's hardware...

IBM does $4B plus in mainframe sales!

Declarative Arg 21 Pseudo Python Algorithms

Declarative Arg 2: Freedom from Algorithms



Analytics Platform

(ex: Hadoop)

Imagine many codes have

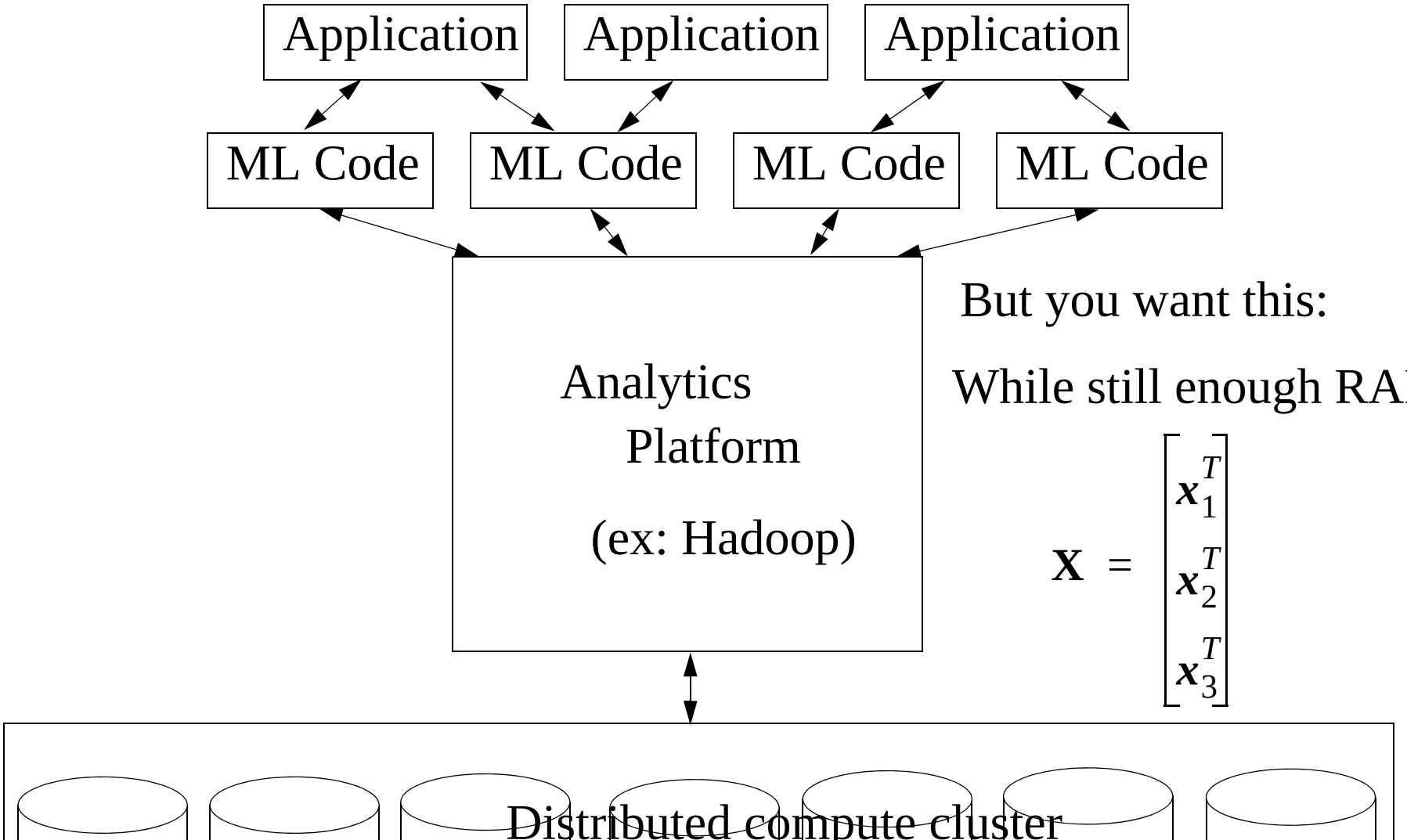
Map:

$$x \rightarrow xx^T$$

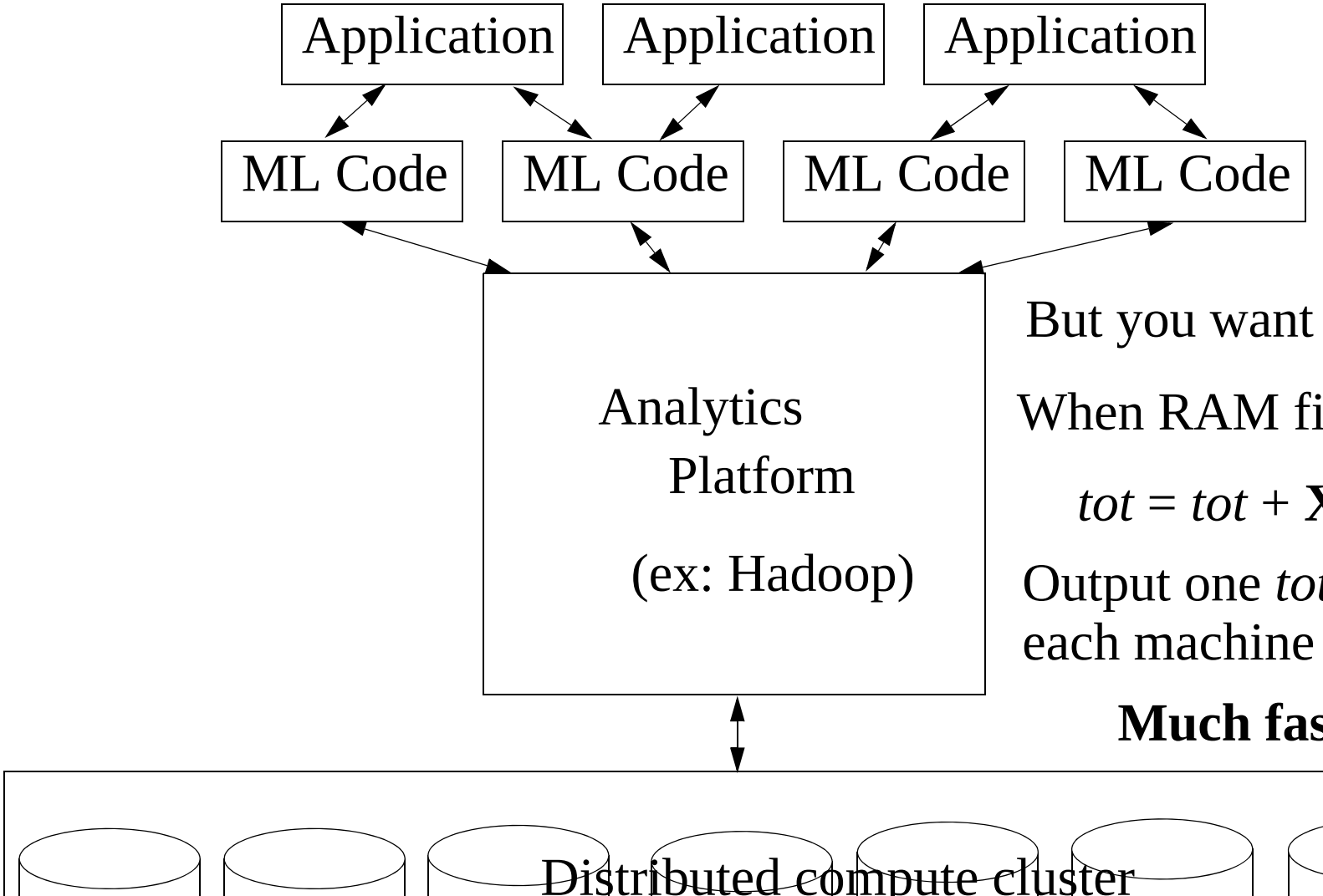
Reduce:

$$\sum xx^T$$

Declarative Alg. Freedom from Algorithms



Declarative Alg 2: Hadoop Mahout Algorithms



But you want this:

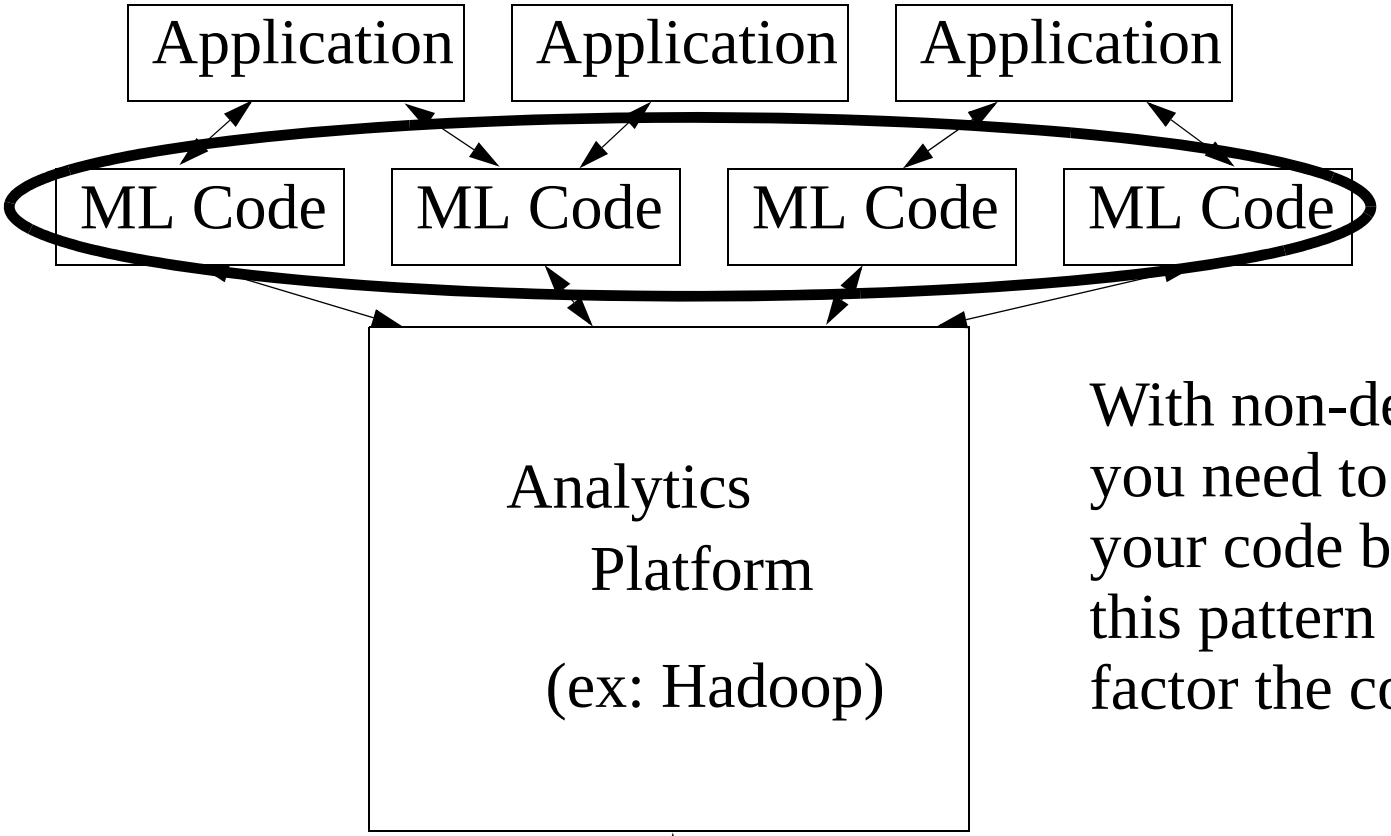
When RAM fills,

$$tot = tot + \mathbf{X}\mathbf{X}^T$$

Output one *tot* from each machine and sum

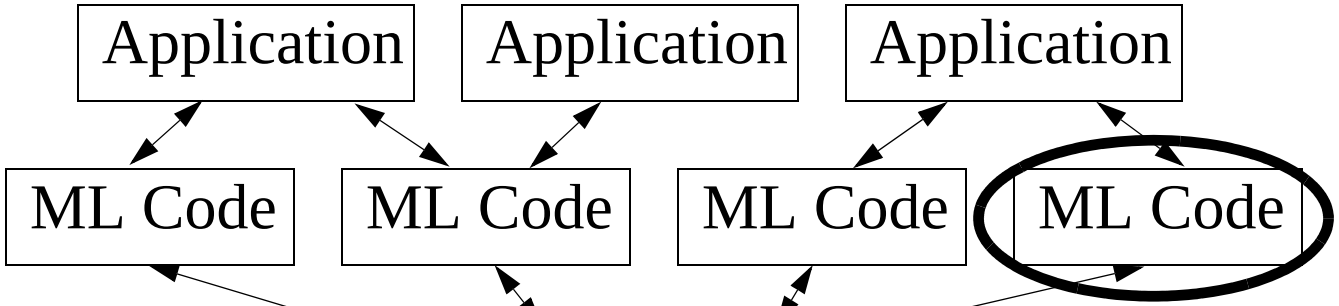
Much faster!

Declarative Algorithms



With non-declarative you need to search your code base for this pattern and re-factor the code!

Declarative Alg 2: Freedom from Algorithms



Analytics Platform

(ex: Hadoop)

Not gonna happen...

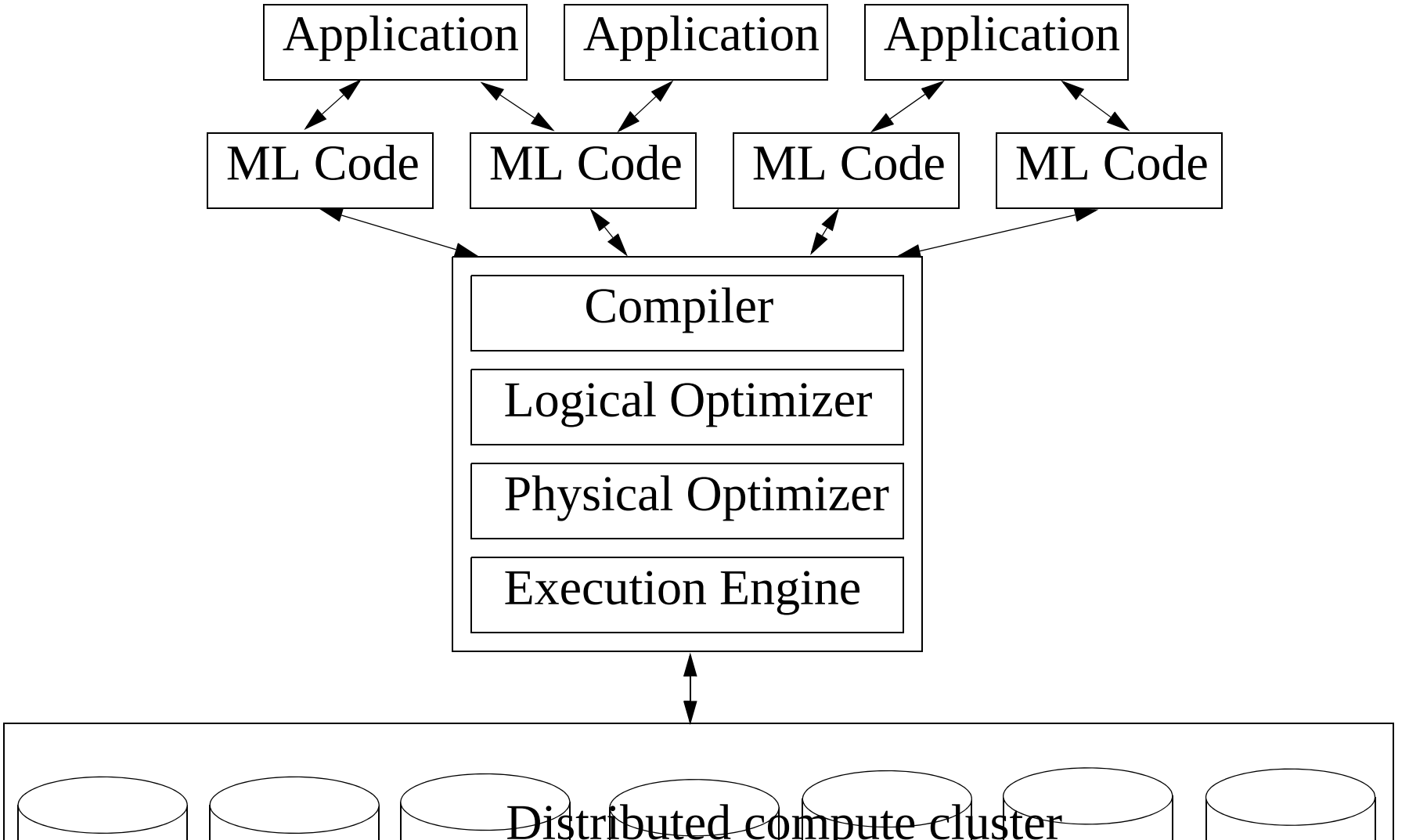
Instead, an engineer own
a code (eg NNMF)

Tinkers and improves it
wo concern for other coc

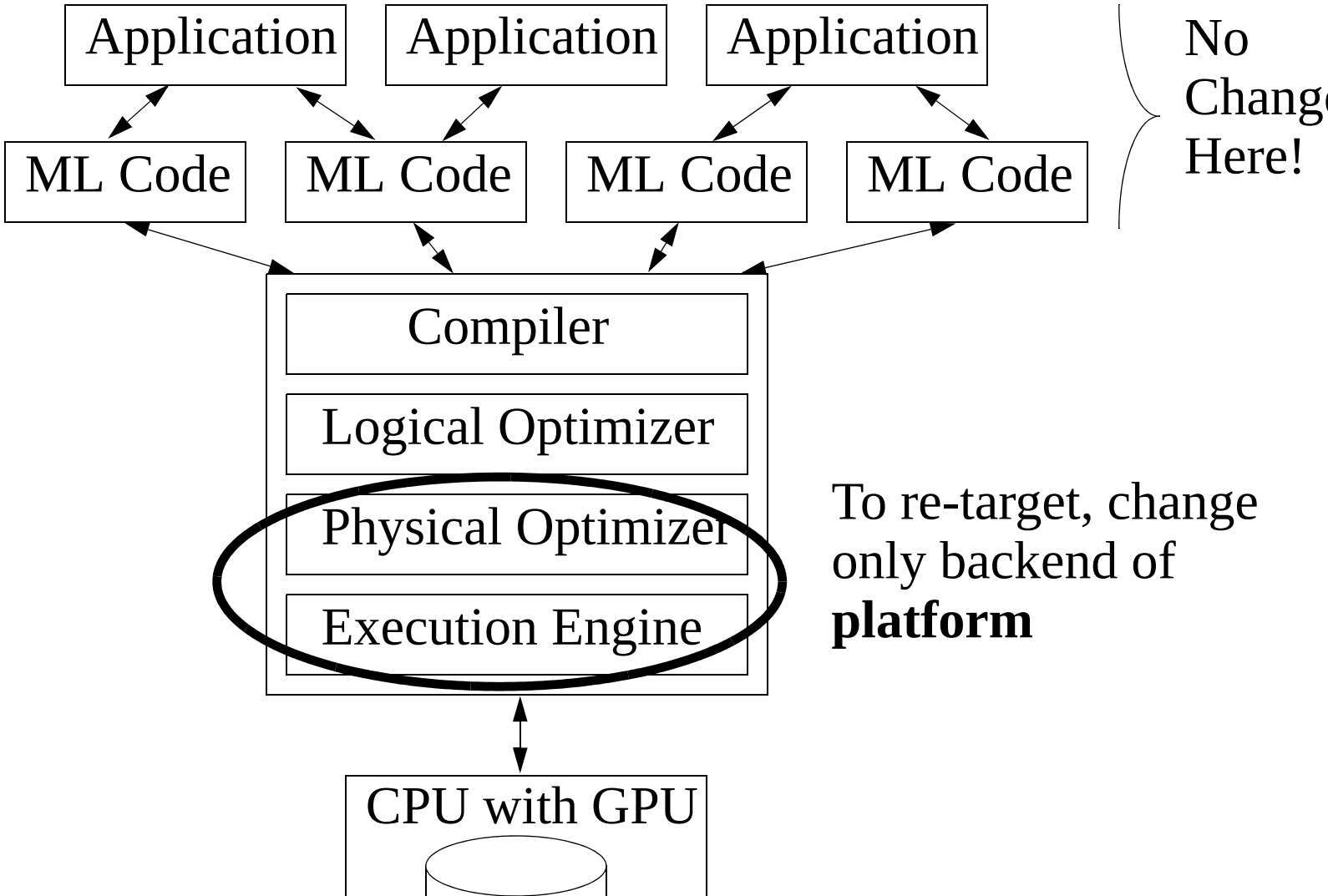
Every man for himself!

Much Better In System Based On Data That

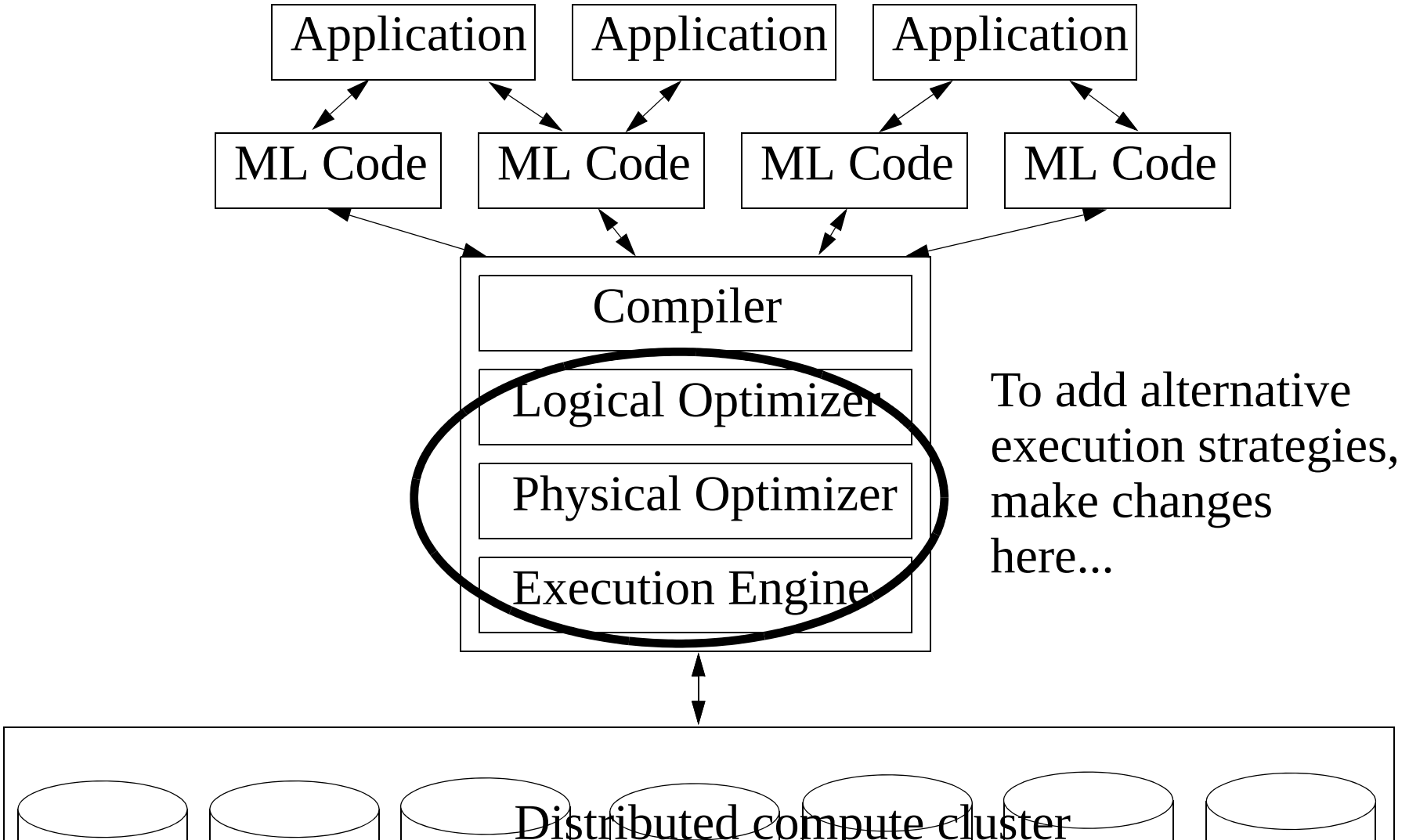
Much Better If System Based On Data Flow



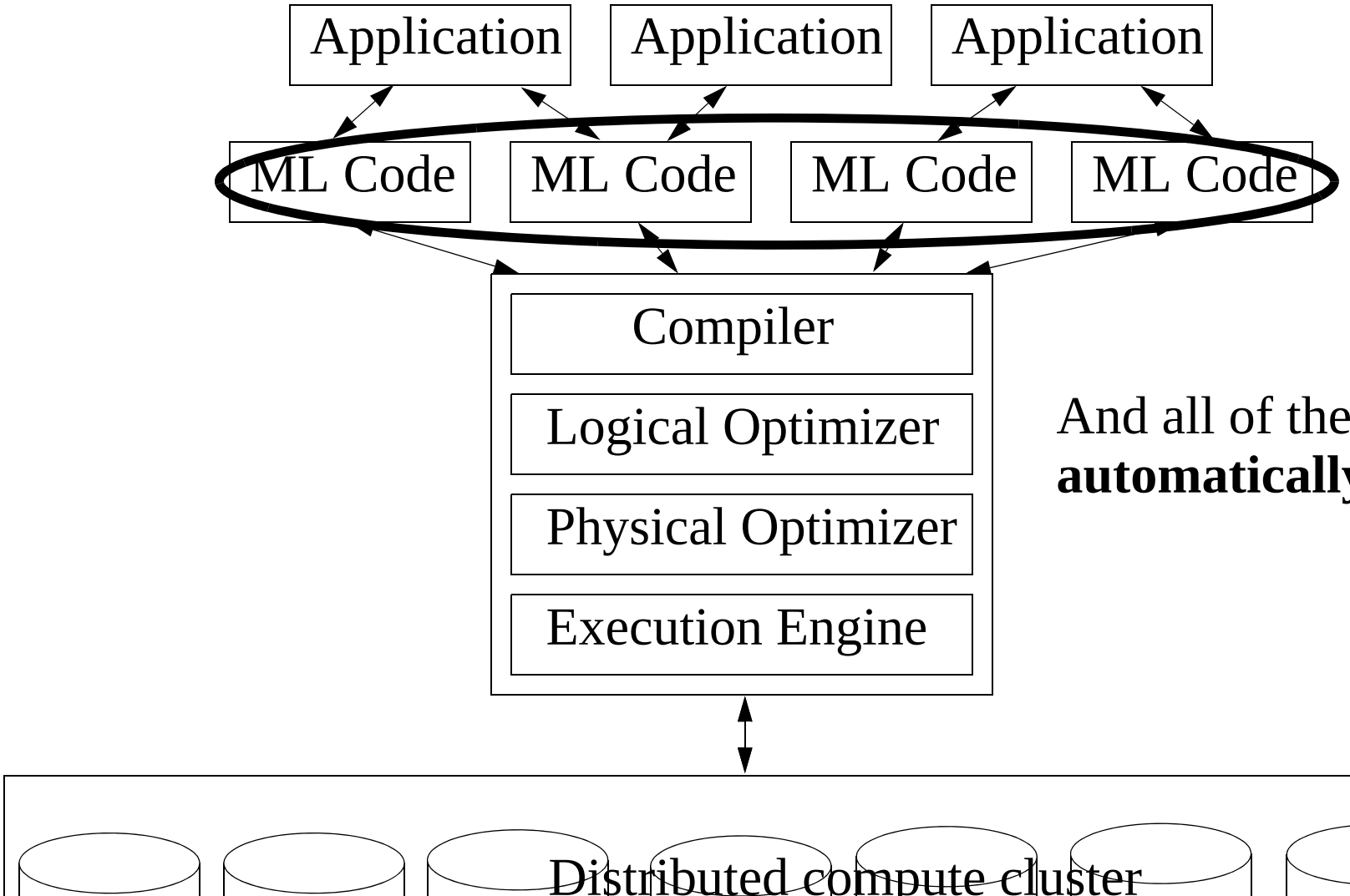
Much Better If System Based On Data Flow



Much Better ML System Based On Data Flow



Much Better If System Based On Data Flow



And all of these codes **automatically** benefit

Huge win

- Once written, code at top of stack can remain unchanged
  - — #1 selling point of RDBMS tech for 30 years!

Huge win

- Once written, code at top of stack can remain unchanged
  - — #1 selling point of RDBMS tech for 30 years!
- Unfortunately, not accepted by ML community
  - — ML people write great ML-on-GPU papers
  - — They design platforms such as GraphLab (higher-level, MPI-like framework)

Huge win

- Once written, code at top of stack can remain unchanged
  - — #1 selling point of RDBMS tech for 30 years!
- Unfortunately, not accepted by ML community
  - — ML people write great ML-on-GPU papers
  - — They design platforms such as GraphLab (higher-level, MPI-like framework)
- Some in DB community have looked at declarative dataflow...
  - — Spark SQL on Spark
  - — Meteor on Stratosphere
  - — Asterix from UC Irvine

Huge Win

- Once written, code at top of stack can remain unchanged
  - — #1 selling point of RDBMS tech for 30 years!
- Unfortunately, not accepted by ML community
  - — ML people write great ML-on-GPU papers
  - — They design platforms such as GraphLab (higher-level, MPI-like framework)
- Some in DB community have looked at declarative dataflow...
  - — Spark SQL on Spark
  - — Meteor on Stratosphere
  - — Asterix from UC Irvine
- But this is far from declarative ML
  - — The codes don't look anything like math!

The Goal

- Start with mathematical spec of learning algorithm...

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

This is math for the Bayesian Lasso, lifted from original paper

— Bayesian regression model with regularizing prior on regression coefs

The Goal

- Programmer writes 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);
```

We call our language “BUDS”

The Goal

- 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);
```

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

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

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

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

The Goal

- And the system compiles and executes this for a huge data set
  - — On hundreds or thousands of machines...
  - — Or on a desktop with a GPU...
  - — Or for whatever backend the system can target...

Also important

- We don’t want to be like everyone and argue for a new DA stack
  - The world has too many dataflow platforms already

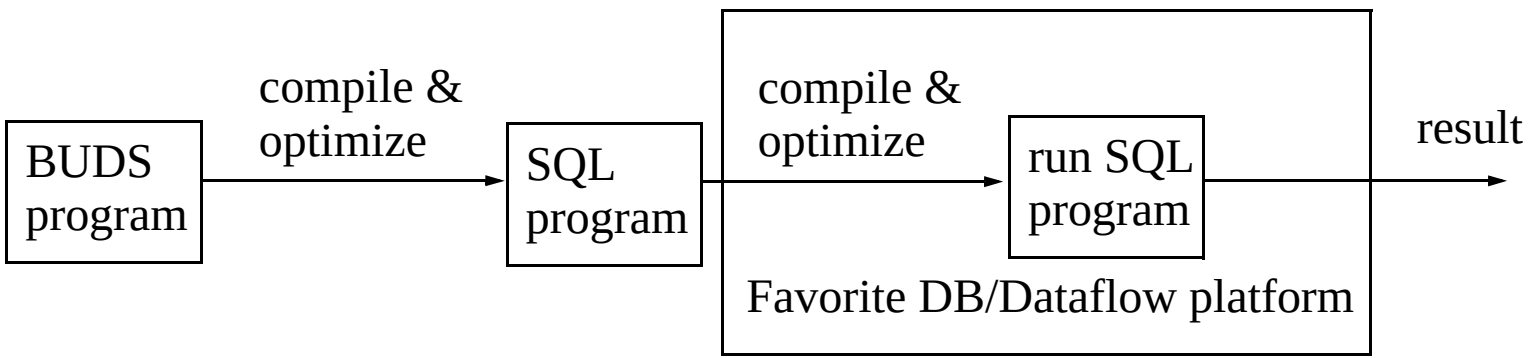
The Fact is

- SQL or SQL-like declarative language is the LCD for all systems
  - — Including commercial databases
  - — Free databases
  - — And newfangled dataflow platforms

The Fact is

- SQL or SQL-like declarative language is the LCD for all systems
  - — Including commercial databases
  - — Free databases
  - — And newfangled dataflow platforms

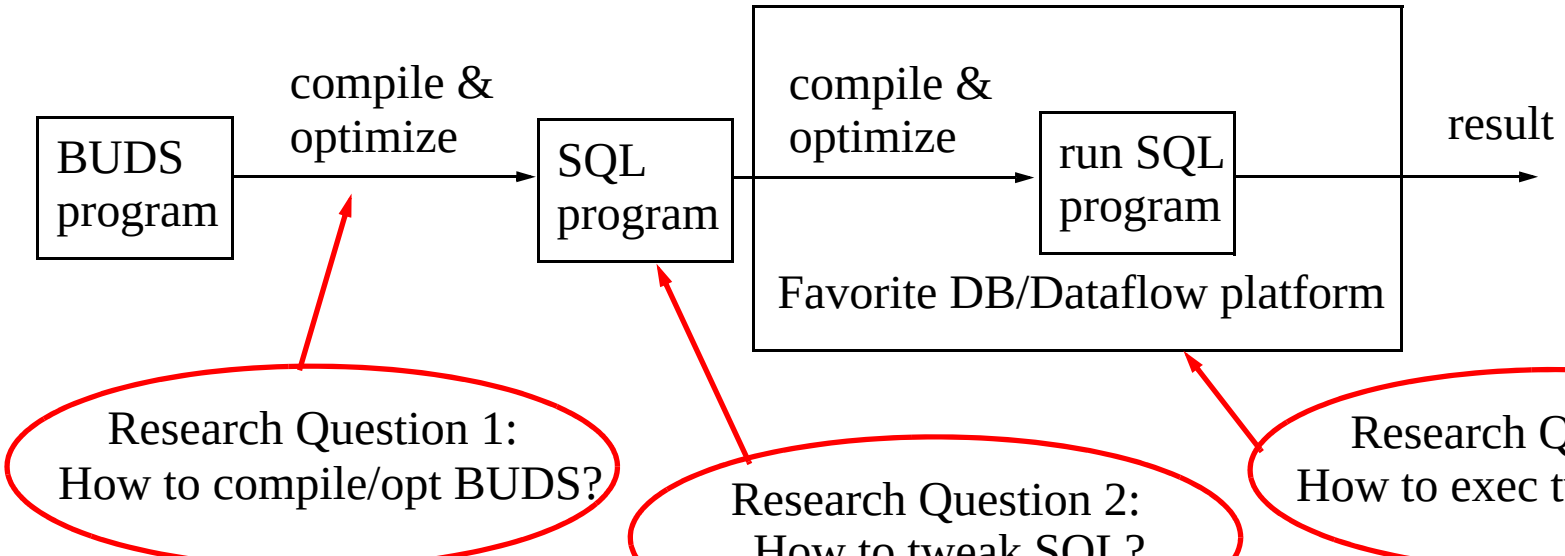
So here's the workflow we envision...



The Fact is

- SQL or SQL-like declarative language is the LCD for all systems
  - — Including commercial databases
  - — Free databases
  - — And newfangled dataflow platforms

So here's the workflow we envision...



Questions?