Automatic Partitioning of Database Applications

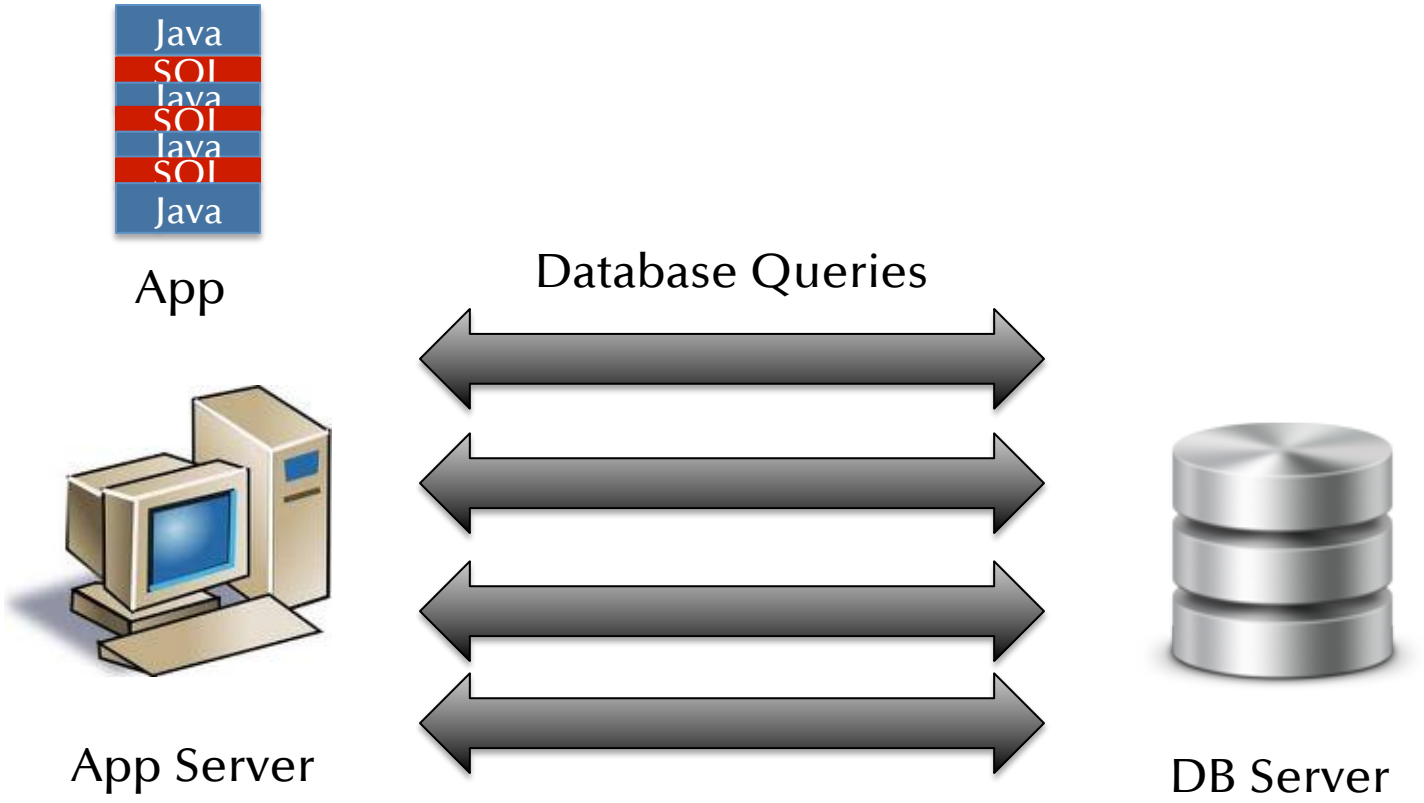
Alvin Cheung
Samuel Madden
MIT

Owen Arden
Andrew C. Myers
Cornell

Writing Efficient DB Apps is Difficult!

- Database applications are everywhere
  - Basically all web apps
- Why is writing efficient database applications so difficult?
  - Let's first review a typical architecture

Architecture of DB Applications



Running Example

```
discount = executeQuery("select discount from customers
                         where id = " + cid);

totalAmount = orderTotal * (1 - discount);

credit = executeQuery("select credit from customers
                       where id = " + cid);

if (credit < totalAmount)
   printToConsole("Only " + credit + " in account!");
else
   executeUpdate("update customer set credit = " +
                 (credit - totalAmount) + " where id = "  + cid);
```

Speeding up DB Apps: Take 1

DB
```
discount = executeQuery("select discount from customers
                         where id = " + cid);
```

APP
```
totalAmount = orderTotal * (1 - discount);
```

DB
```
credit = executeQuery("select credit from customers
                       where id = " + cid);
```

APP
```
if (credit < totalAmount)
   printToConsole("Only " + credit + " in account!");
else
```

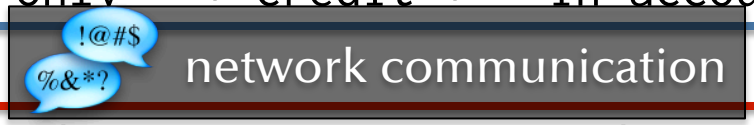
DB
```
   executeUpdate("update customer set credit = " +
                 (credit - totalAmount) + " where id = "  + cid);
```

Speeding up DB Apps: Take 1

DB
```
discount = executeQuery("select discount from customers
                         where id = " + cid);
```


APP
```
totalAmount = orderTotal * (1 - discount);
```


DB
```
credit = executeQuery("select credit from customers
                       where id = " + cid);
```


APP
```
if (credit < totalAmount)
   printToConsole("Only " + credit + " in account!");
else
```


DB
```
   executeUpdate("update customer set credit = " +
                 (credit - totalAmount) + " where id = "  + cid);
```

Speeding up DB Apps: Take 2

DB

```
discount = executeQuery("select discount from customers
                         where id = " + cid);

totalAmount = orderTotal * (1 - discount);

credit = executeQuery("select credit from customers
                       where id = " + cid);

if (credit < totalAmount)
```

APP

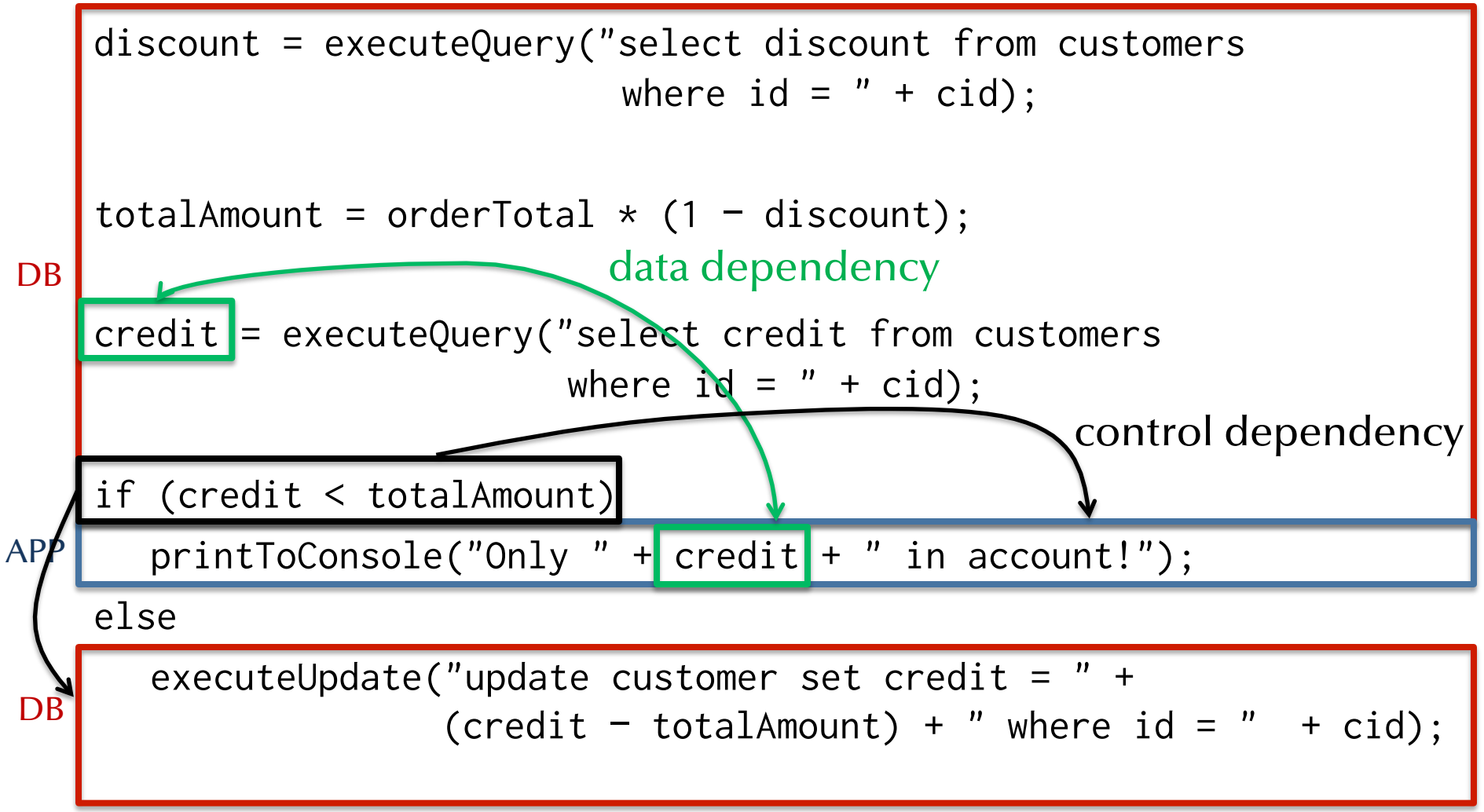
```
   printToConsole("Only " + credit + " in account!");
```

```
else
```

DB

```
   executeUpdate("update customer set credit = " +
                 (credit - totalAmount) + " where id = "  + cid);
```

Speeding up DB Apps: Take 2



Speeding up DB Apps: Take 3

```
discount = executeQuery("select discount from customers
                         where id = " + cid);

totalAmount = orderTotal * (1 - discount);
```

data dependency

DB

```
credit = executeQuery("select credit from customers
                       where id = " + cid);
```

DB Server

control dependency

```
if (credit < totalAmount)
```

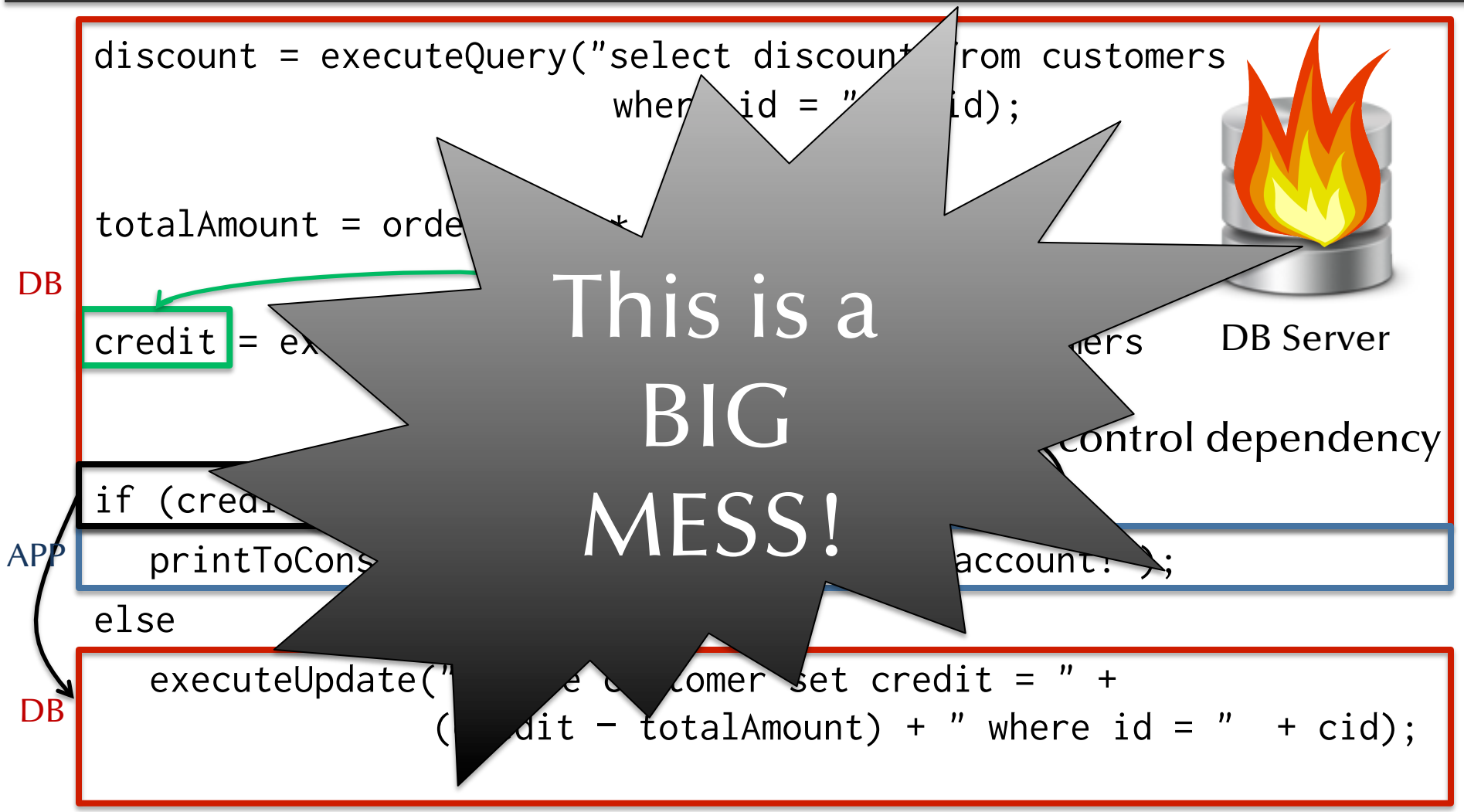
APP

```
   printToConsole("Only " + credit + " in account!");
else
```

DB

```
   executeUpdate("update customer set credit = " +
                 (credit - totalAmount) + " where id = "  + cid);
```

Speeding up DB Apps: Take 3

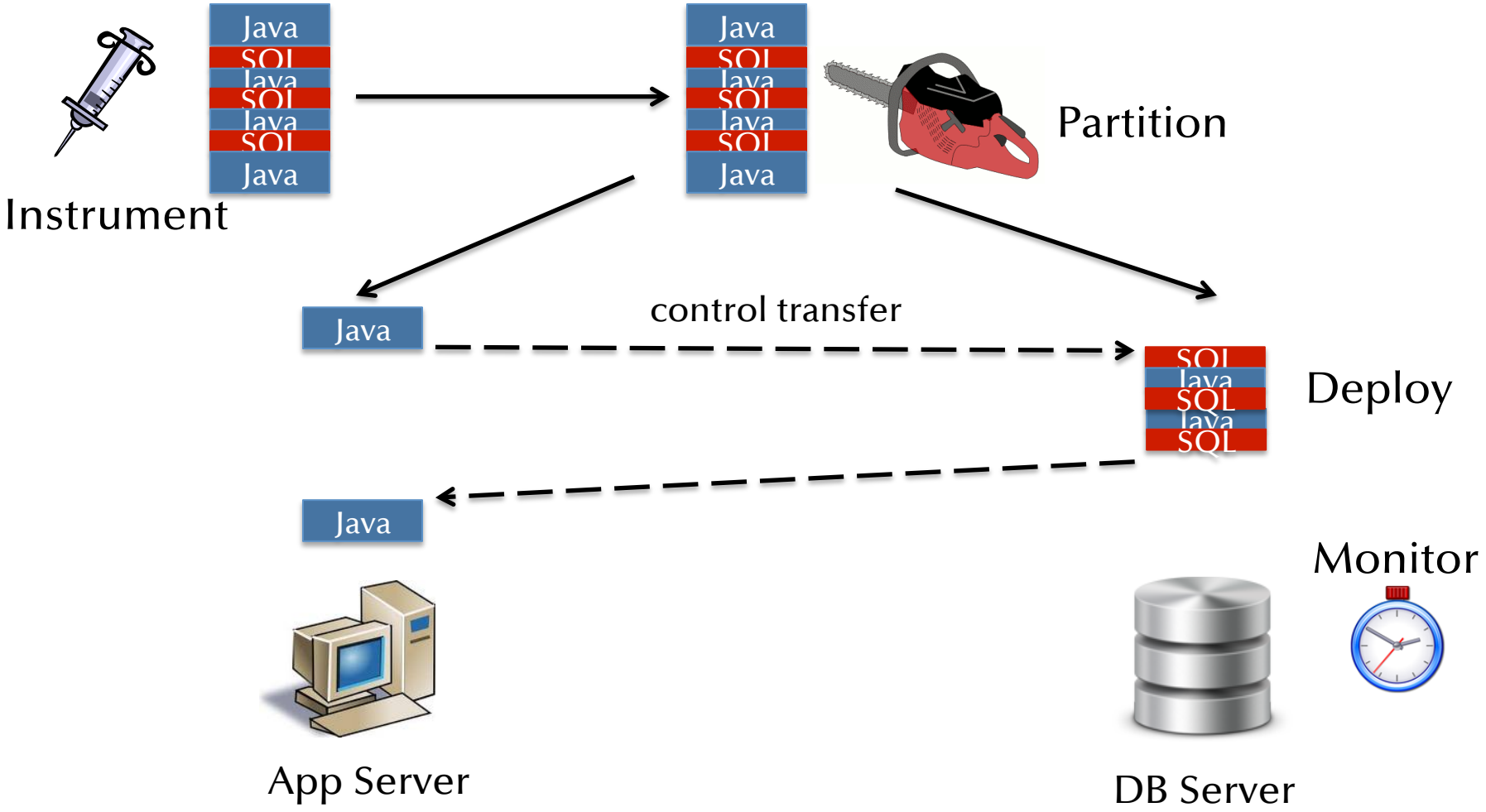


Introducing Pyxis

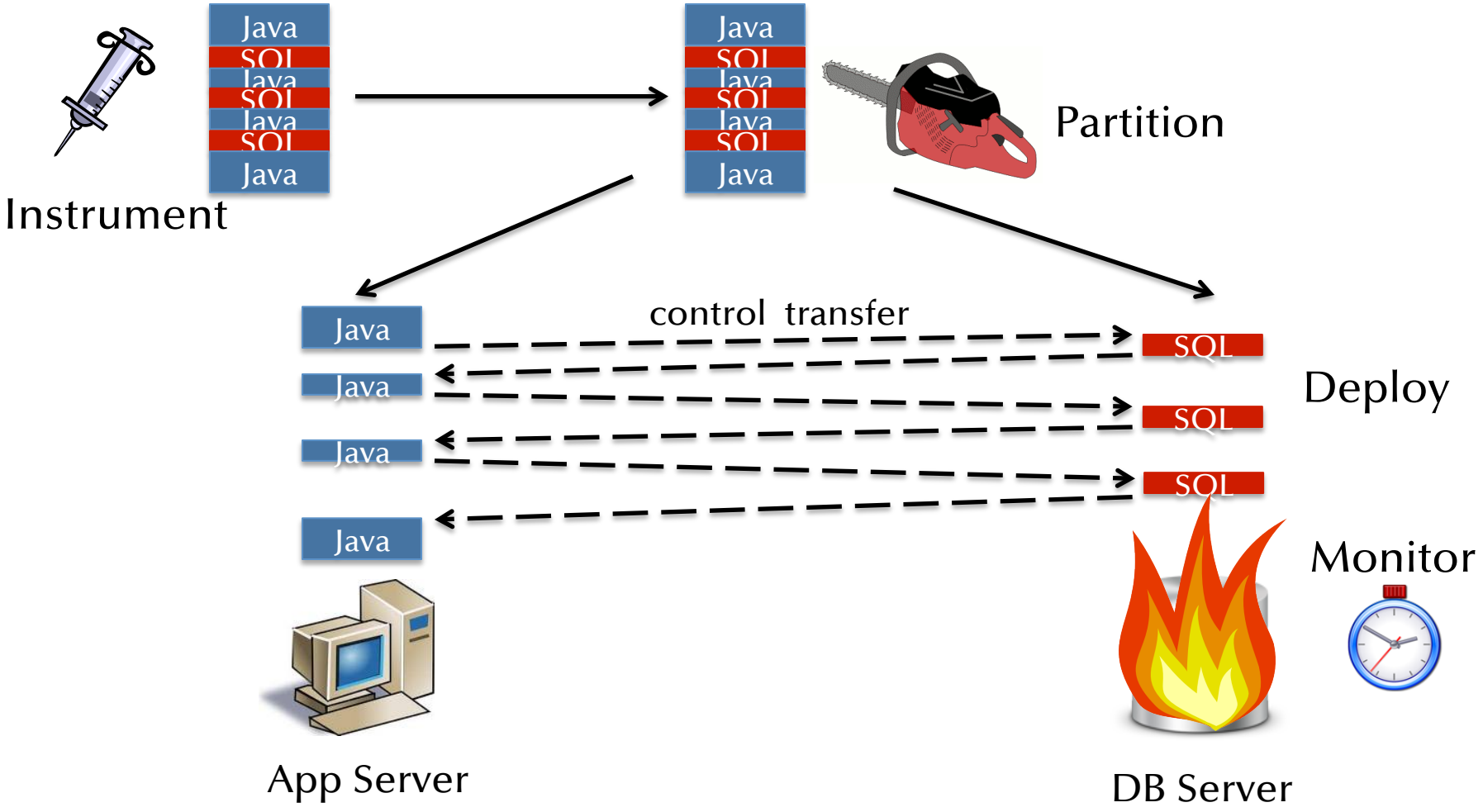
- "Store-procedurizes" DB apps and pushes computation to the DB
- Adaptively controls the amount of computation pushed to DB for optimal performance
- No programmer intervention required

Using Pyxis

How Pyxis Works



How Pyxis Works



Source Code Partitioning

Application Profiling

- Automatically instrument source code to count the number of times each statement was executed for a short period of time
- Measure capabilities of the application and DB servers

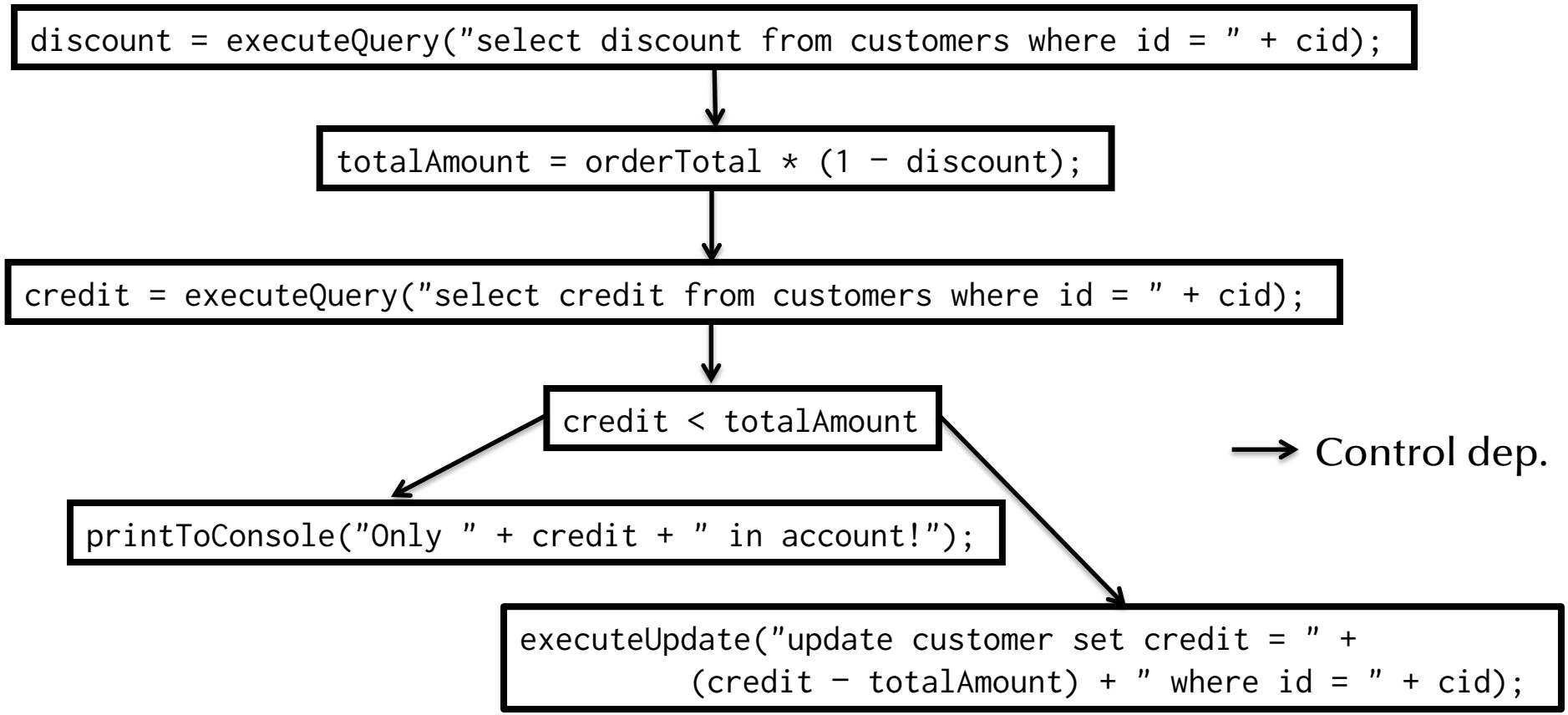
Counts

```
100   discount = executeQuery("select discount from customers
                                where id = " + cid);
100   totalAmount = orderTotal * (1 - discount);
100   credit = executeQuery("select credit from customers
                              where id = " + cid);

100   if (credit < totalAmount)
 25     printToConsole("Only " + credit + " in account!");
      else
 75     executeUpdate("update customer set credit = " +
                      (credit - totalAmount) + " where id = " +
                       cid);
```

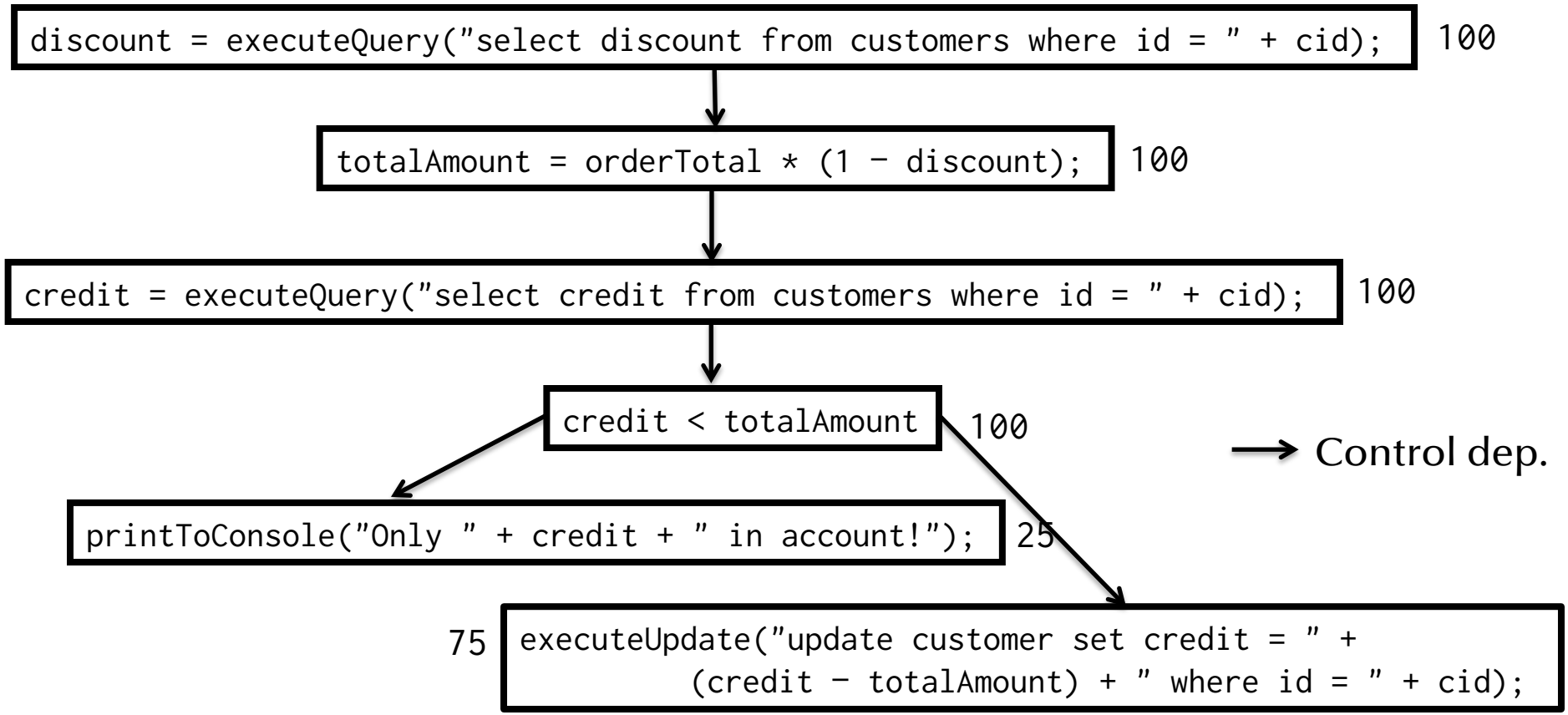
Control Dependencies

- Create program dependence graph that describes control flow and links up variable definitions and uses



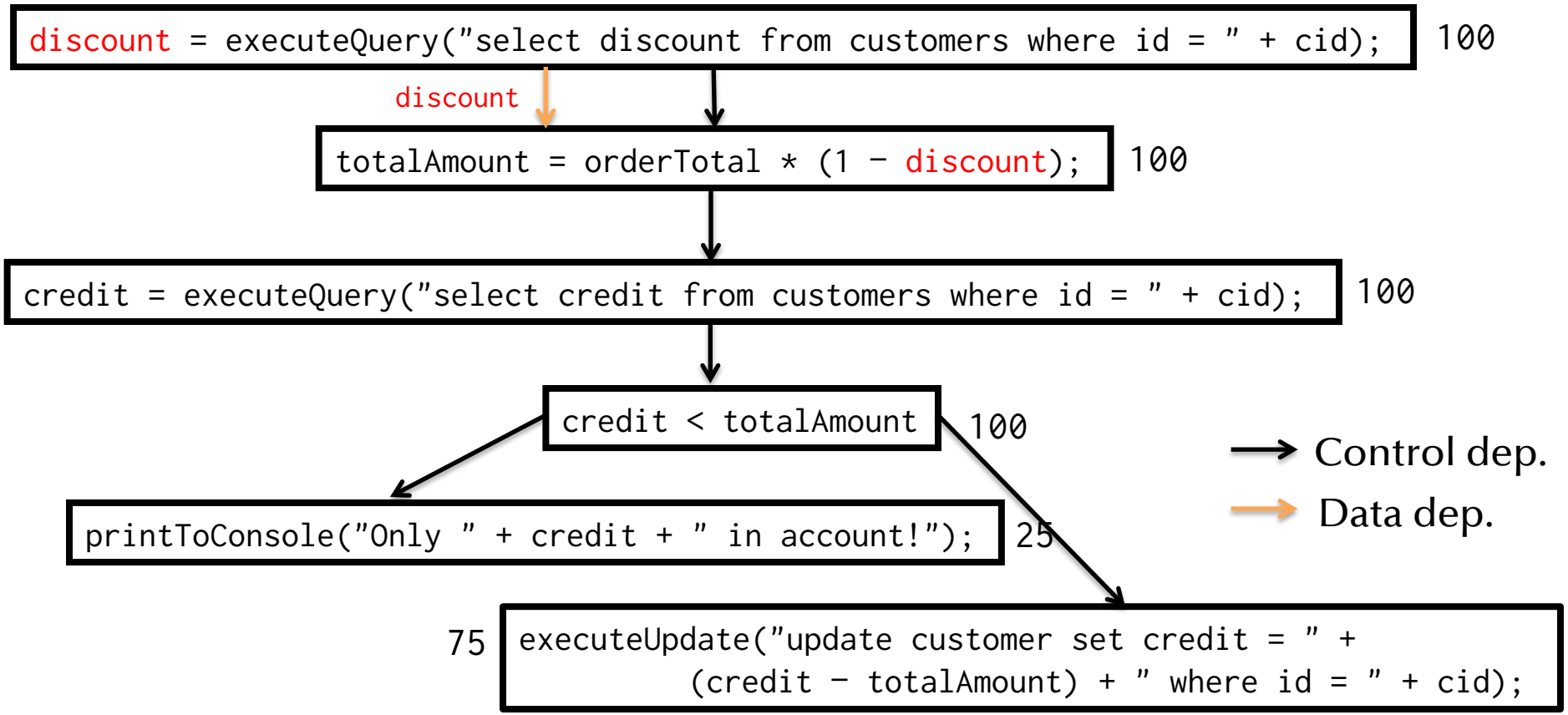
Node Weights

- Create program dependence graph that describes control flow and links up variable definitions and uses



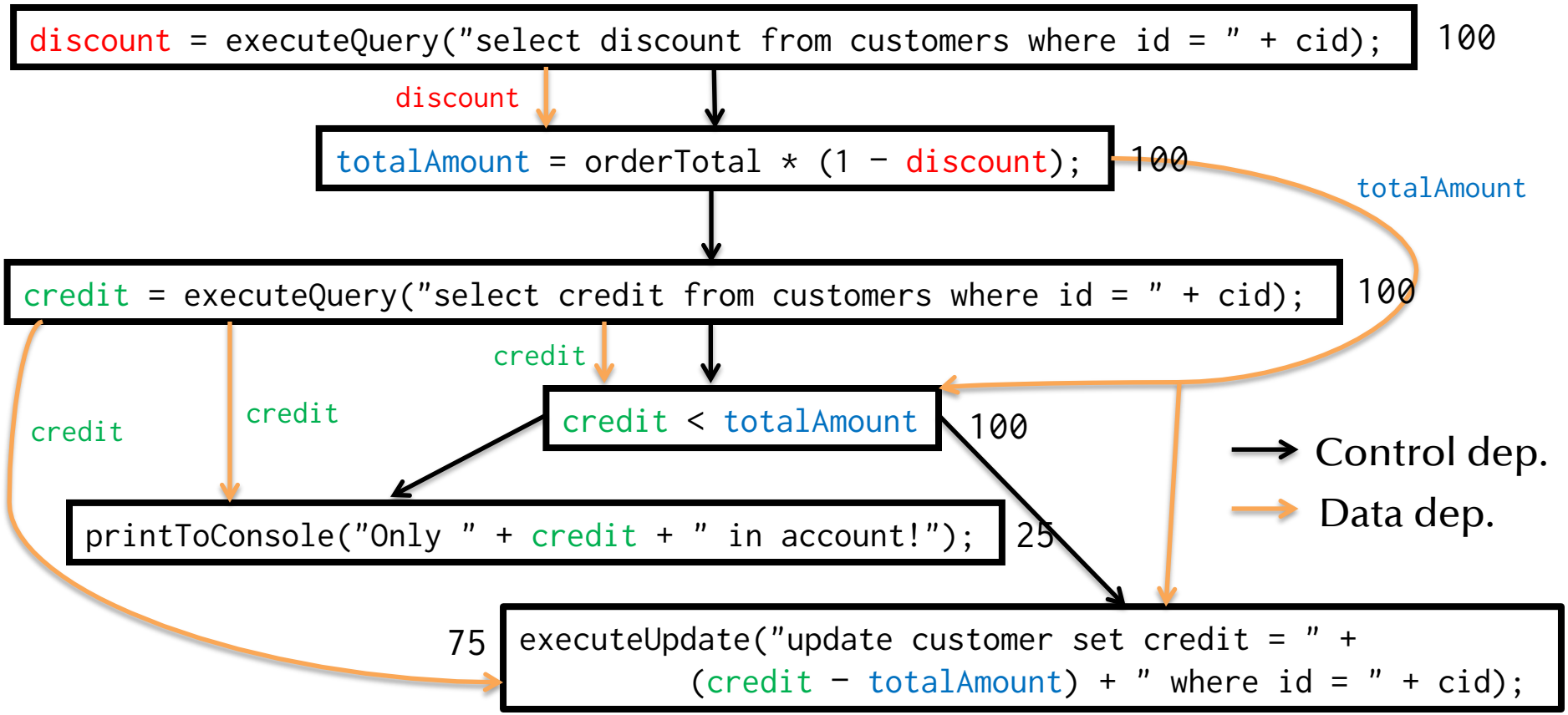
Data Dependencies

- Create program dependence graph that describes control flow and links up variable definitions and uses



Data Dependencies

- Create program dependence graph that describes control flow and links up variable definitions and uses



Linear Program Formulation

Minimize: $\sum_{e_i \in \text{edges}} e_i \cdot w_{e_i}$

Subject to: $e_i = 1$ iff $n_i \neq n_j$ where e_i connects n_i and n_j, and 0 otherwise

$$\sum_{n_i \in \text{nodes}} n_i \cdot w_{n_i} \leq \text{budget}$$

Solution

$n_i = 0 \rightarrow$ APP

$n_i = 1 \rightarrow$ DB

Represented using Pyxil (PYXis Intermediate Language)

Pyxil Example

```
:D: discount = executeQuery("select discount from customers
                             where id = " + cid);
:D: totalAmount = orderTotal * (1 - discount);
:D: credit = executeQuery("select credit from customers
                           where id = " + cid);

if (:D: credit < totalAmount)
   :A: printToConsole("Only " + credit + " in account!");
else
   :D: executeUpdate("update customer set credit = " +
                             (credit - totalAmount) +
                     " where id = " + cid);
```

Pyxil Compilation & Runtime

Pyxil Compiler and Runtime

- Compiles pyxil program into two Java programs
  - To be executed by the Pyxis runtimes on the app and DB server
- Pyxis runtime is simply a Java program running on a standard JVM on the two servers

Pyxil Compilation

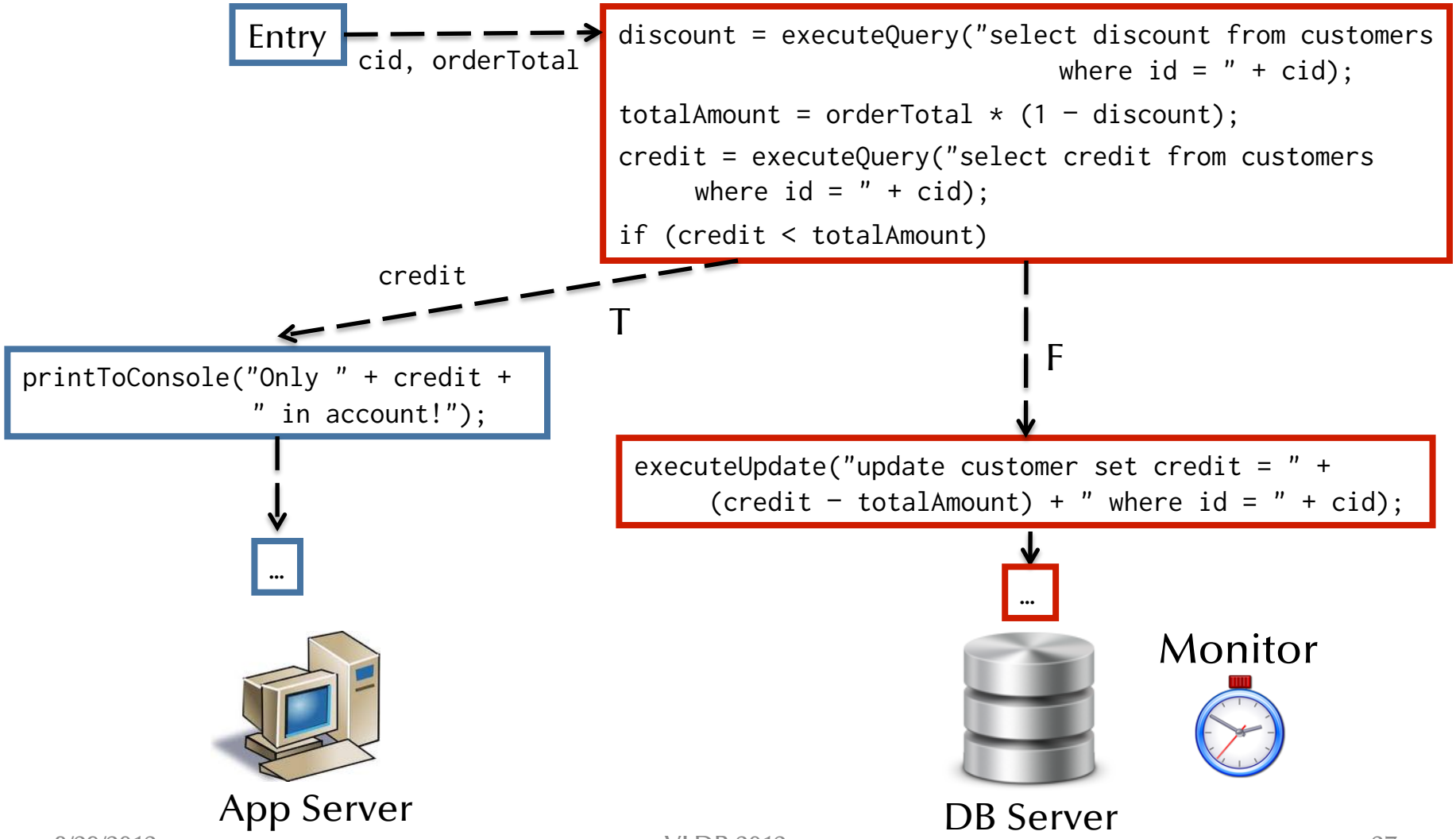
- Two issues:
  - Control transfer implementation
  - Heap Synchronization

Control Transfer and Heap Sync

```
DB  :D: discount = executeQuery("select discount from customers
                                 where id = " + cid);
    :D: totalAmount = orderTotal * (1 - discount);
DB  :D: credit = executeQuery("select credit from customers
                               where id = " + cid);

    if (:D: credit < totalAmount)
APP    :A: printToConsole("Only " + credit + " in account!");
    else
DB     :D: executeUpdate("update customer set credit = " +
                         (credit - totalAmount) +
                    " where id = " + cid);
```

Control Transfer and Heap Sync



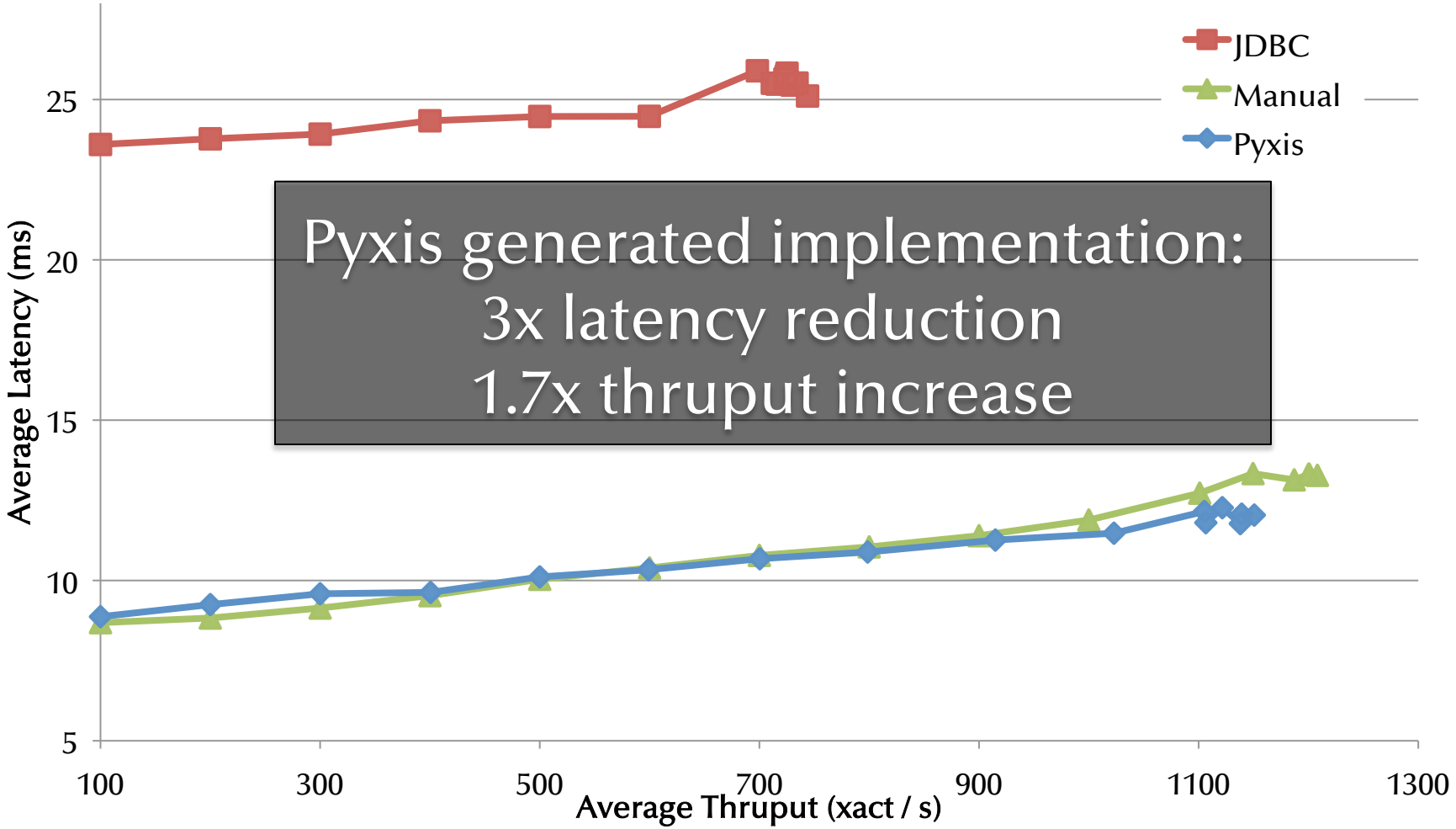
Experiments

Experiment Setup

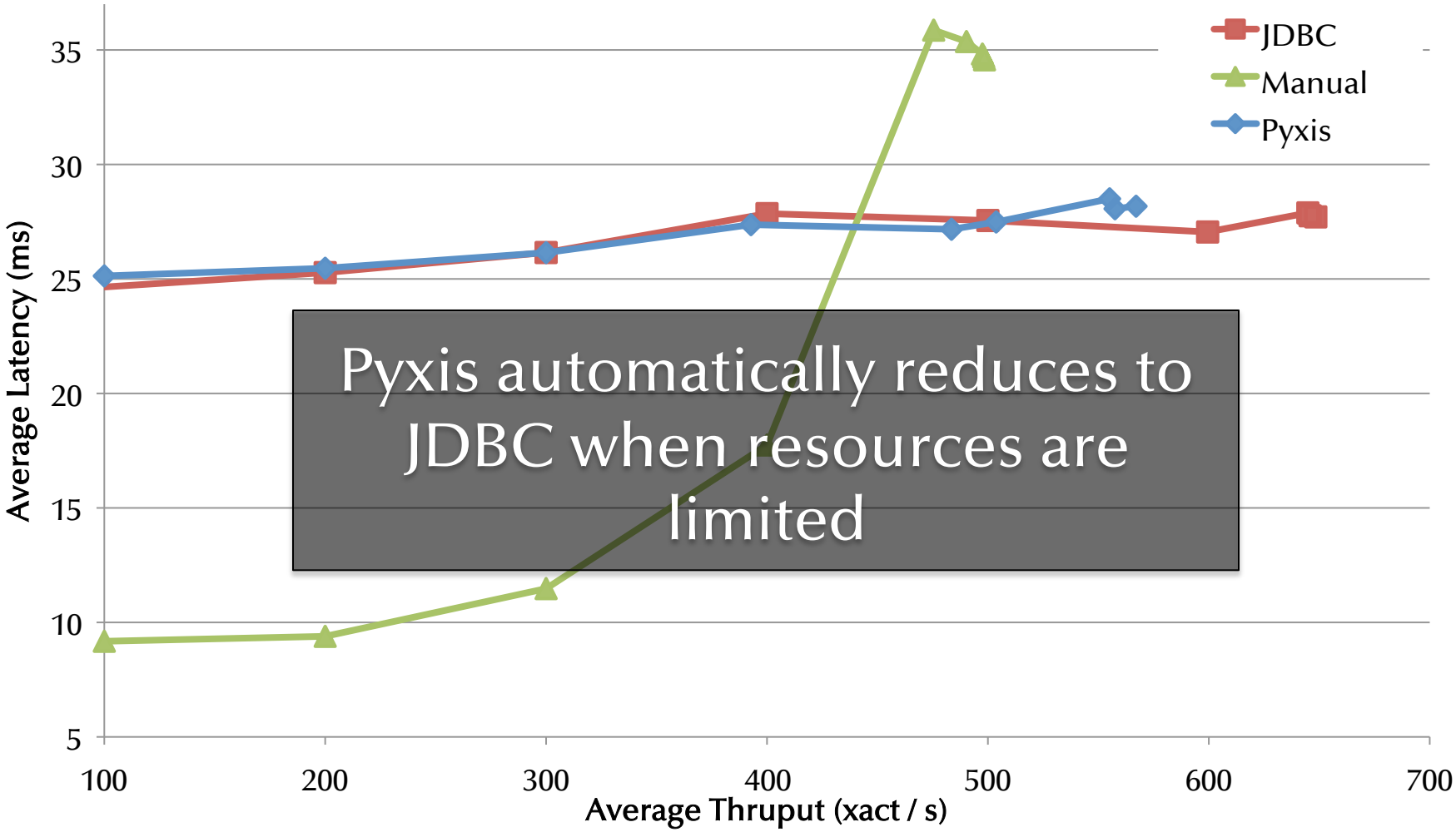
- TPC-C Java implementation
  - 20 terminals issuing new order transactions
  - 0.2ms RTT between app and DB servers
  - DB server has 16 cores total

  - Compared against two implementations:
    - JDBC: everything on app server except for JDBC stmts
    - Manual: custom "store procedurized" implementation where everything is on the DB server

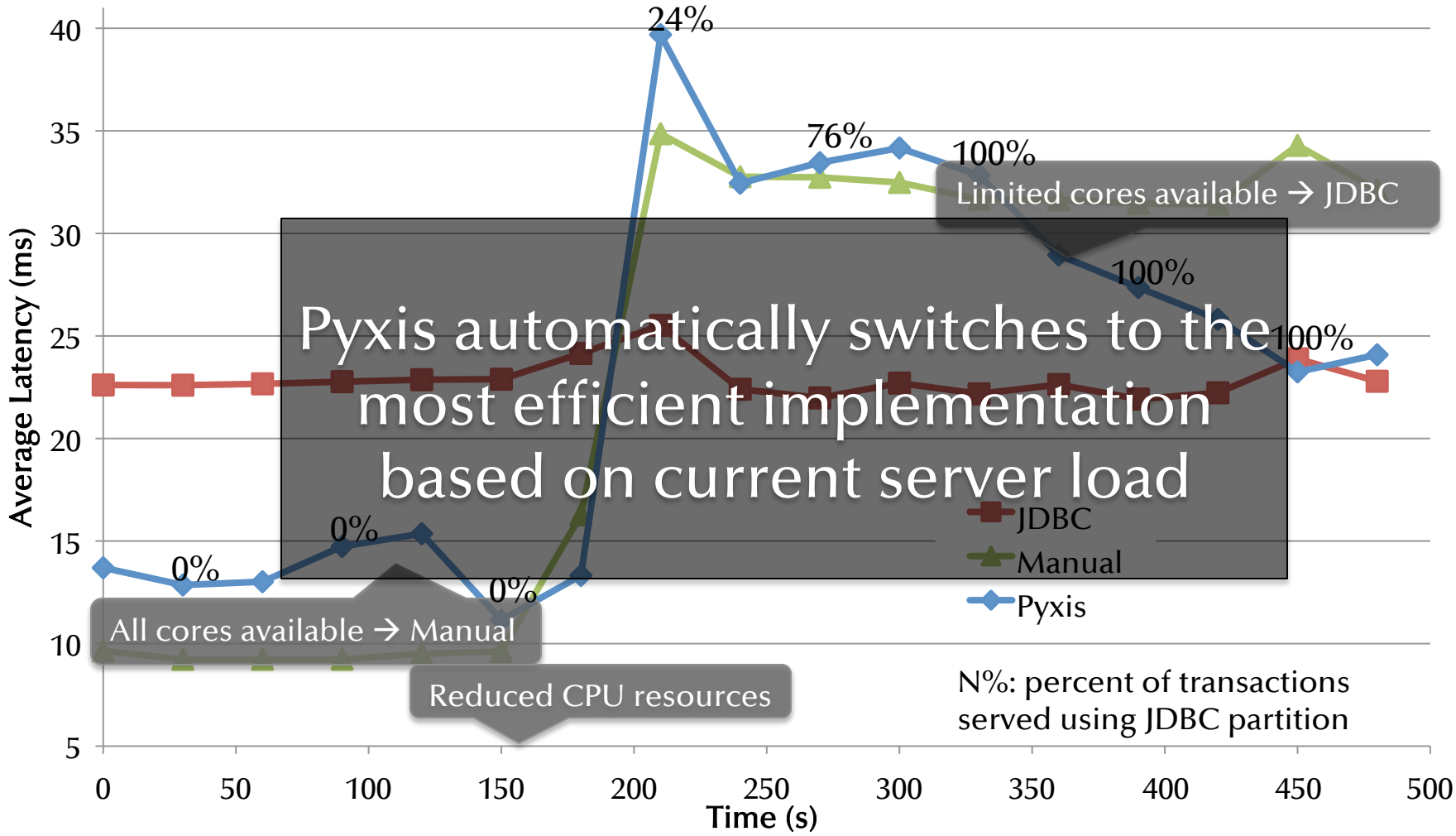
All Cores Available



Limited Cores Available



Dynamic Switching



Pyxis

Ease DB application development

Fully automatic code partitioning using application and server characteristics

Adaptive optimization based on server load

`db.csail.mit.edu/pyxis`