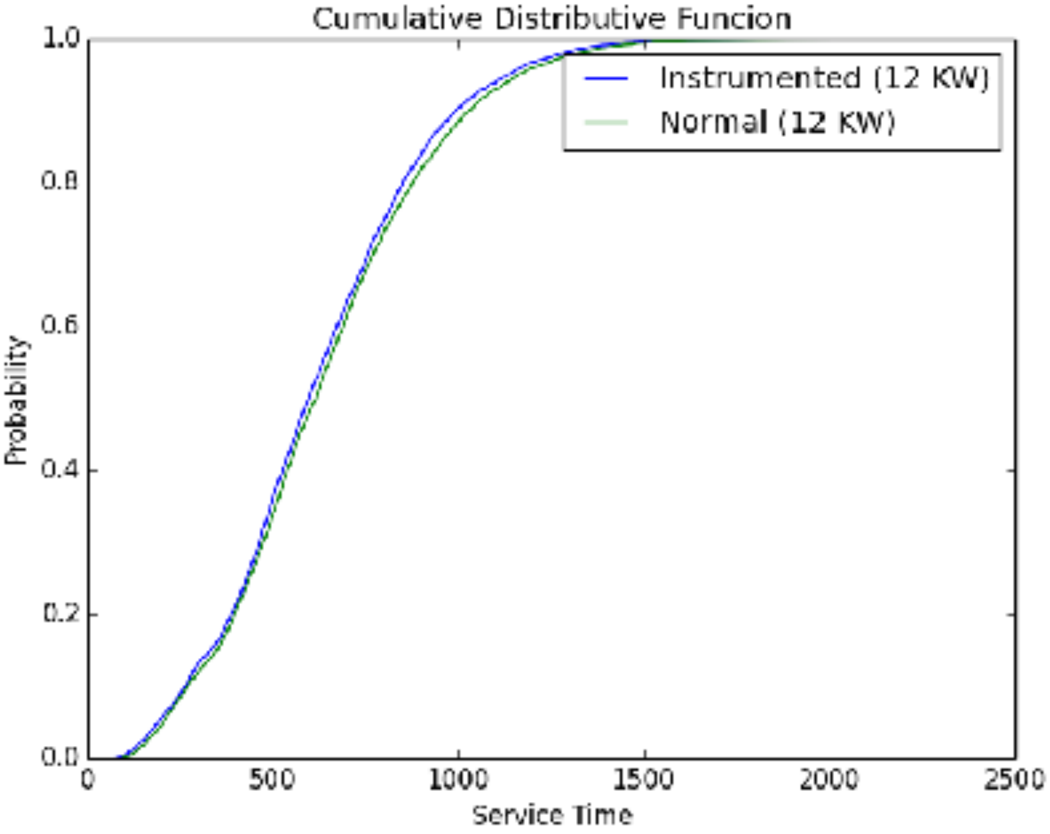
STATUS

The method consists in selecting a hot function and, whenever the thread access it, promote it to big core. After the thread exits the method, it goes back to small.

Results:



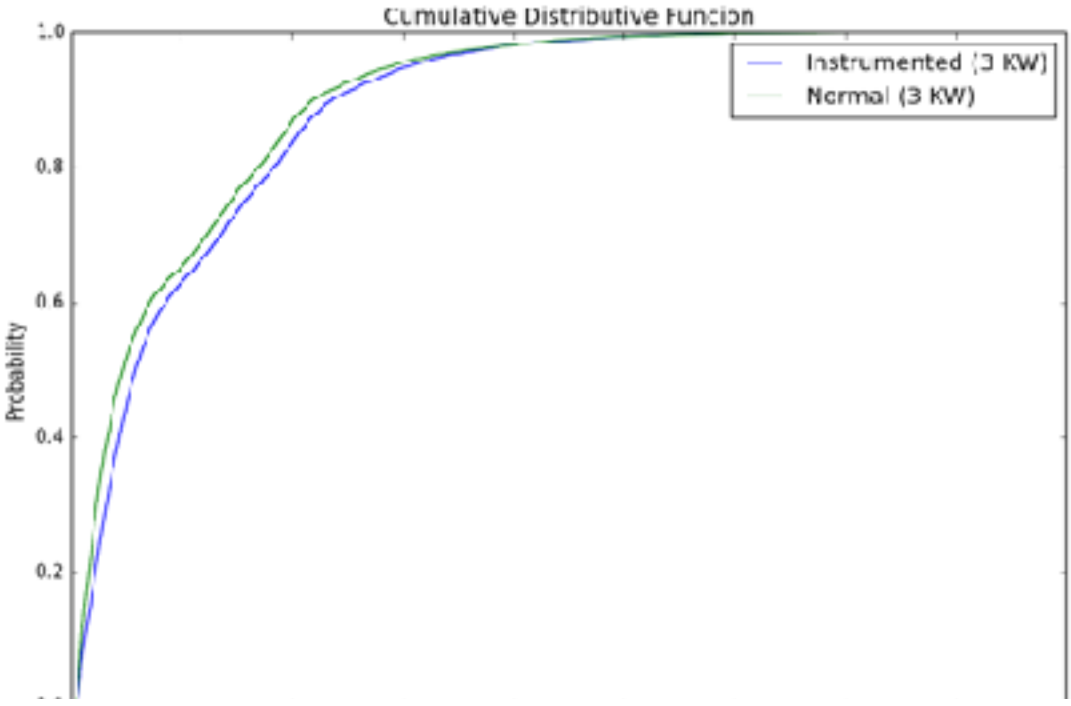
This CDF shows that only when the service time is big (mostly because the keyword length which AVERAGES 12 keywords), there's a difference. On 100%, the difference goes to about 20% (1931ms vs 2321ms).

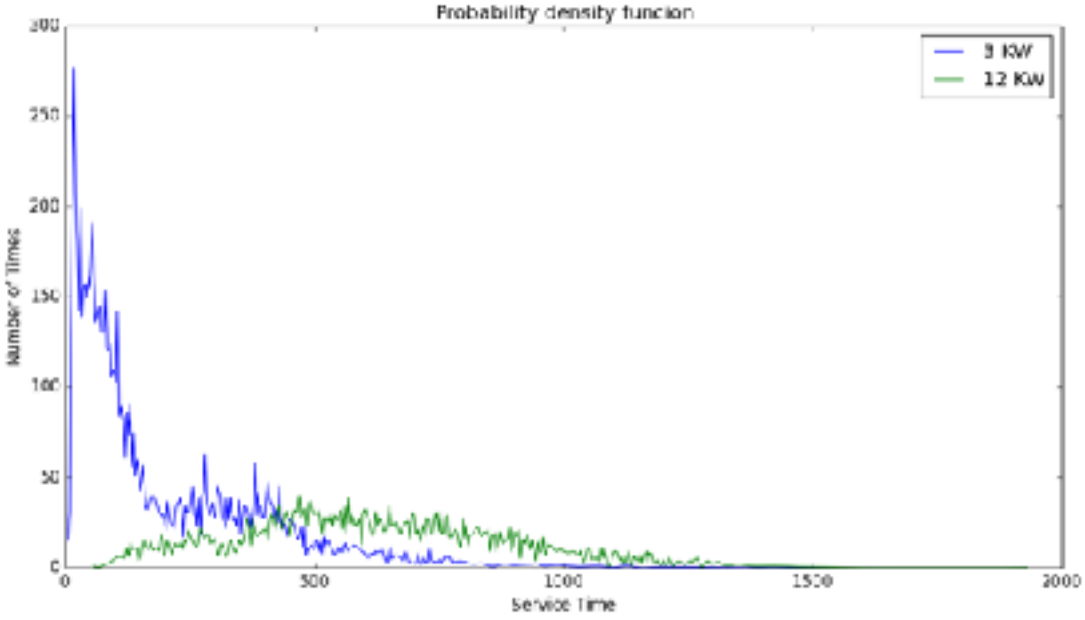
However, the graph below for small keyword length (3 keywords on average) shows that the instrumented version is slower than the normal version. By checking the PDF (also below) of service time, we see that many requests on small keywords length takes way less than 100ms to be done. Our hypothesis now is that the requests are so fast that the time to move the thread is higher than solving the thread itself. We're working on it.

So, for now:

- First, we're trying to change the procedure of how energy is measured. For this, we need to measure how much time the JVM takes to measure it. We can explain it in detail later.
- Another thing is that we're trying to use another function to instrument. As the actual one is "hot" and too fast for a number of requests, maybe the parent is slower and may be instrumented.

I think that if we manage to do it in Elasticsearch, Cassandra may be piece of cake.


Cumulative Distributive Funcion
1.0
0.8
0.6
0.4
0.2
Probability
Instrumented (3 KW)
Normal (3 KW)


Probability density funcion
300
250
200
150
100
50
0
Number of Times
0
500
1000
1500
2000
Service Time
3 KW
12 KW