# Lessons Learned Implementing StreamBase

Matt Fowles

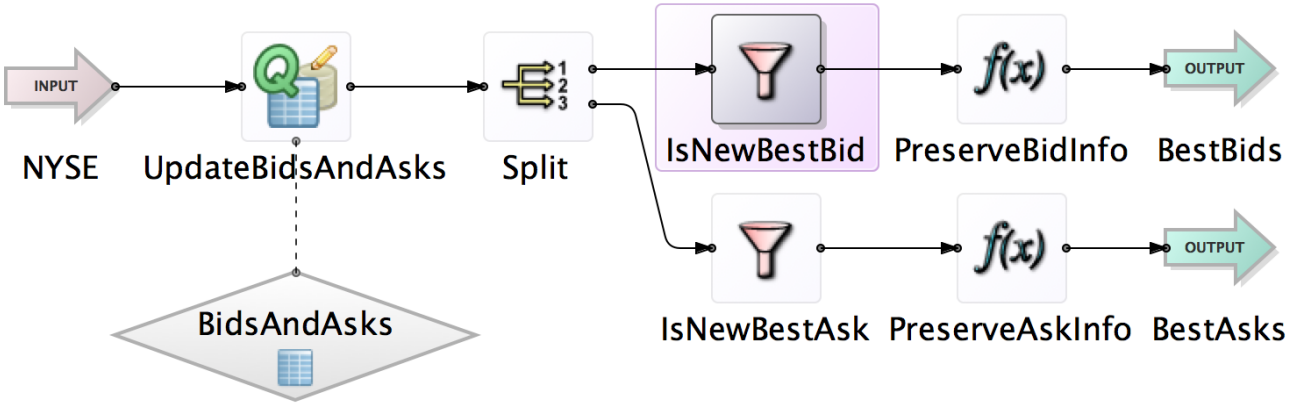
StreamBase Systems

JVM Languages 2012

# Agenda

- What is StreamBase?
- Lessons Learned
  - Spaghetti code is bad; Spaghetti code generation is worse
  - Choose between Generated Code vs Runtime Library
  - Parallel APIs for generating code and for runtime code
  - Type Specialization makes code fast
  - Separate Compilation requires planning
  - Classloaders hurt the unwary
- Case Studies
  - Inherited control flow enables Disruptors
  - Pluggable table implementations and Client API Access
  - JDBC
- Conclusions and Future Work

# StreamBase EventFlow




Editor | Definitions | Parameters | Dynamic Variables | **Annotations** | Metadata

StreamBase Properties (IsNewBestBid – Filter operator) | Typecheck Errors | Problems | StreamBase Extension Points | Console

Specify at least one predicate by adding and completing rows in the table below.

General | Predicate Settings | Concurrency

Create output port for non-matching tuples

**Predicates:**

| Output Port | Predicate |
|---|---|
| 1 | best_bid == bid_price |

Streams | Functions | Expression QuickRef

Input | Output

- input1 (6 fields)
  - time_int int
  - symbol string
  - bid_price double
  - ask_price double
  - best_bid double
  - best_ask double

# Problem Domain for Event Processing

- Interoperate with many external systems
  - Java, C++ Libraries
  - External Systems (Reuters, 29West)
- Large Heaps (40-120 GB)
- Low Latency (20-2000 microseconds)
- Predictable performance and efficiency
- Easily Shardable
- Rapid Development
- 6+ years of experience:
  - extendability
  - debugability
  - optimizability
  - refactorability
  - understandability

## Lessons Learned

- **Spaghetti code is bad; spaghetti code generation is worse**
- Choose between generated code vs runtime library
- Parallel APIs for generating code and for runtime code
- Type specialization makes code fast
- Separate compilation requires planning
- Classloaders hurt the unwary
- Improve tooling

## Lessons Learned

- Spaghetti code is bad; spaghetti code generation is worse
- **Choose between generated code vs runtime library**
- Parallel APIs for generating code and for runtime code
- Type specialization makes code fast
- Separate compilation requires planning
- Classloaders hurt the unwary
- Improve tooling

## Generated Code vs Runtime Library

"passive voice" "active voice"

- Eliminate virtual dispatch
- Allows type specialization
- Monomorphic call sites
- Extra level of indirection
- Data can get big
- Code can get big too

- Easier to debug
- Easier to read
- Easier to test
- Faster to code
- Faster start-up
- Polymorphic call sites
- Less indirection

# General Approach

- Abstract classes provide code flow within operators
- Generate type specializations
  - Heroic downcast
  - Cache according to type signature
- Generate mathematical expressions
- Generate application level control flow

```
abstract void fillPrimaryKeyFromDataClass(DiskPackage dp, DC dc);
abstract void fillDataClassFromValue(DC dc, DiskPackage sdp);
abstract DC createDataClass();

public DC getPrimaryImpl(DC pkey) {
  TableTransaction txn = null;  Cursor cursor = null;
  try {
    txn = beginTransaction(); cursor = openCursor(txn);

    DiskPackage pack = new DiskPackage();
    fillPrimaryKeyFromDataClass(pack, pkey);
    cursor.getSearchKey(pack);

    if (pack.hasValue()) {
      DC dc = createDataClass();
      fillDataClassFromValue(dc, pack);
      return dc;
    } else {
      return null;
    }
  } catch (DatabaseException de) {
    throw createEvalException(de);
  } finally {
    finishTransaction(txn, cursor);
  }
}
```

## Lessons Learned

- Spaghetti code is bad; spaghetti code generation is worse
- Choose between generated code vs runtime library
- **Parallel APIs for generating code and for runtime code**
- Type specialization makes code fast
- Separate compilation requires planning
- Classloaders hurt the unwary
- Improve tooling

# Active voice

```
/**
 * A collection of bit flipping operations, with useful names.
 */
public final class BitOps {
    //...
    public static int roundToWord(int i) {
        return (i + 3) & ~3;
    }
    //...
}
```

# Passive voice

```
/**
 * Generated equivalents for the operations in BitOps
 */
public final class BitOpsGen {
    //...
    public static CG.Expr roundToWord(CG.Expr i) {
        return new CG.Op.BitAnd(CG.Op.ADD(i, CG.literal(3)),
                                CG.literal(~3));
    }
    //...
}
```

## Lessons Learned

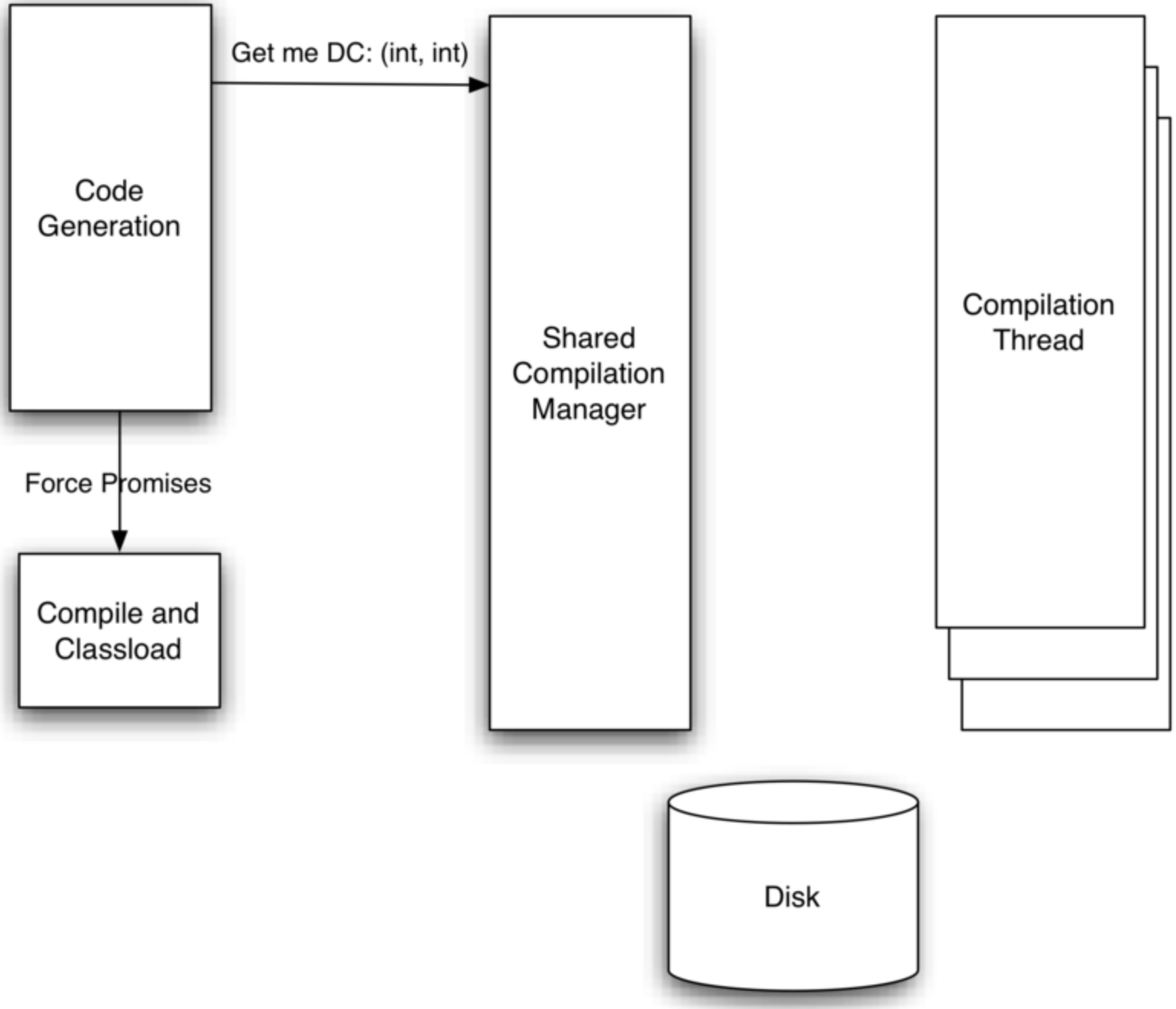
- Spaghetti code is bad; spaghetti code generation is worse
- Choose between generated code vs runtime library
- Parallel APIs for generating code and for runtime code
- **Type specialization makes code fast**
- Separate compilation requires planning
- Classloaders hurt the unwary
- Improve tooling

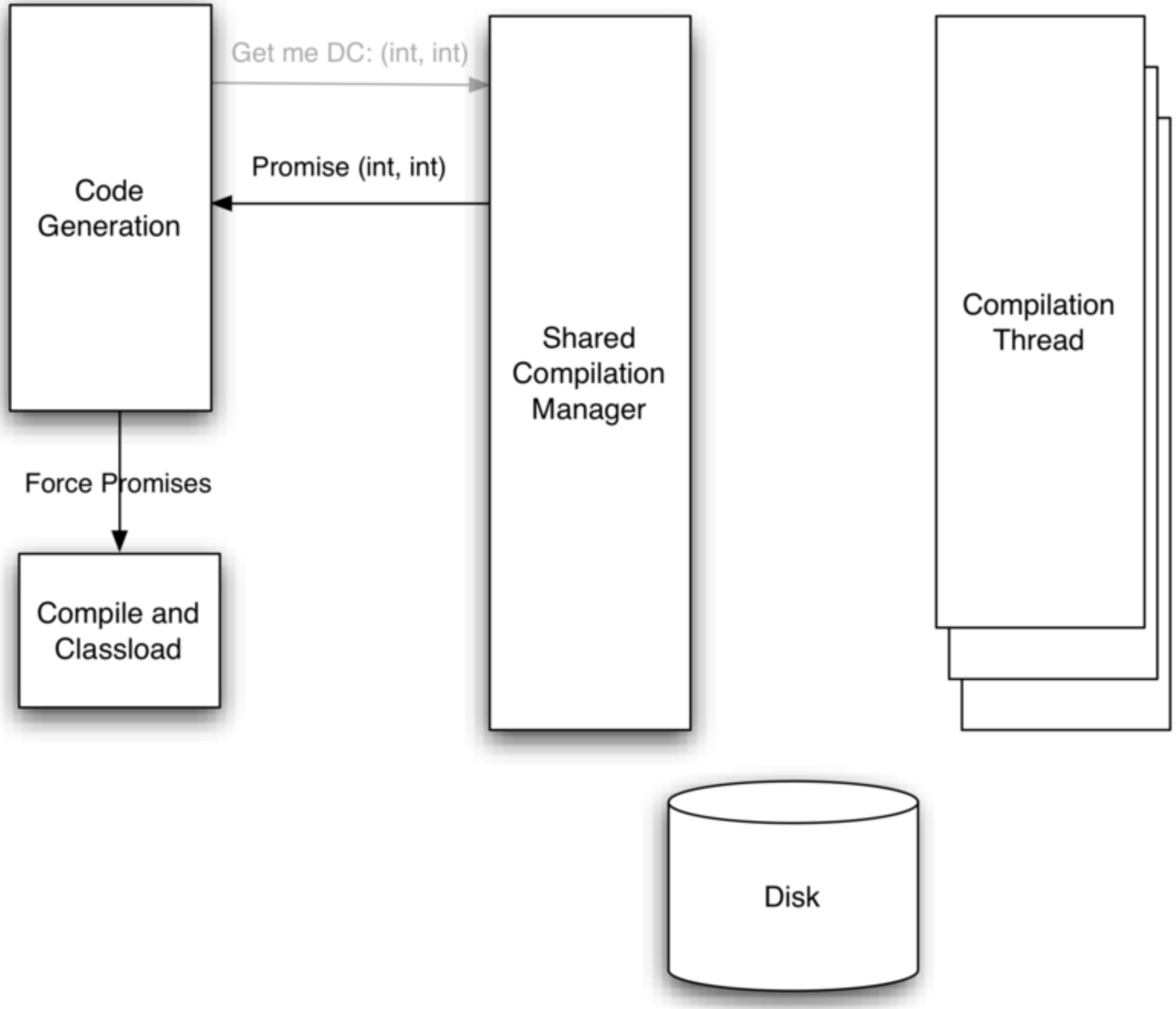
# Type Specialization

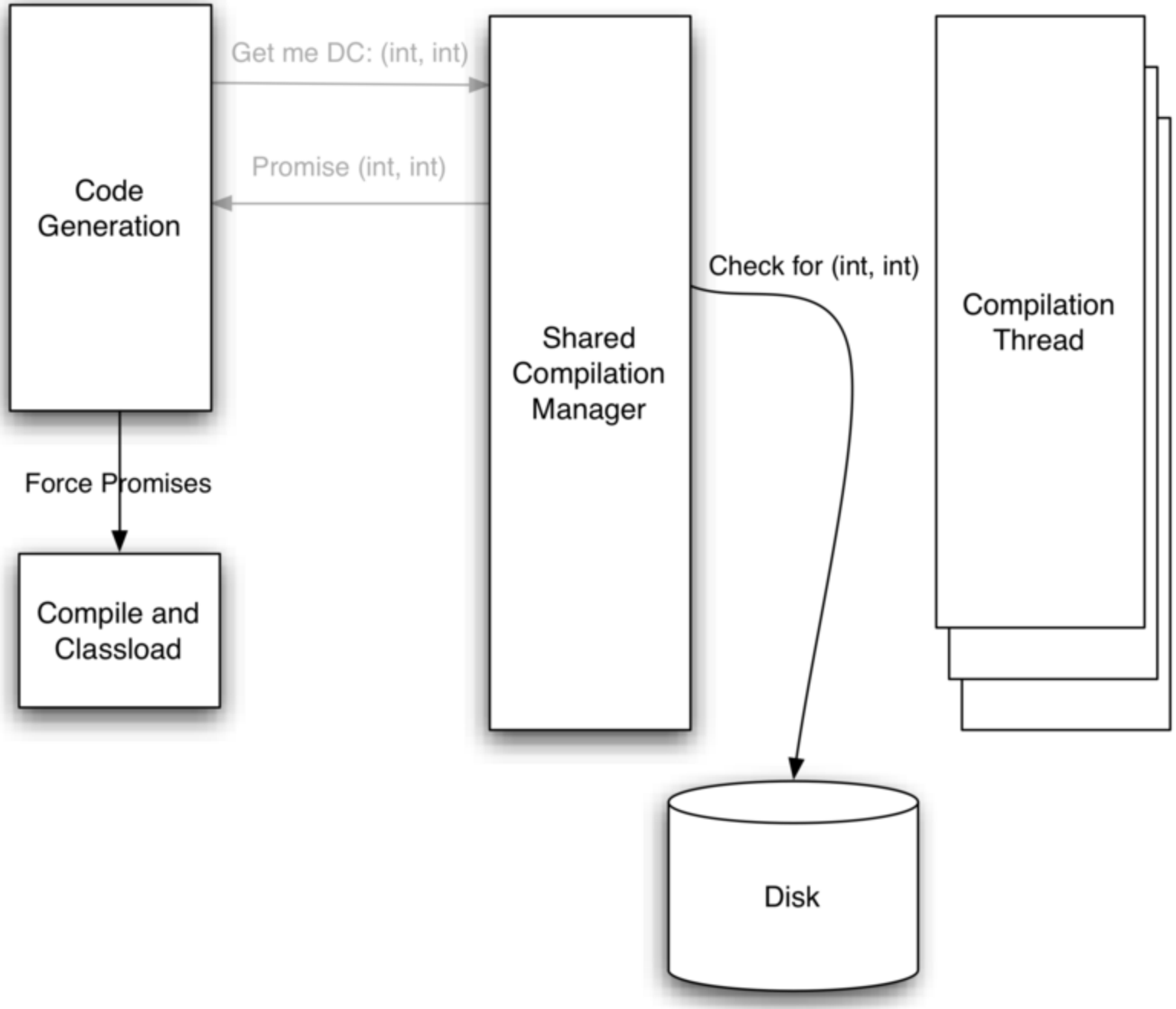
- Monomorphic call sites
- Unrolled loops
- Avoids garbage
- Lengthens compile time
- Can lead to massive methods

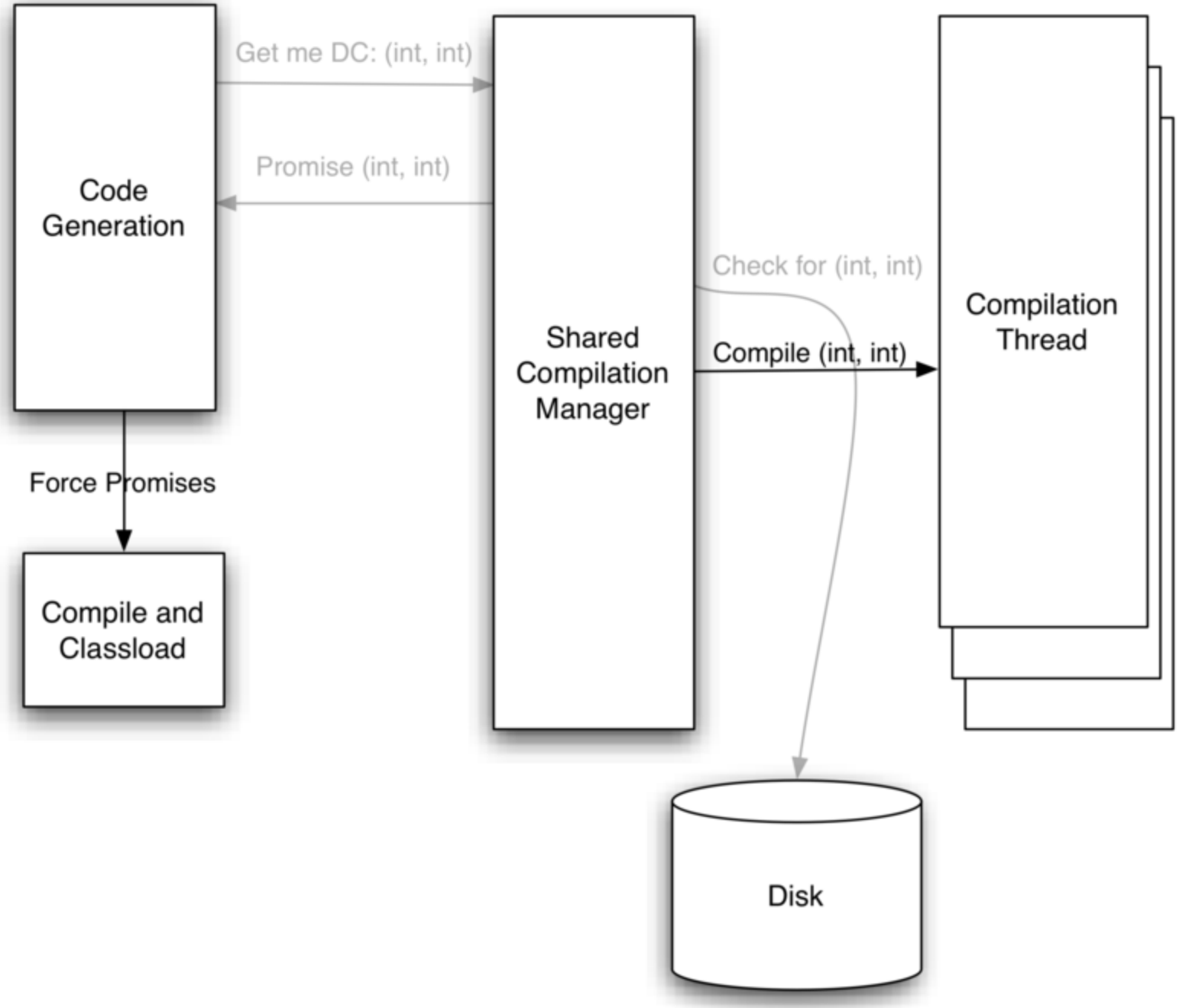
## Lessons Learned

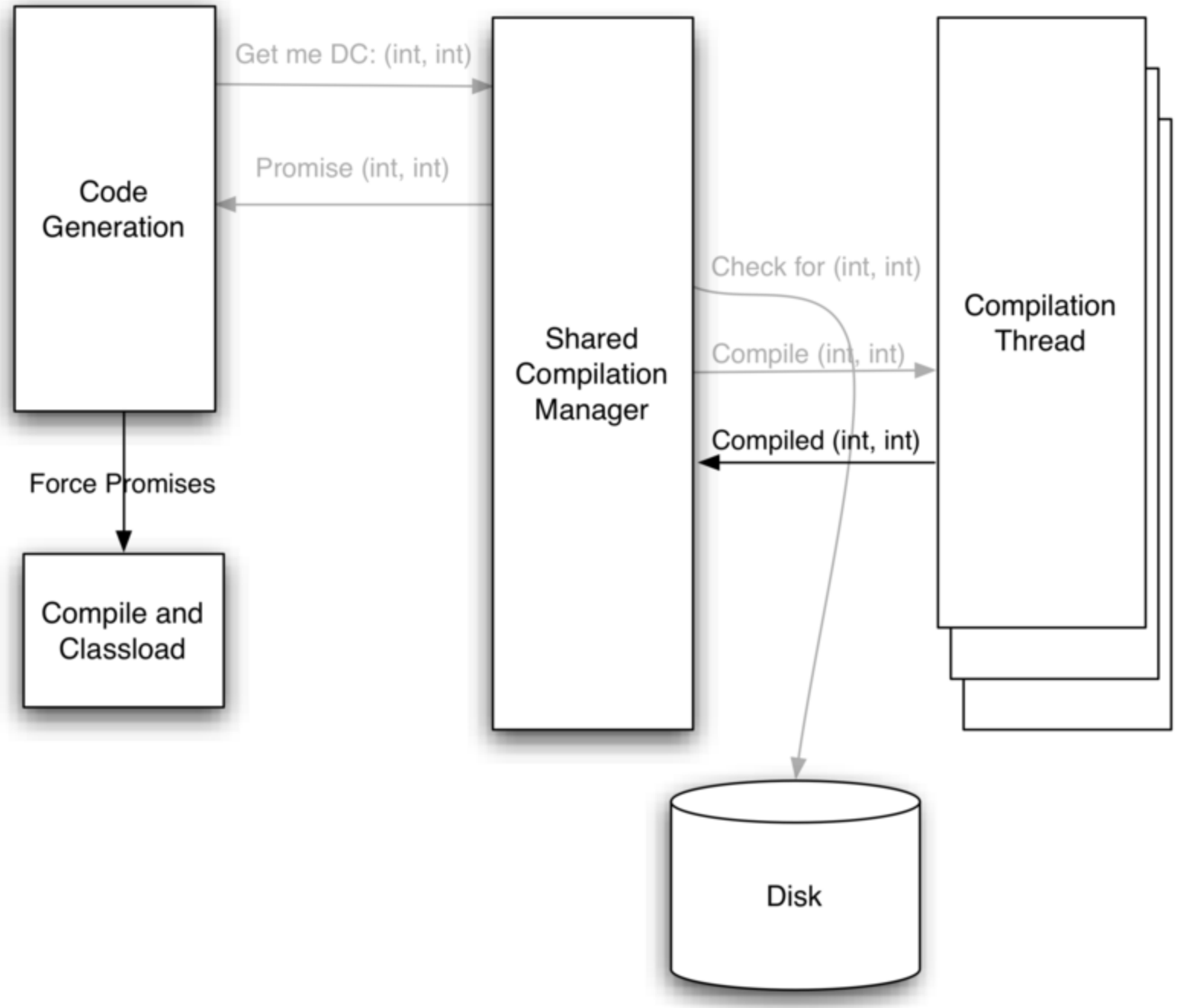
- Spaghetti code is bad; spaghetti code generation is worse
- Choose between generated code vs runtime library
- Parallel APIs for generating code and for runtime code
- Type specialization makes code fast
- **Separate compilation requires planning**
- Classloaders hurt the unwary
- Improve tooling

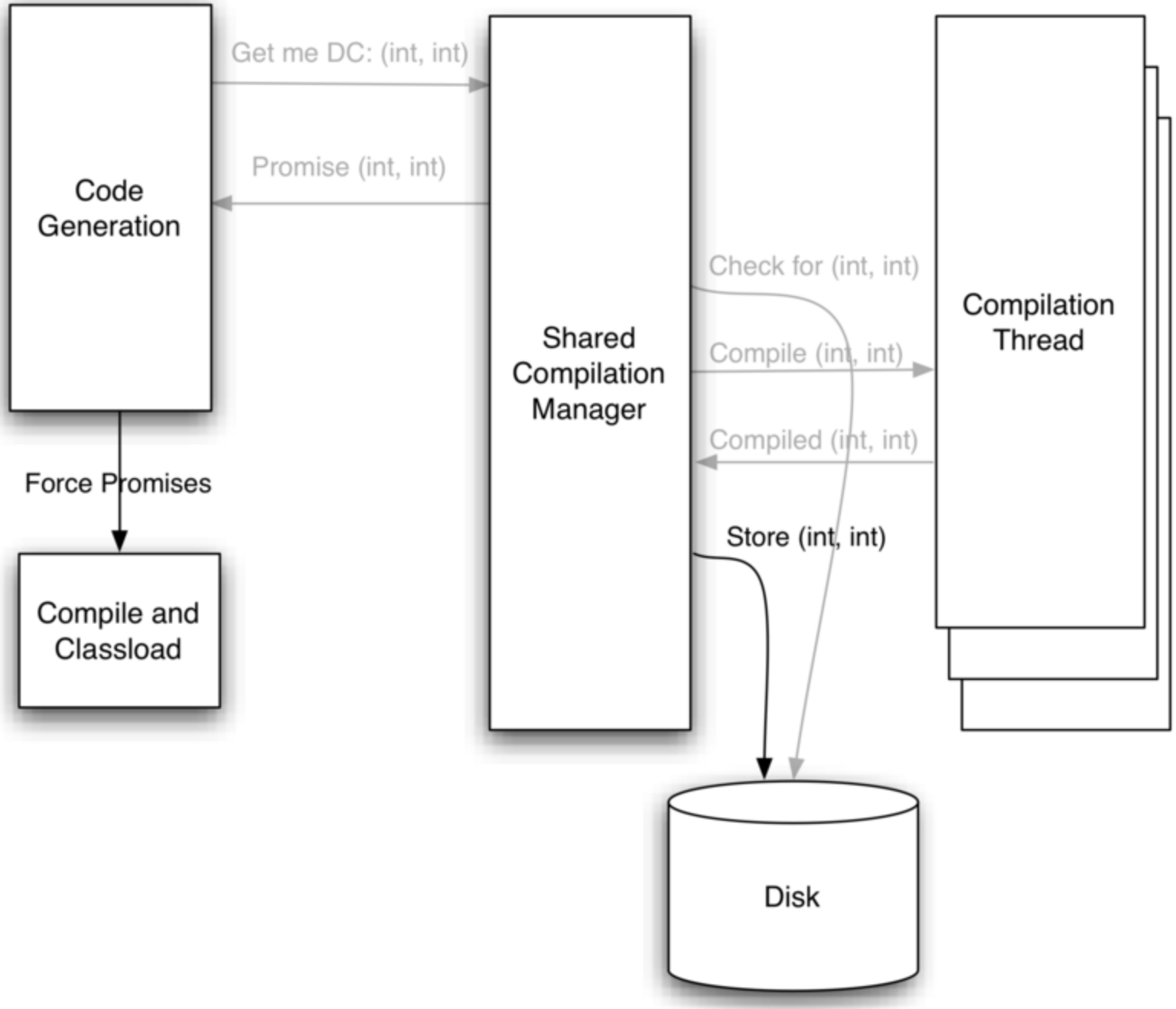

Code Generation
Get me DC: (int, int)
Shared Compilation Manager
Compilation Thread
Force Promises
Compile and Classload
Disk

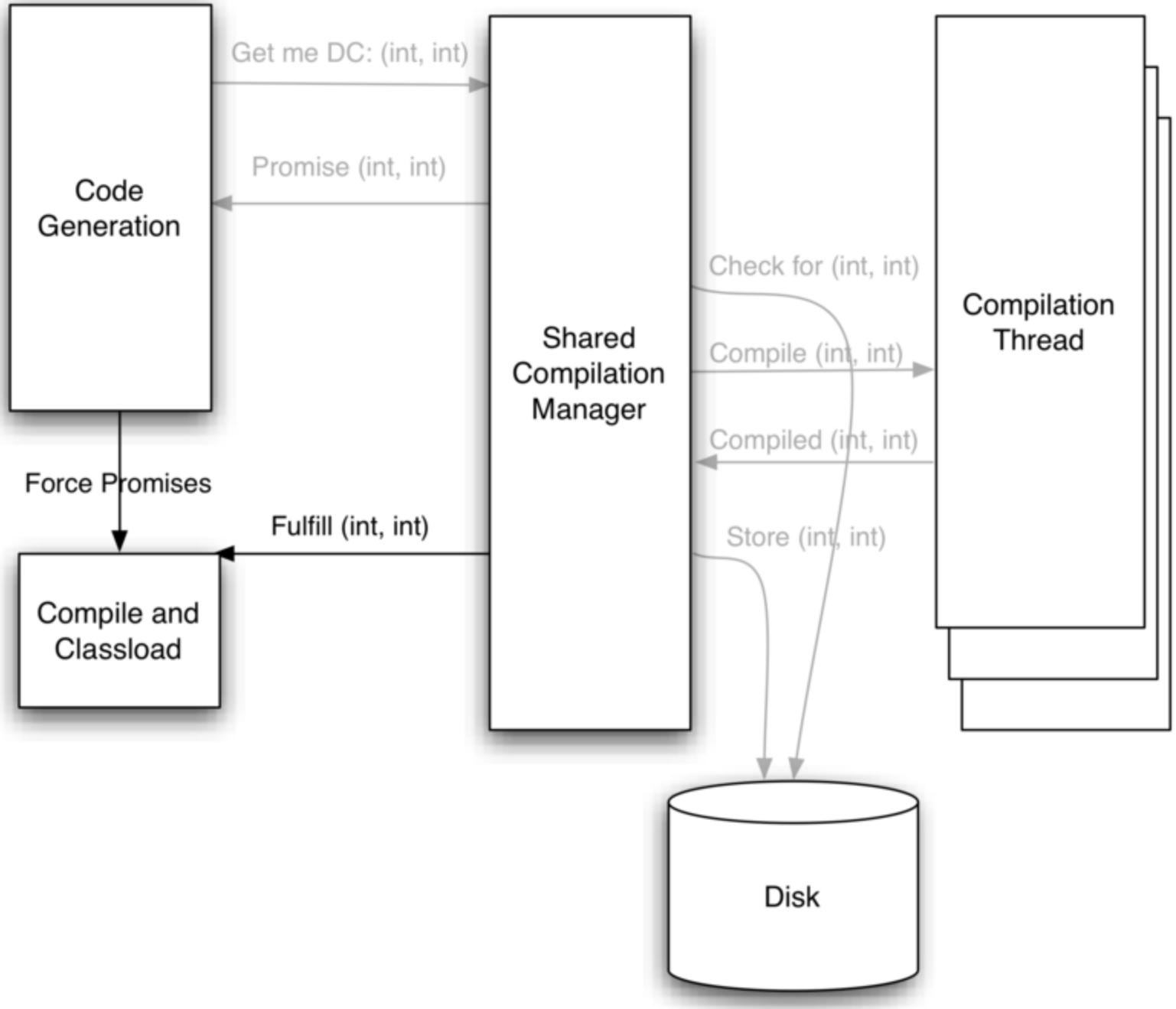

Code Generation
Get me DC: (int, int)
Promise (int, int)
Force Promises
Compile and Classload
Shared Compilation Manager
Compilation Thread
Disk


Code Generation
Get me DC: (int, int)
Promise (int, int)
Force Promises
Compile and Classload
Shared Compilation Manager
Check for (int, int)
Compilation Thread
Disk


Code Generation
Get me DC: (int, int)
Promise (int, int)
Force Promises
Compile and Classload
Shared Compilation Manager
Check for (int, int)
Compile (int, int)
Compilation Thread
Disk


Code Generation
Get me DC: (int, int)
Promise (int, int)
Force Promises
Compile and Classload
Shared Compilation Manager
Check for (int, int)
Compile (int, int)
Compiled (int, int)
Compilation Thread
Disk


Code Generation
Get me DC: (int, int)
Promise (int, int)
Force Promises
Compile and Classload
Shared Compilation Manager
Check for (int, int)
Compile (int, int)
Compiled (int, int)
Store (int, int)
Compilation Thread
Disk

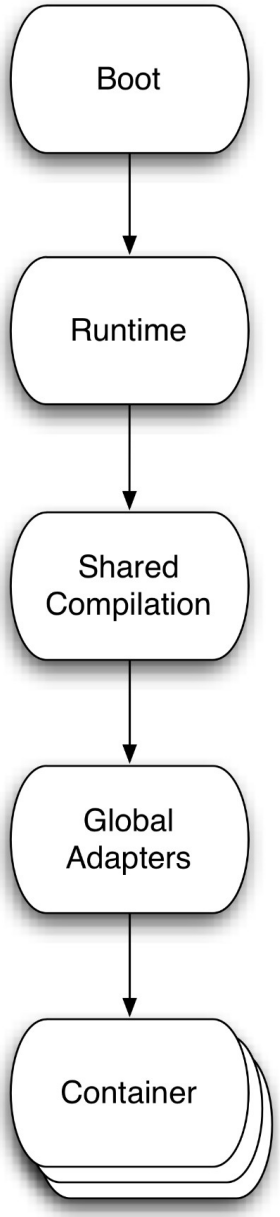

Code Generation
Get me DC: (int, int)
Promise (int, int)
Force Promises
Fulfill (int, int)
Compile and Classload
Shared Compilation Manager
Check for (int, int)
Compile (int, int)
Compiled (int, int)
Store (int, int)
Compilation Thread
Disk

## Lessons Learned

- Spaghetti code is bad; spaghetti code generation is worse
- Choose between generated code vs runtime library
- Parallel APIs for generating code and for runtime code
- Type specialization makes code fast
- Separate compilation requires planning
- **Classloaders hurt the unwary**
- Improve tooling

# Classloaders

- Sharing
- Lifecycle
- MS Windows



## Lessons Learned

- Spaghetti code is bad; spaghetti code generation is worse
- Choose between generated code vs runtime library
- Parallel APIs for generating code and for runtime code
- Type specialization makes code fast
- Separate compilation requires planning
- Classloaders hurt the unwary
- **Improve tooling**

## Shameless Plugs

- StreamBase
  - You could build one of these yourself, or use ours
  - Download, Buy, OEM
  - We're hiring
- DEBS – Distributed Event Based Systems
  - Academic (ACM) Conference: Arlington TX in July 2013
- EPTS – Event Processing Technology Society
  - http://ep-ts.org, industry consortium
- IDDS – International Development Design Summit
  - http://iddsummit.org/


# Questions?

Download StreamBase and More Information

http://www.streambase.com