Function-based Scheduling

Experiment 1 + Methodology

Experiment 1

1.a) Using 1-4 keywords on a big core

- Energy avg: 0.55 J (A57), 0.045 J(A53)
- Service time*: 96 ms
- executeQueryPhase mean: 61ms (6887** ms)

2.a) Using 1-4 keywords on a little core

- Energy avg: 0.11 J (A57), 0.10 J (A53)
- Service time*: 295 ms
- executeQueryPhase mean: 220ms (7338** ms)

* *Measured by Loadgen

* ** Without excluding outliers (>5000)

Experiment 1

1.b) Using 8-14 keywords on a big core

- Energy avg: 0.73 J (A57), 0.046 J (A53)
- Service time*: 361 ms
- executeQueryPhase mean: 316 ms (5943 ms**)

2.b) Using 8-14 keywords on a little core

- Energy avg: 0.10 J (A57), 0.10 J (A53)
- Service time*: 1216 ms
- executeQueryPhase mean: 1142ms (3158 ms**)

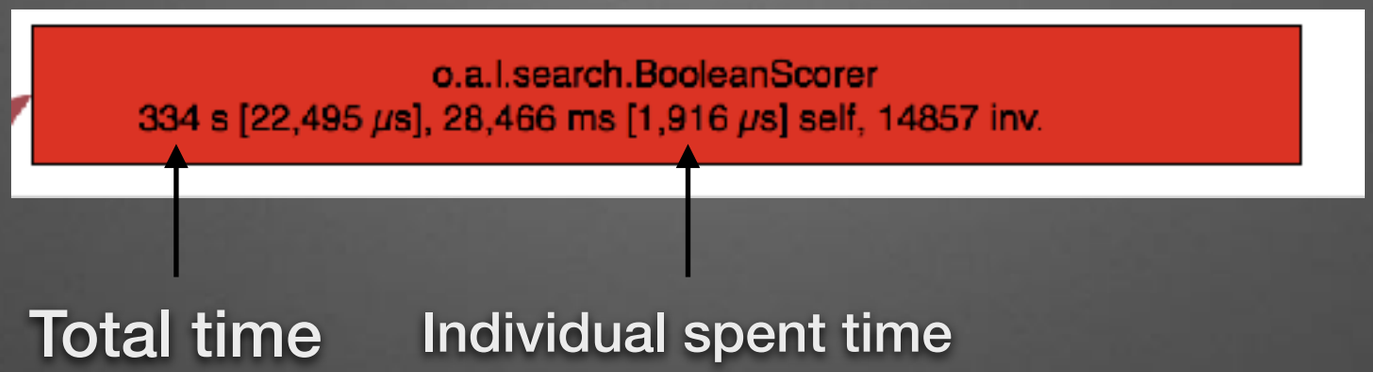
* *Measured by Loadgen

* ** Without excluding outliers (>5000)

Proposal for Identifying Hot function

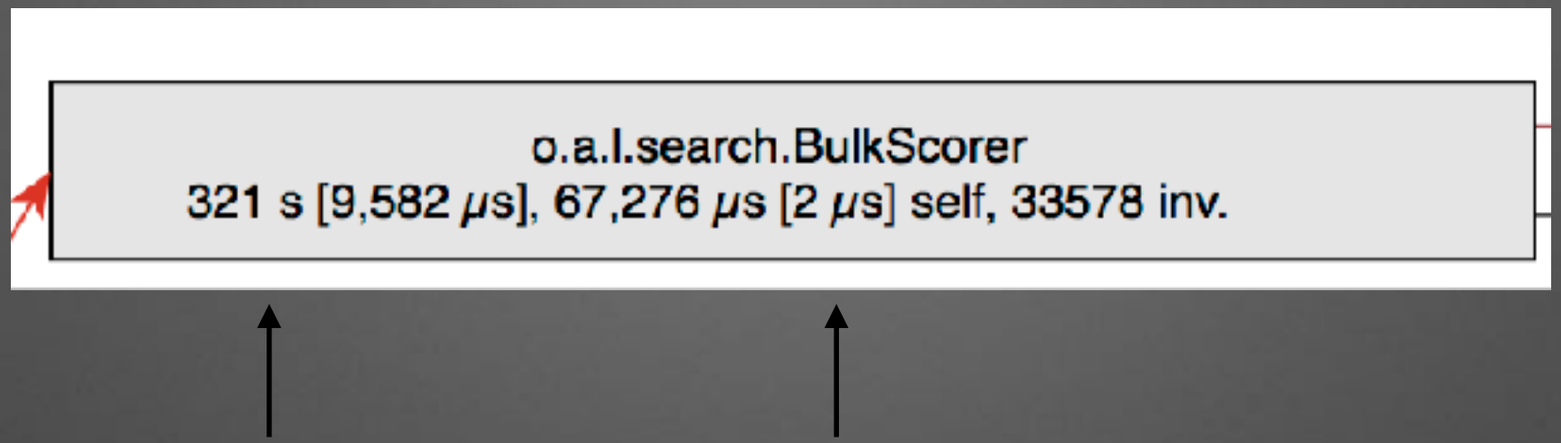
1. Get the function with the highest load (total spent time) of the application.
2. If individual spent time is lower than a certain value (overhead, 10microseconds), trace it back once.
3. If there are multiples tracebacks, then choose the one with highest individual spent time.
4. Repeat. Do it until the traceback reaches a function higher than the desired value.

Example: Elasticserch



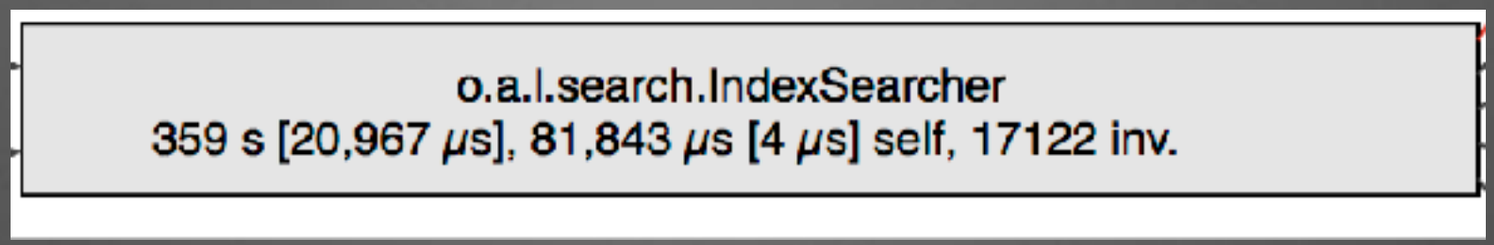
- That time might be not enough time to changing cores. We go back one step.

Example: Elasticserch



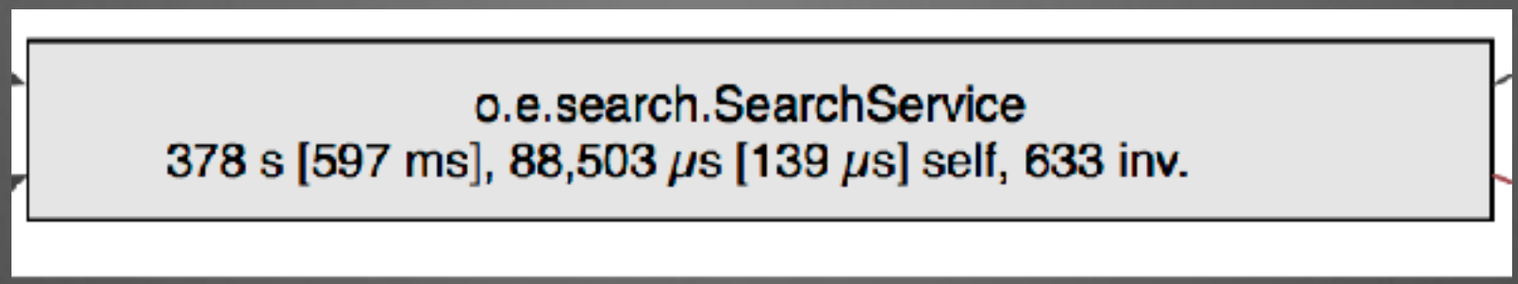
- The immediate back function of BooleanScorer might be not enough time to change cores (2us). We go back one step.

Example: Elasticserch



- The immediate back function of BulkScorer might be not enough time to change cores (4us). We go back one step.

Example: Elasticserch



- This function is higher than the desired overhead. So this is the one to be instrumented.

Challenges

1. Find a tool that can be adaptable for our needs.
2. Measure the changing time between cores.
3. (Also check consistency in Cassandra and Memcached)