Pivot Tracing

Jonathan Mace, Ryan Roelke, Rodrigo Fonseca
Brown University

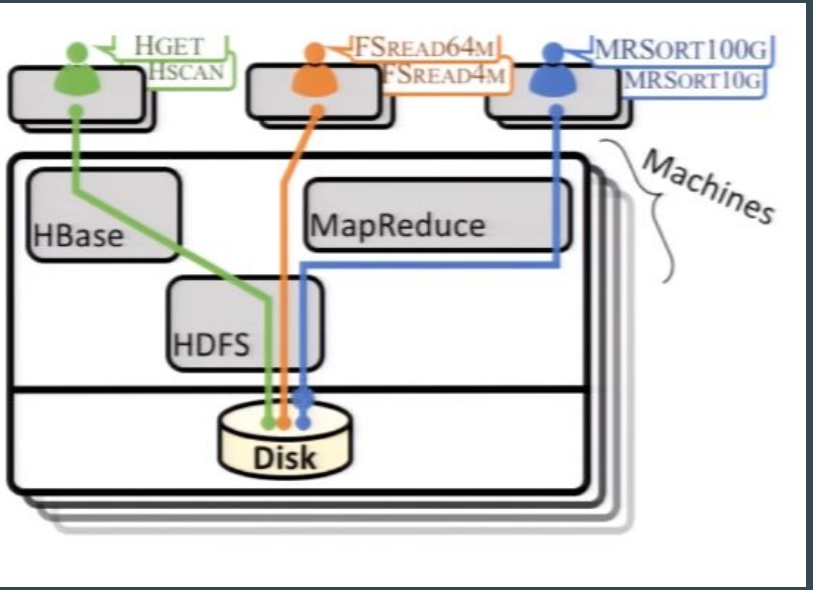
Problem

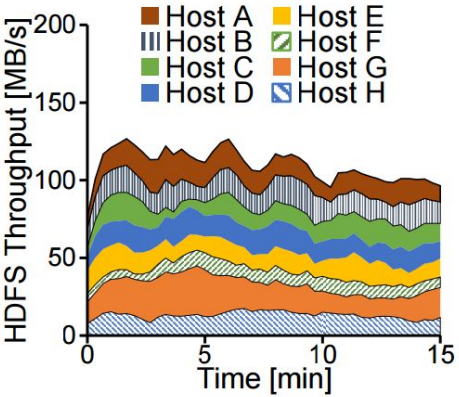
It's often difficult to judge what instrumentation should be implemented at development time

Debug logs can lack the information needed for your issue and/or have a lot of irrelevant information

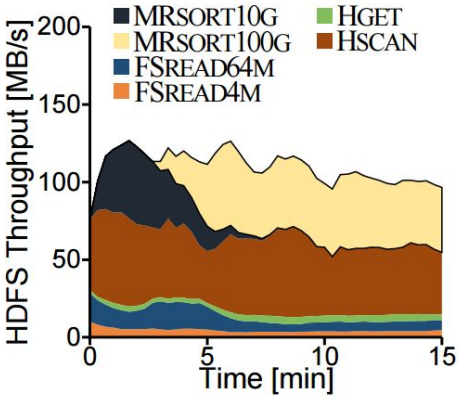
Configuration of nodes might be dynamic

It would be nice to be able to issue different queries for what to monitor at runtime and trace the path of execution





(a) HDFS DataNode throughput per machine from instrumented `DataNodeMetrics`.

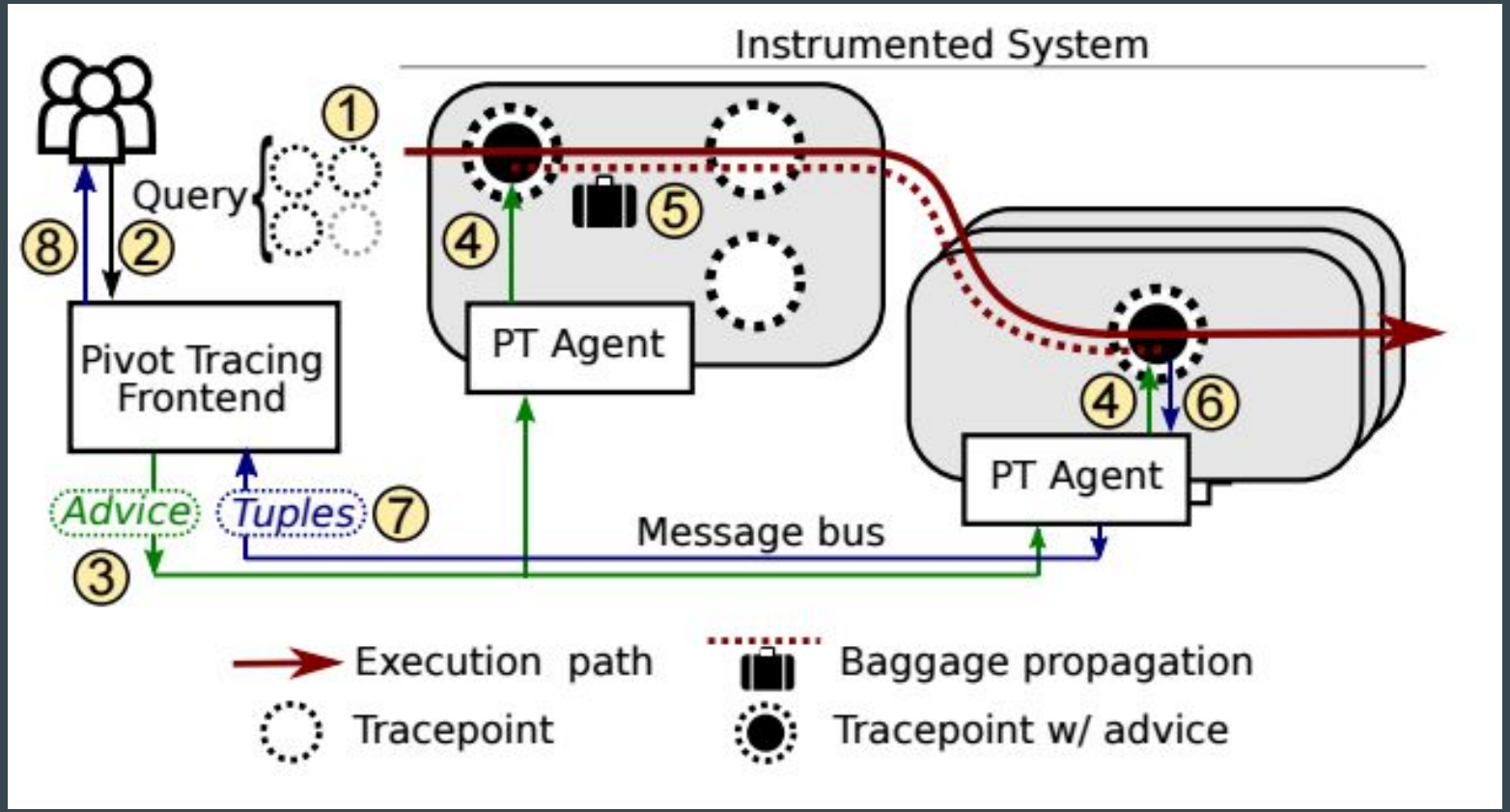


(b) HDFS DataNode throughput grouped by high-level client application.

Solution

Piggyback a tuple onto each message containing relevant information such as hostname or process id and propagate the tuple throughout execution

Issue novel happened-before join queries at runtime which are evaluated by agent-threads in the processes


Instrumented System
Query
1
2
3
4
5
6
7
8
Pivot Tracing Frontend
PT Agent
PT Agent
Advice
Tuples
Message bus
Execution path
Baggage propagation
Tracepoint
Tracepoint w/ advice

```
A1: OBSERVE procName
    PACK-FIRST procName

A2: OBSERVE delta
    UNPACK procName
    EMIT procName, SUM(delta)
```

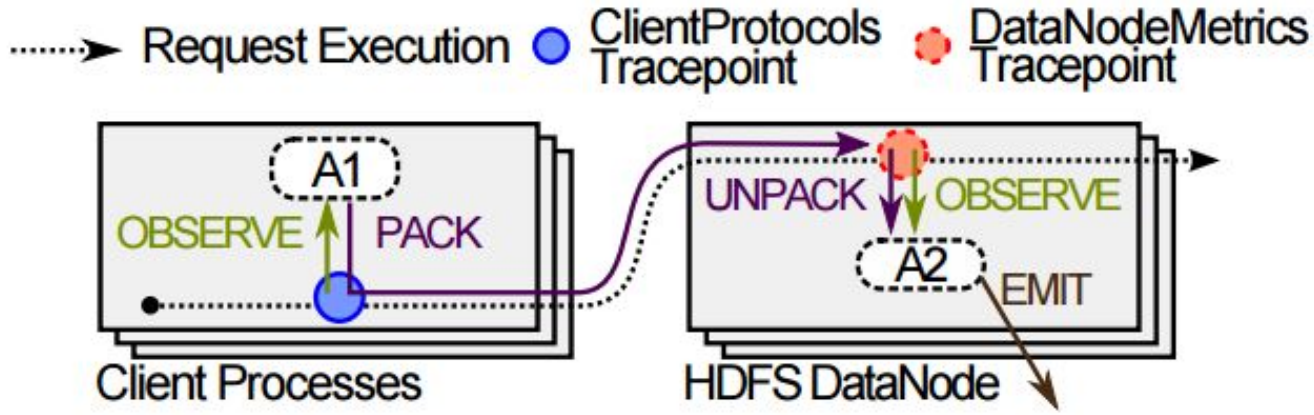


Figure 4: Advice generated for Q2 from §2: A1 observes and packs procName; A2 unpacks procName, observes delta, and emits (procName, SUM(delta))

Dynamic Instrumentation

“Java version 1.5 onwards supports dynamic method body rewriting via the java.lang.instrument package

The Pivot Tracing agent programmatically rewrites and reloads class bytecode from within the process using Javassist [44].”

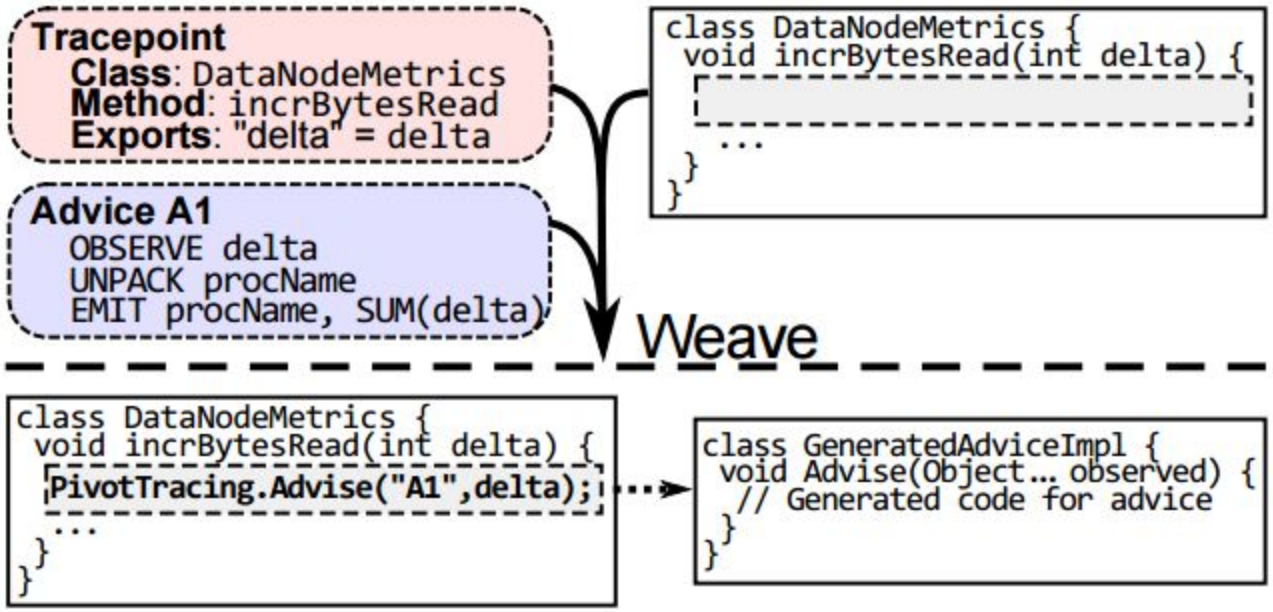


Figure 5: Advice for Q2 is *woven* at the `DataNodeMetrics` tracepoint. Variables exported by the tracepoint are passed when the advice is invoked.

Development Overhead?

“Pivot Tracing relies on developers to implement Baggage propagation when a request crosses thread, process, or asynchronous execution boundaries. In our experience, this entails adding a baggage field to existing application-level request contexts and RPC headers”

~50-200 LOC modification per system

Requires thorough knowledge of the code to know how to propagate the tuples

Performance Overhead?

“Pivot Tracing has a zero-probe effect: methods are unmodified by default, so tracepoints impose truly zero overhead until advice is woven into them.”

Baseline overhead 0.3% (no queries installed, only modifications to fields to include “baggage” and running PT Agent threads)

With queries from paper installed, maximum of 14.3% overhead

Among the queries tested, even the most sophisticated ones required a baggage size of less than 137 bytes

Performance Overhead? (cont.)

Although latency curve looks slightly super-linear, even with 256 tuples to pack, serialize, deserialize, and unpack, the total latency incurred was only 5 + 1 + 14 + 22 = 42 microseconds

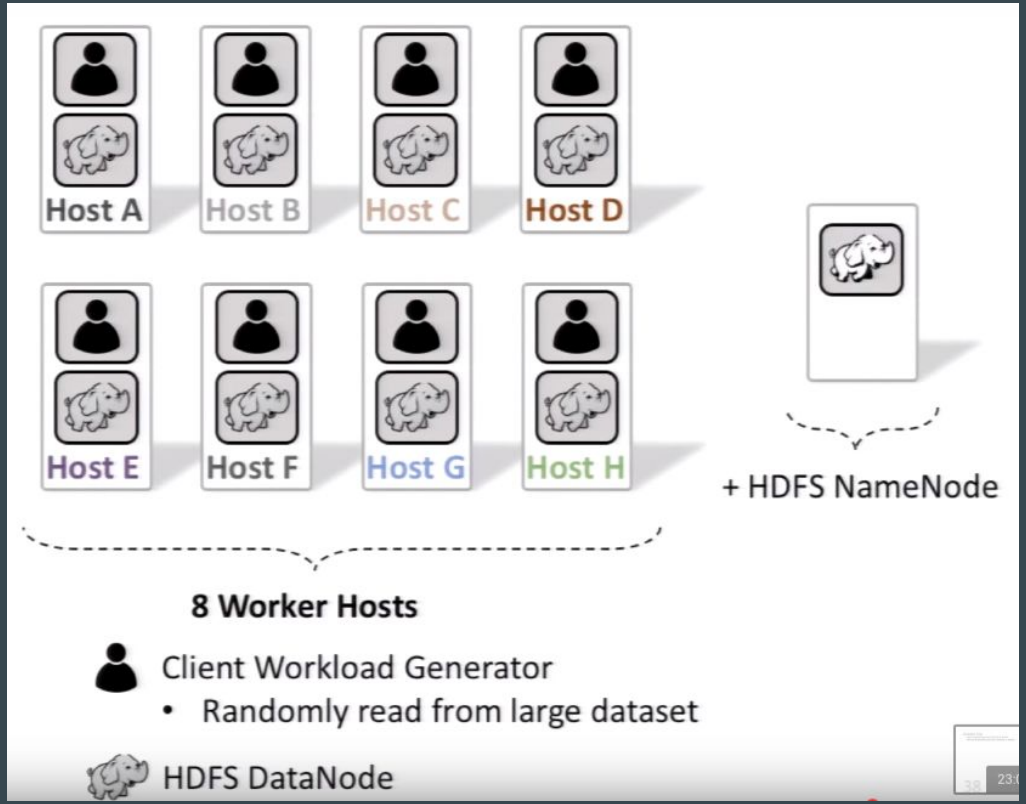
“The most noticeable overheads are incurred when propagating 60 tuples in the baggage, incurring 15.9% overhead for Open. Since this is a short CPU-bound request (involving a single read-only lookup), 16% is within reasonable expectations”

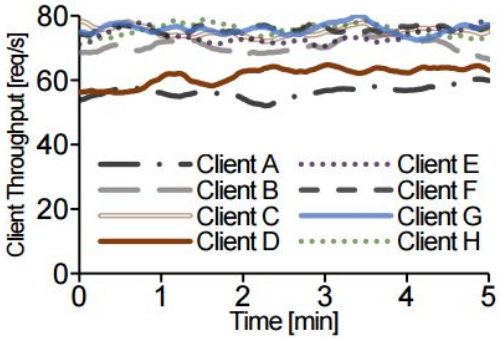
Performance Overhead? (cont.)

“JVM HotSwap requires Java’s debugging mode to be enabled, which causes come compiler optimizations to be disabled...[However], our HDFS throughput experiments above measure only a small overhead between debugging enabled and disabled”

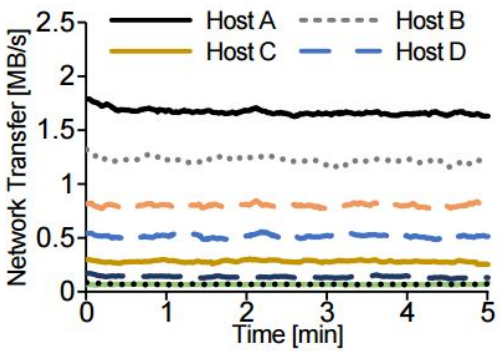
“Reloading a class with woven advice has a one-time cost of approximately 100ms, depending on the size of the class being reloaded”

Case Study: HDFS Replica Selection Bug

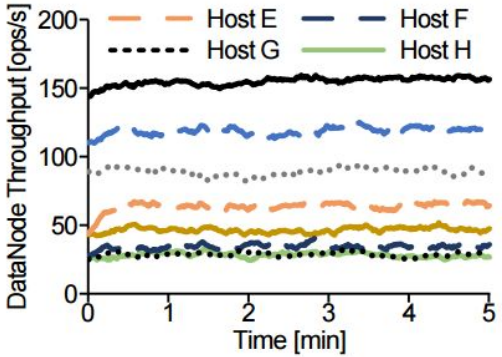




(a) Clients on Hosts A and D experience reduced workload throughput.

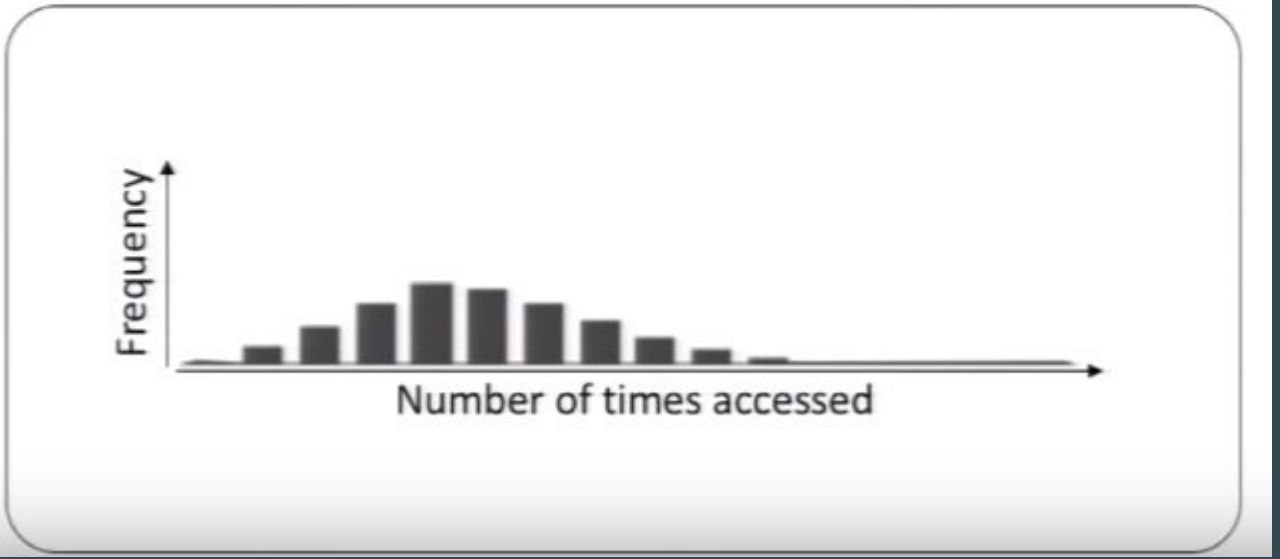


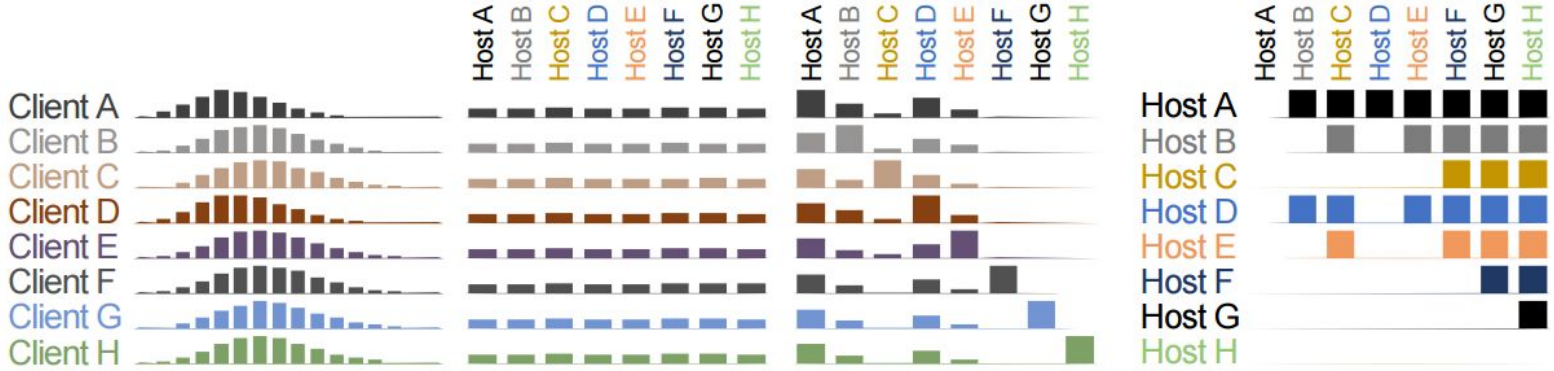
(b) Network transfer is skewed across machines.



(c) HDFS DataNode throughput is skewed across machines.

```
From blockLocations In NameNode.GetBlockLocations
  GroupBy blockLocations.fileName
  Select blockLocations.fileName, COUNT
```





(d) Observed HDFS file read distribution (row) per client (col).

(e) Frequency each client (row) sees each DataNode (col) as a replica location.

(f) Frequency each client (row) subsequently selects each DataNode (col).

(g) Observed frequency of choosing one replica host (row) over another (col)

Discussion

Dependent on Java?

Discussion

A user need to be very familiar with the code to use pivot tracing. He needs to know the exact name of the clients, processes and methods. Therefore, the tracing system can only be used by the developers, but not regular users?

Discussion

“In the worst case Pivot Tracing may need to pack an unbounded number of tuples in the baggage, one for each tracepoint invoked”

Not sure how significant the overhead would be when passing the ‘baggage’ objects in a large cluster (evaluation was only based on a 8 node cluster)

Discussion

Security vulnerabilities? Dynamically generating code and reloading it? (hot-swapping)

Discussion

More evaluation between existing systems?

Discussion

Possible to apply Pivot Tracing to wireless sensor networks (WSN)?

Conclusions

This is an important research problem: CS425 MPs gave me a deep appreciation for the difficulty of debugging even small and simple distributed systems, and any tools that could simplify the process is great