Becoming friends

— A common foundation for data processing —

Sebastian Ertel

Programming Languages vs. Databases

Analytical Queries
(TPC-H)

Data Processing Engine

Transactional Programs
(TPC-C)

Data Integration
(ETL, BI-Tools)

Data Streaming
(CQL, etc.)

Data Mining
(Machine Learning)

Programming Languages vs. Databases

Analytical Queries
(TPC-H)

Data Processing Engine

Transactional Programs
(TPC-C)

Data Integration
(ETL, BI-Tools)

Data Streaming
(CQL, etc.)

Data Mining
(Machine Learning)

Programming Languages vs. Databases

Analytical Queries
(TPC-H)

SQL / JavaScript

Data Processing Engine

C/C++, Java, (Scala)

Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.

Data Integration
(ETL, BI-Tools)

Data Streaming
(CQL, etc.)

Data Mining
(Machine Learning)

Programming Languages vs. Databases

Analytical Queries
(TPC-H)

SQL / JavaScript

Data Processing Engine

C/C++, Java, (Scala)

Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.

Dataflow

- parallel execution
- I/O optimizations
- dynamic software evolution

Data Integration
(ETL, BI-Tools)

Data Streaming
(CQL, etc.)

Data Mining
(Machine Learning)

Sebastian Ertel, Andrés Goens, Justus Adam, and Jeronimo Castrillon. *Compiling for concise code and efficient I/O*. CC 2018.
Sebastian Ertel, Justus Adam, and Jeronimo Castrillon. *Supporting fine-grained dataflow parallelism in big data systems*. PMAM 2018.
Sebastian Ertel and Pascal Felber. *A framework for the dynamic evolution of highly-available data flow programs*. Middleware 2015.

Programming Languages vs. Databases

Analytical Queries
(TPC-H)

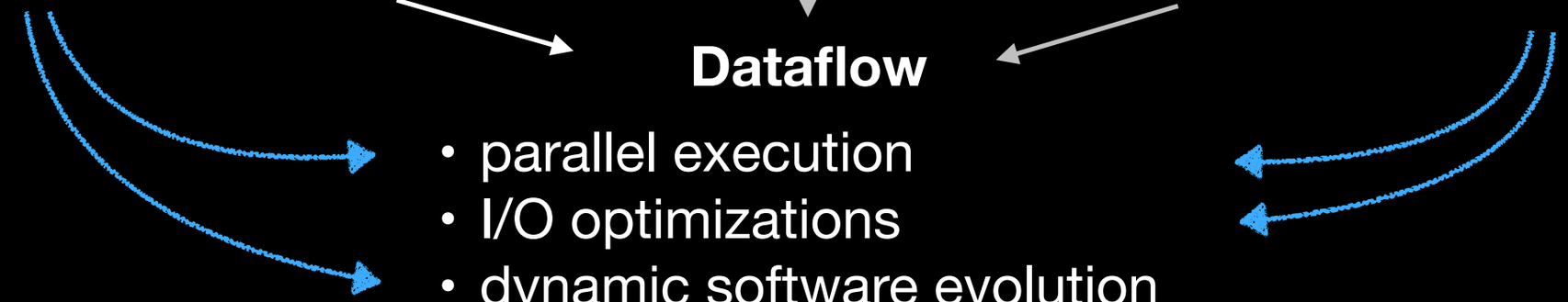
SQL / JavaScript

Data Processing Engine

C/C++, Java, (Scala)

Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.



Dataflow

- parallel execution
- I/O optimizations
- dynamic software evolution

Data Integration
(ETL, BI-Tools)

Data Streaming
(CQL, etc.)

Data Mining
(Machine Learning)

Sebastian Ertel, Andrés Goens, Justus Adam, and Jeronimo Castrillon. *Compiling for concise code and efficient I/O*. CC 2018.
Sebastian Ertel, Justus Adam, and Jeronimo Castrillon. *Supporting fine-grained dataflow parallelism in big data systems*. PMAM 2018.
Sebastian Ertel and Pascal Felber. *A framework for the dynamic evolution of highly-available dataflow programs*. Middleware 2015.

Programming Languages vs. Databases

Analytical Queries
(TPC-H)

SQL / JavaScript

Data Processing Engine

C/C++, Java, (Scala)

Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.

Dataflow

- parallel execution
- I/O optimizations
- dynamic software evolution

Data Integration
(ETL, BI-Tools)

Data Streaming
(CQL, etc.)

Data Mining
(Machine Learning)

Sebastian Ertel, Andrés Goens, Justus Adam, and Jeronimo Castrillon. *Compiling for concise code and efficient I/O*. CC 2018.
Sebastian Ertel, Justus Adam, and Jeronimo Castrillon. *Supporting fine-grained dataflow parallelism in big data systems*. PMAM 2018.
Sebastian Ertel and Pascal Felber. *A framework for the dynamic evolution of highly-available data flow programs*. Middleware 2015.

Programming Languages vs. Databases

Analytical Queries
(TPC-H)

SQL / JavaScript

Data Processing Engine

C/C++, Java, (Scala)

Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.

Dataflow

- parallel execution
- I/O optimizations
- dynamic software evolution

Data Integration
(ETL, BI-Tools)

Data Streaming
(CQL, etc.)

Data Mining
(Machine Learning)

Sebastian Ertel, Andrés Goens, Justus Adam, and Jeronimo Castrillon. *Compiling for concise code and efficient I/O*. CC 2018.
Sebastian Ertel, Justus Adam, and Jeronimo Castrillon. *Supporting fine-grained dataflow parallelism in big data systems*. PMAM 2018.
Sebastian Ertel and Pascal Felber. *A framework for the dynamic evolution of highly-available data flow programs*. Middleware 2015.

Enter Ohua!

Algorithm:

(Stateful) function:

https://ohua-dev.github.io/ohua/

Enter Ohua!

Algorithm:

```
void mapAlgo(InputIterator recordsOnDisk) {
  var mapper = initMapper();
  var writer = initWriter();

  for(var chunk : recordsOnDisk){
    var (line, content) = chunk;
    var kvPairs = mapper.map(line, content);
    for(var kvPair : kvPairs){
      var (key, value) = kvPair;
      writer.output(key, value);
    }
  }
}
```

(Stateful) function:

https://ohua-dev.github.io/ohua/

Enter Ohua!

Algorithm:

```
void mapAlgo(InputIterator recordsOnDisk) {
  var mapper = initMapper();
  var writer = initWriter();

  for(var chunk : recordsOnDisk){
    var (line, content) = chunk;
    var kvPairs = mapper.map(line, content);
    for(var kvPair : kvPairs){
      var (key, value) = kvPair;
      writer.output(key, value);
    }
  }
}
```

(Stateful) function:

```
Writer initWriter(){
  return new Writer();
}

class Writer {
  private Mapper$Context _ctxt; // state

  public Writer(){ /* code omitted */ }

  void write(Object key, Object value){
    _ctxt.write(key, value);
  }
}
```

https://ohua-dev.github.io/ohua/

Enter Ohua!

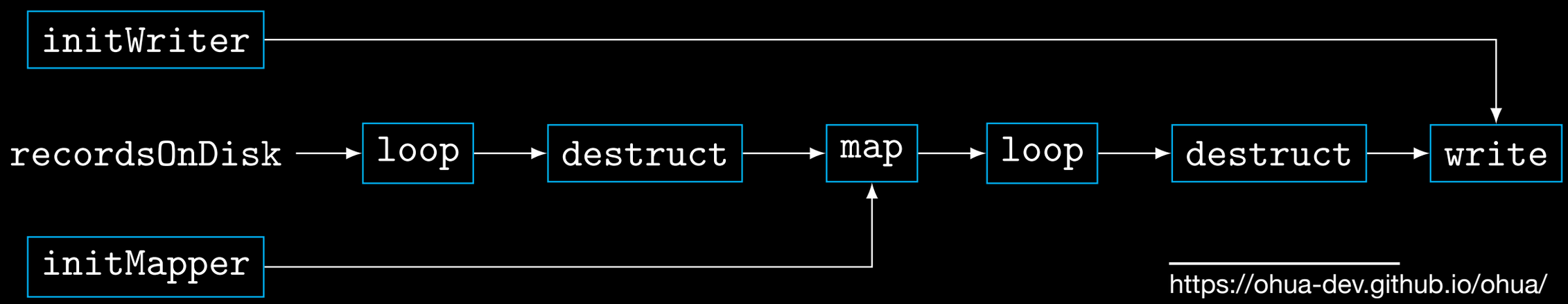
Algorithm:

```
void mapAlgo(InputIterator recordsOnDisk) {
  var mapper = initMapper();
  var writer = initWriter();

  for(var chunk : recordsOnDisk){
    var (line, content) = chunk;
    var kvPairs = mapper.map(line, content);
    for(var kvPair : kvPairs){
      var (key, value) = kvPair;
      writer.output(key, value);
    }
  }
}
```

(Stateful) function:

```
Writer initWriter(){
  return new Writer();
}

class Writer {
  private Mapper$Context _ctxt; // state

  public Writer(){ /* code omitted */ }

  void write(Object key, Object value){
    _ctxt.write(key, value);
  }
}
```

Dataflow Graph:



https://ohua-dev.github.io/ohua/

Enter Ohua!

Algorithm:

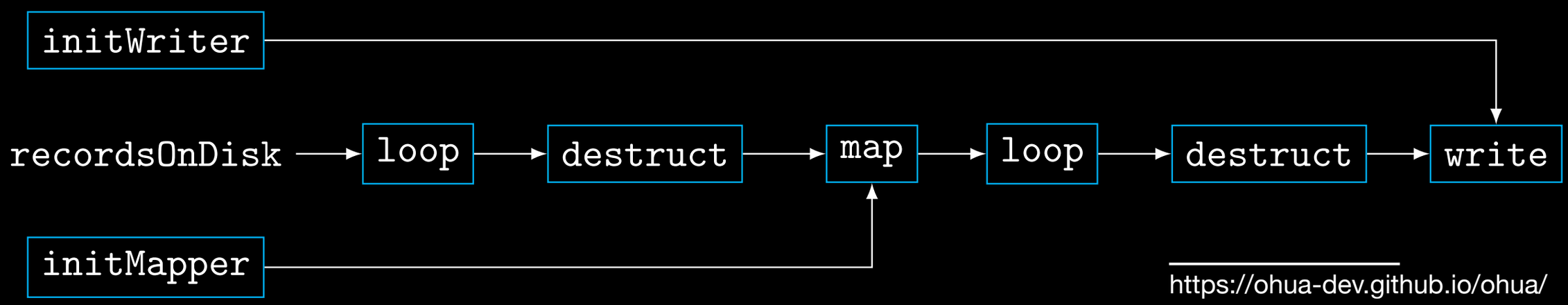
```
void mapAlgo(InputIterator recordsOnDisk) {
  var mapper = initMapper();
  var writer = initWriter();

  for(var chunk : recordsOnDisk){
    var (line, content) = chunk;
    var kvPairs = mapper.map(line, content);
    for(var kvPair : kvPairs){
      var (key, value) = kvPair;
      writer.output(key, value);
    }
  }
}
```

(Stateful) function:

```
Writer initWriter(){
  return new Writer();
}

clas
private Mapper$Context _ctxt; // state

public Writer(){ /* code omitted */ }

void write(Object key, Object value){
  _ctxt.write(key, value);
}
}
```

lambda-calculus-based IR

Dataflow Graph:



https://ohua-dev.github.io/ohua/

Enter Ohua!

Algorithm:

```
void mapAlgo(InputIterator recordsOnDisk) {
  var mapper = init
  var writer = init

  for(var chunk : r                ){
    var (line, cont                ;
    var kvPairs = m                ne, content);
    for(var kvPair
      var (key, val                ;
      writer.output                ;
    }
  }
}
```

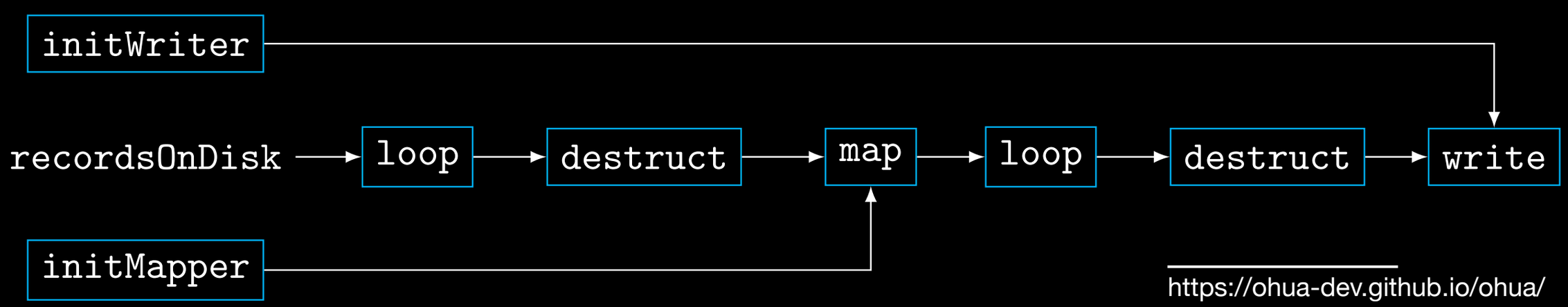
Rust
Java
Scala
C
C++
Haskell
JavaScript
...

(Stateful) function:

```
Writer initWriter(){
  return new Writer();
}

clas
  private Mapper$Context _ctxt; // state

  public Writer(){ /* code omitted */ }

  void write(Object key, Object value){
    _ctxt.write(key, value);
  }
}
```

lambda-calculus-based IR

Dataflow Graph:



https://ohua-dev.github.io/ohua/

Enter Ohua!

Algorithm:

```
void mapAlgo(InputIterator recordsOnDisk) {
  var mapper = init
  var writer = init

  for(var chunk : r
    var (line, cont
    var kvPairs = m
    for(var kvPair
      var (key, val
      writer.output
    }
  }
}
```

Rust
Java
Scala
C
C++
Haskell
JavaScript
...

(Stateful) function:

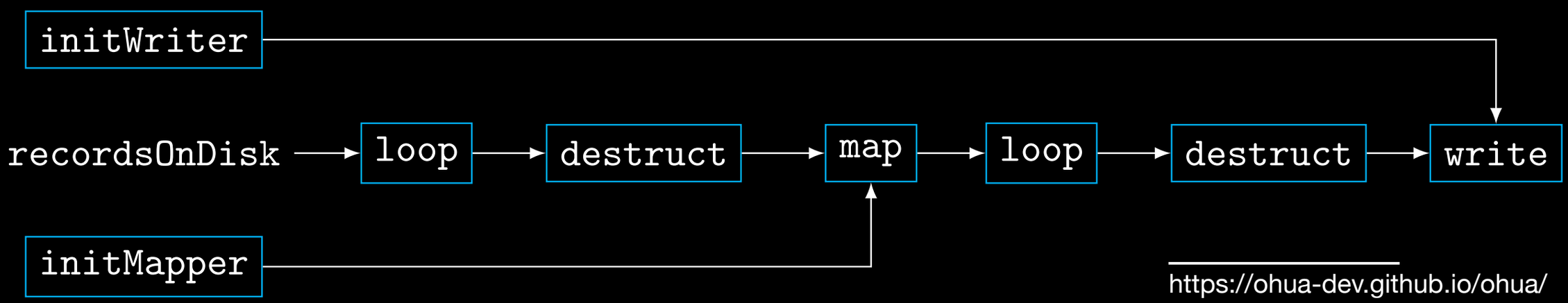
```
Writ
  re
}

clas
  pr
pu
void write(Object key, Object value){
  _ctxt.write(key, value);
}
}
```

lambda-calculus-based IR

generic optimizations

domain-specific optimizations

Dataflow Graph:



https://ohua-dev.github.io/ohua/

Fully-integrated Dataflow Parallelism

Analytical Queries
(TPC-H)

SQL / JavaScript

✓ ✗

Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.

✓ ✗

<u>Challenge</u>: Speed is in SQL vs. rich JavaScript eco system!

Fully-integrated Dataflow Parallelism

Analytical Queries
(TPC-H)

SQL / JavaScript

✓ ✗

Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.

✓ ✗

Dataflow

Example UDF:

```
function factorial(D) {
  if (D <= 0) {
    return 1;
  } else {
    var result = 1;
    for (var i = 2; i <= D; i++) {
      result = result * i;
    }
    return result;
  }
}
```

Fully-integrated Dataflow Parallelism

Analytical Queries
(TPC-H)

SQL / JavaScript

✓ ✗

Transactional Programs
(TPC-C)

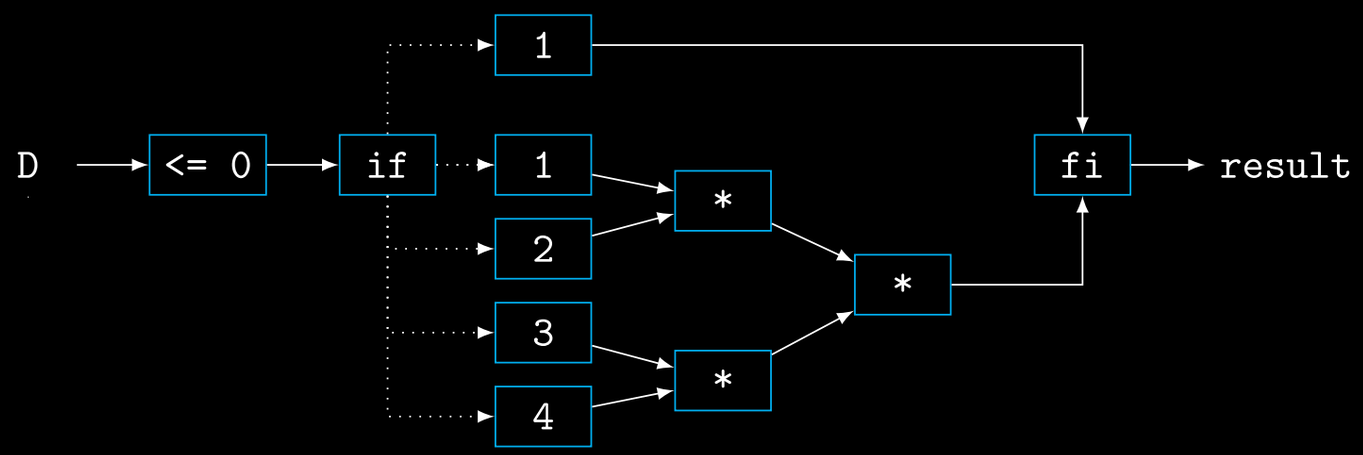
SQL / C/C++, Java, etc.

✓ ✗

Dataflow

Example UDF:

```
function factorial(D) {
  if (D <= 0) {
    return 1;
  } else {
    var result = 1;
    for (var i = 2; i <= D; i++) {
      result = result * i;
    }
    return result;
  }
}
```



Fully-integrated Dataflow Parallelism

Analytical Queries
(TPC-H)

SQL / JavaScript

✓ ✗

Transactional Programs
(TPC-C)

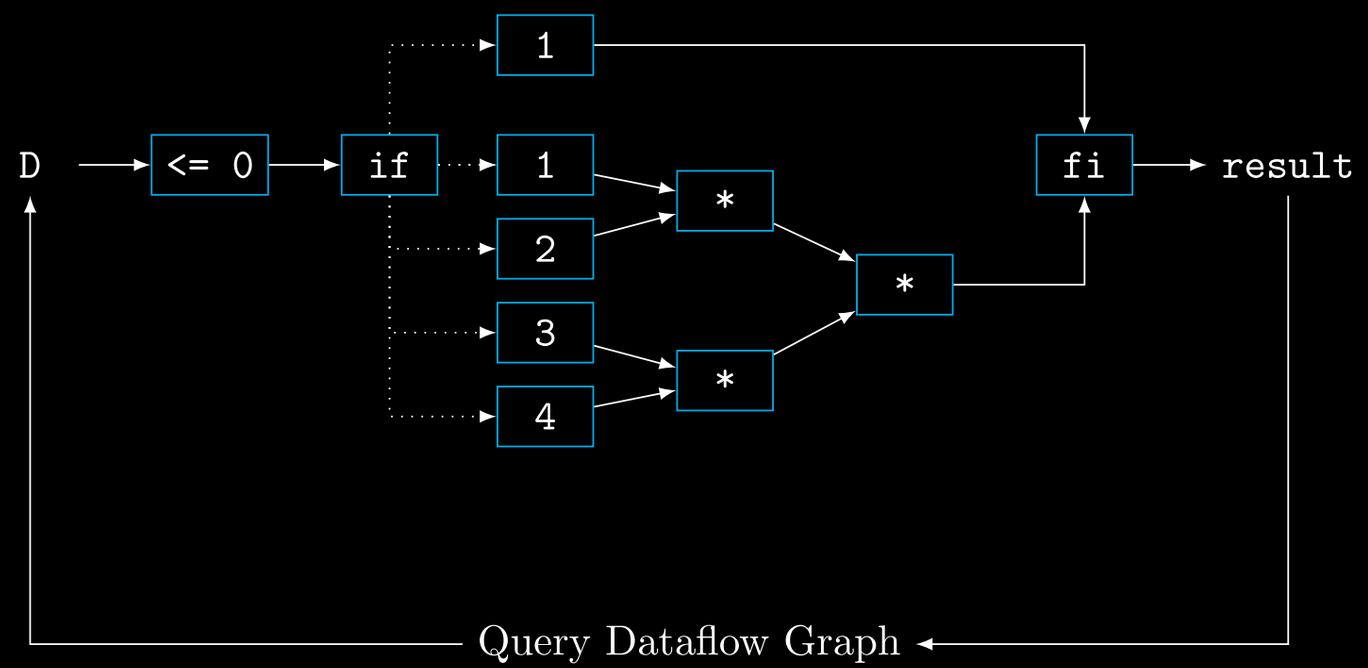
SQL / C/C++, Java, etc.

✓ ✗

Dataflow

Example UDF:

```
function factorial(D) {
  if (D <= 0) {
    return 1;
  } else {
    var result = 1;
    for (var i = 2; i <= D; i++) {
      result = result * i;
    }
    return result;
  }
}
```



Fully-integrated Dataflow Parallelism

Analytical Queries
(TPC-H)

SQL / JavaScript

✓ ✓

Transactional Programs
(TPC-C)

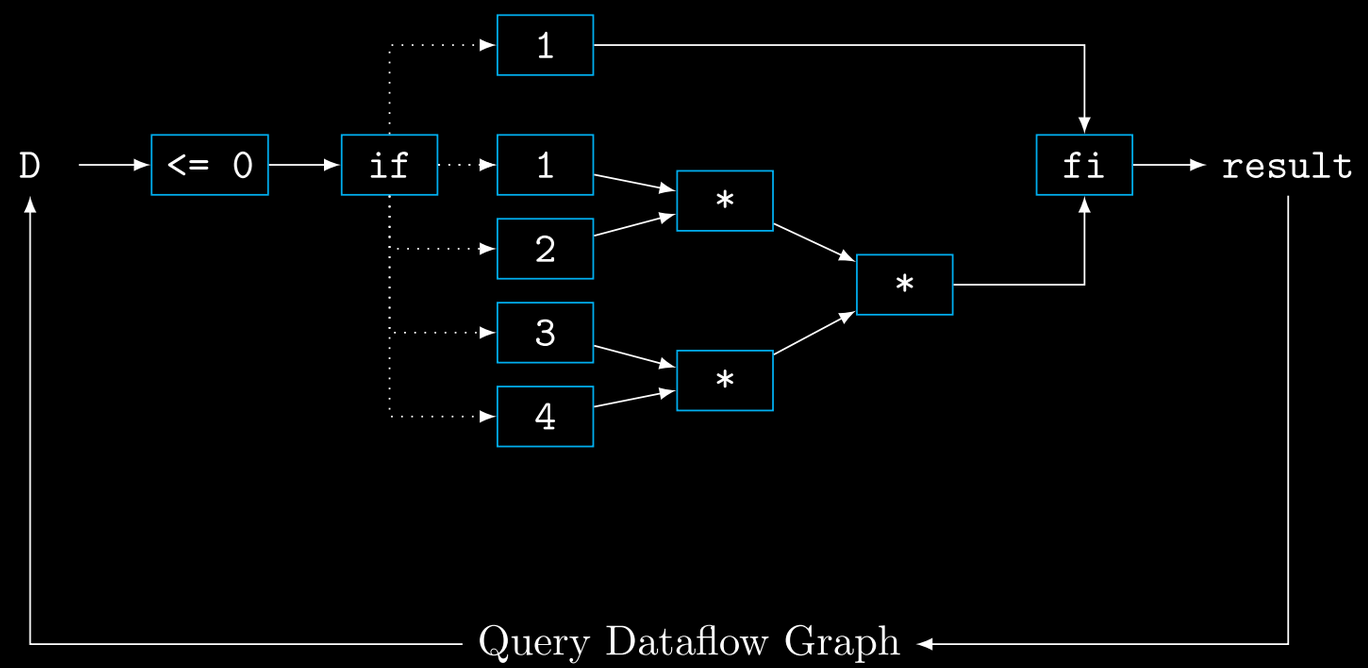
SQL / C/C++, Java, etc.

✓ ✗

Dataflow

Example UDF:

```
function factorial(D) {
  if (D <= 0) {
    return 1;
  } else {
    var result = 1;
    for (var i = 2; i <= D; i++) {
      result = result * i;
    }
    return result;
  }
}
```



One data flow graph to rule them all!

Fully-integrated Dataflow Parallelism

Analytical Queries
(TPC-H)

SQL / JavaScript

✓ ✓

Transactional Programs
(TPC-C)

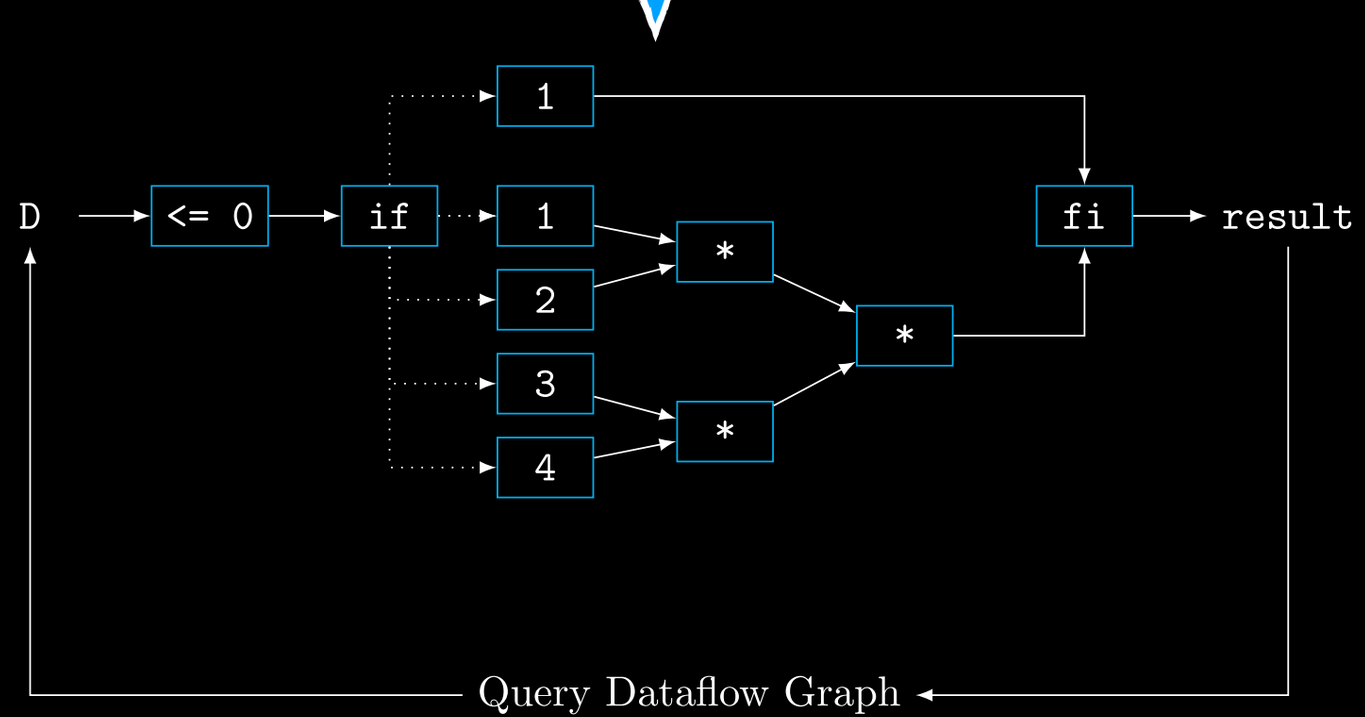
SQL / C/C++, Java, etc.

More optimizations apply:
pure function -> caching

Example UDF:

```
function factorial(D) {
  if (D <= 0) {
    return 1;
  } else {
    var result = 1;
    for (var i = 2; i <= D; i++) {
      result = result * i;
    }
    return result;
  }
}
```

Dataflow

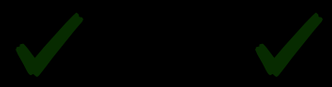


One data flow graph to rule them all!

Compiling for Efficient I/O

Analytical Queries
(TPC-H)

SQL / JavaScript



Transactional Programs
(TPC-C)

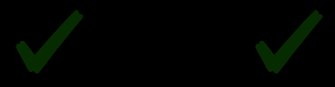
SQL / C/C++, Java, etc.


<u>Challenge</u>: Shipping computation vs. shipping data!

Compiling for Efficient I/O

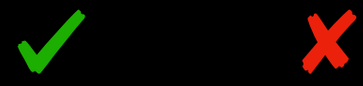
Analytical Queries
(TPC-H)

SQL / JavaScript



Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.



New Order TA (TPC-C):

Compiling for Efficient I/O

Analytical Queries
(TPC-H)

SQL / JavaScript

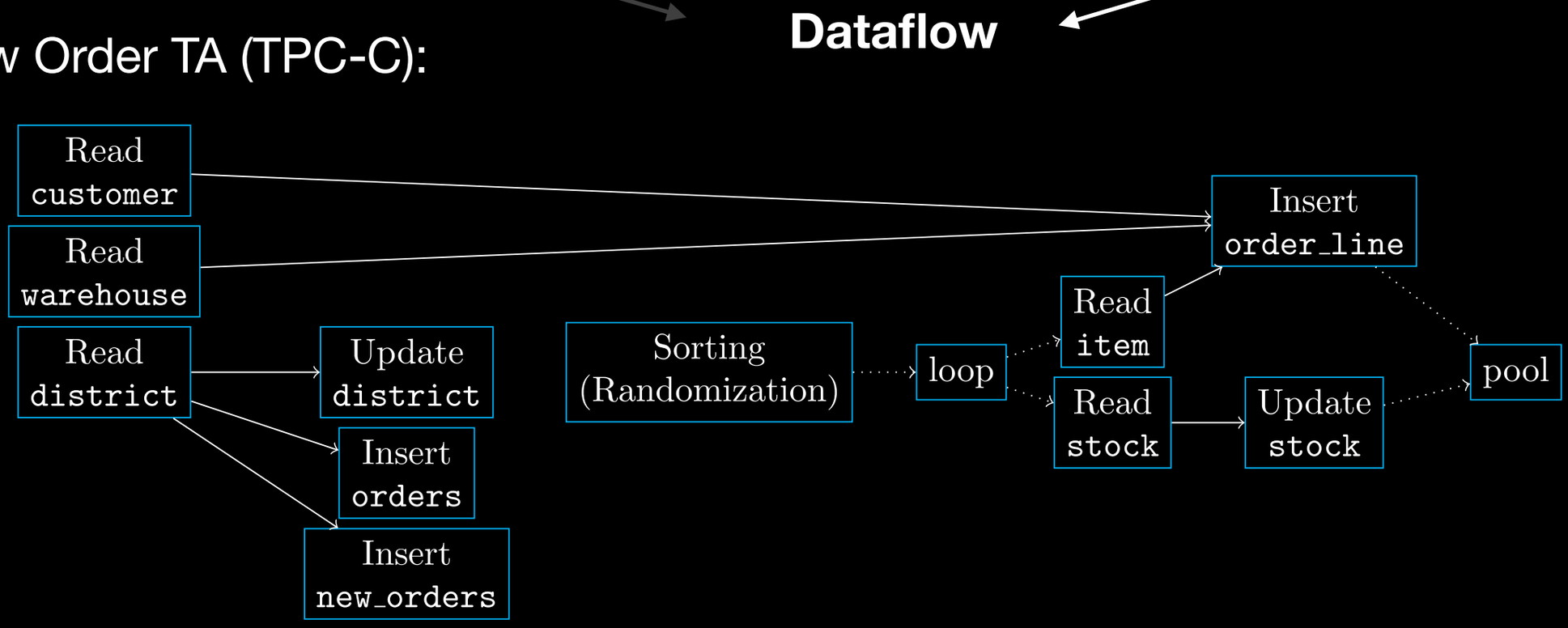
✓ ✓

Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.

✓ ✓

New Order TA (TPC-C):



Compiling for Efficient I/O

Analytical Queries
(TPC-H)

SQL / JavaScript

✓ ✓

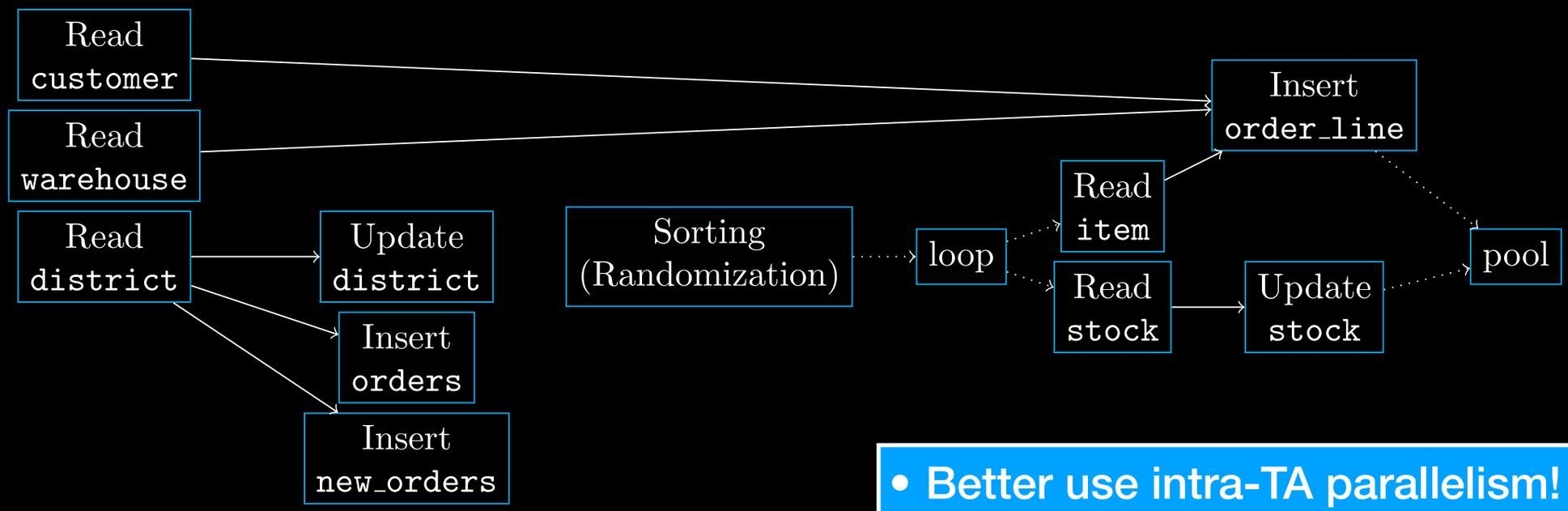
Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.

✓ ✓

Dataflow

New Order TA (TPC-C):



- **Better use intra-TA parallelism!**

Compiling for Efficient I/O

Analytical Queries
(TPC-H)

SQL / JavaScript

✓ ✓

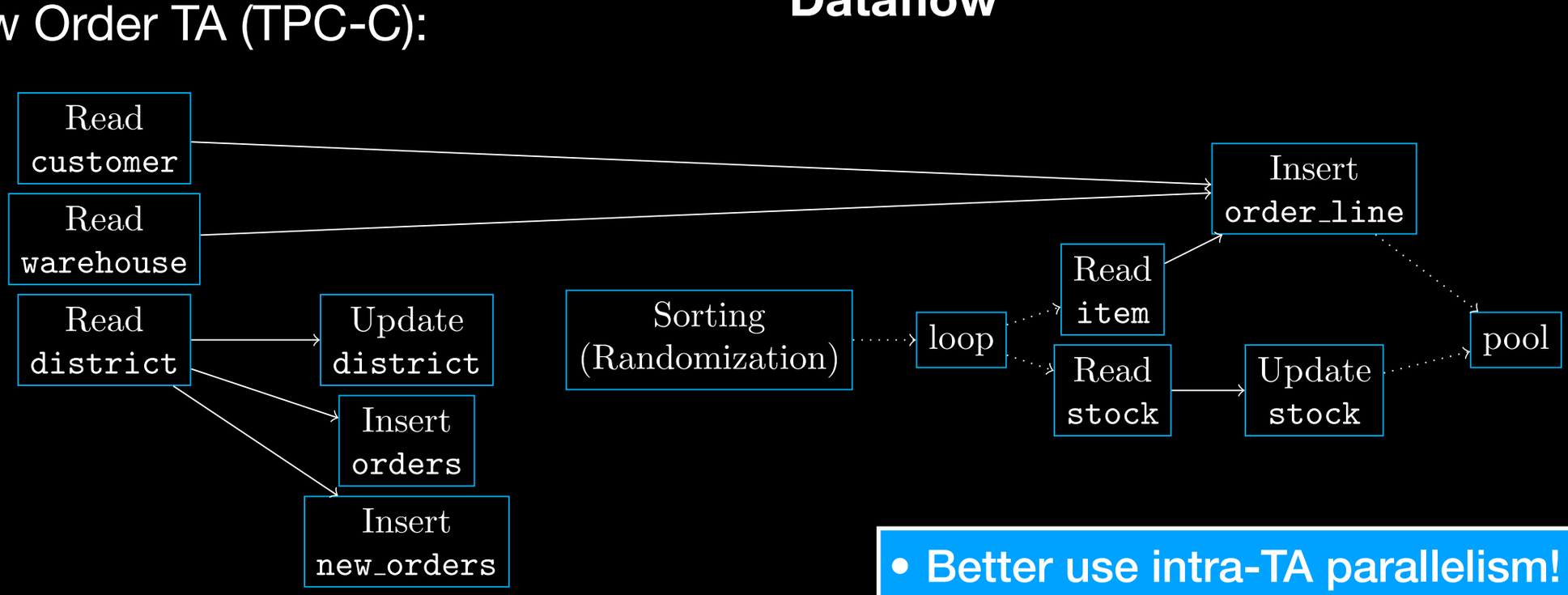
Transactional Programs
(TPC-C)

SQL / C/C++, Java, etc.

✓ ✓

Dataflow

New Order TA (TPC-C):



- **Better use intra-TA parallelism!**
- **Migrate not only query but computation!**

Key Insight

Instead of SQL *vs.* Java, C/C++ etc. better have a *common* foundation!

Thank you!