Hyperscan in Rspamd

Harry Chang

Agenda

PCRE vs Hyperscan

Hsbench – benchmark tool in Hyperscan

Performance comparison

Hyperscan in Rspamd

- multi-pattern matching
- prefiltering
- serialize & deserialize
- performance change

PCRE vs Hyperscan

PCRE (Perl Compatible Regular Expressions)

- Slower
- Only block mode supported
- Compile 1 regex at a time
- Run same input data multi times for multi regexes

Hyperscan

- Faster
- Both block and streaming mode supported
- Compile multi regexes at a time
- Run same input data only once for multi regexes

hsbench

Hyperscan benchmark tool (from 4.4).

Usually we run:

```
    bin/hsbench -e <EXP> -s <SIG> -c <DB> -N –n 2000

    bin/hsbench -e <EXP> -z <ID> -c <DB> -N –n 2000
```

EXP: regex set, also called rules, patterns.

SIG: subset of expressions, multiple ids of expressions.

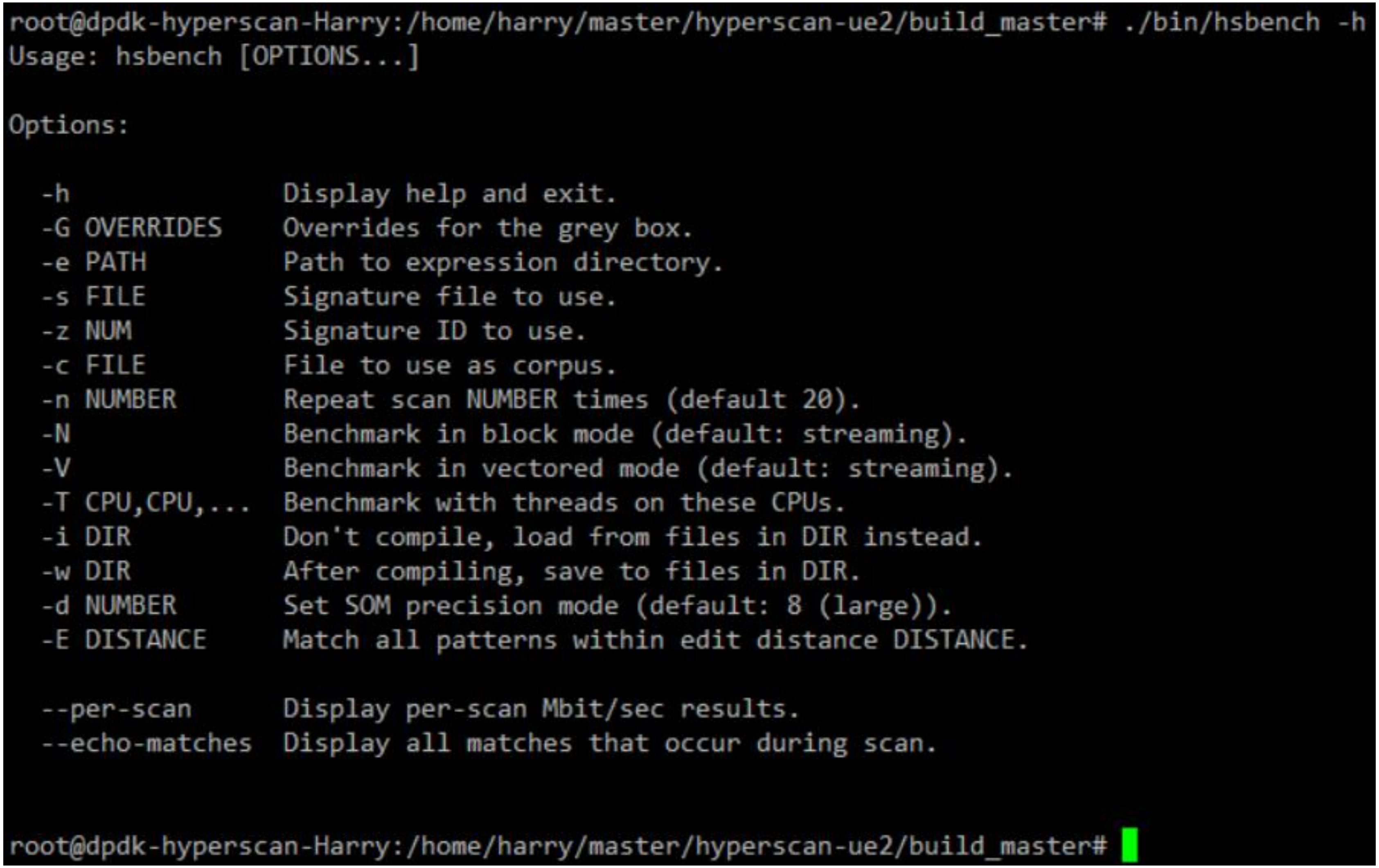
ID: single id of 1 expression.

DB: corpus, input data, in sqlite3 format.

-N: BLOCK mode, default is STREAMING mode.

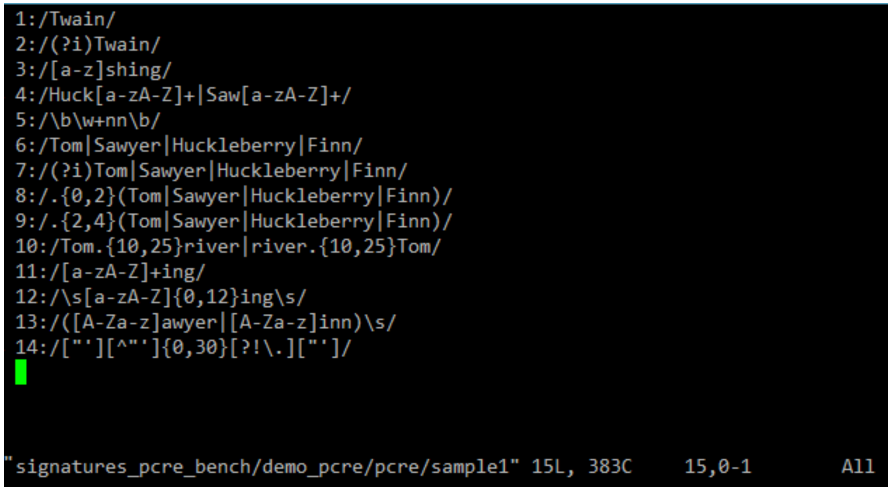
-n: repeats, bigger number of repeats can deliver more accurate results.

hsbench

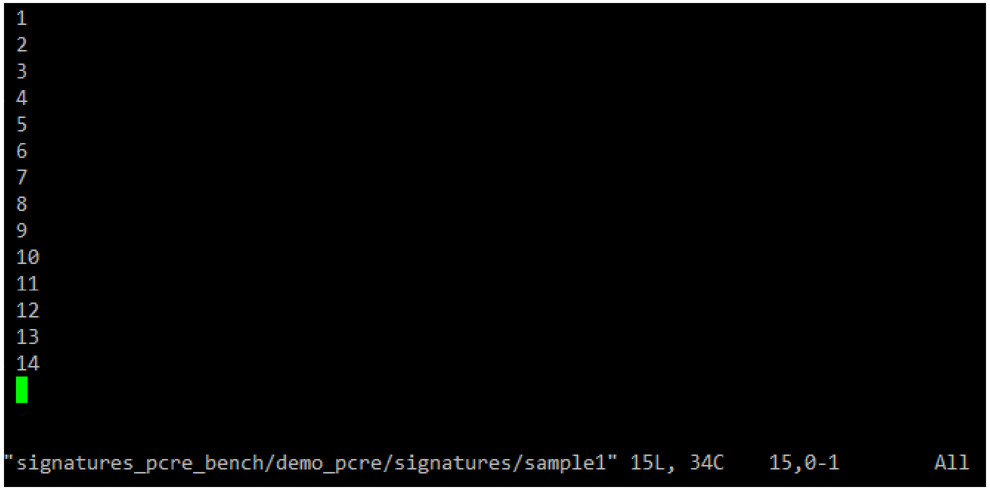


```
root@dpdk-hyperscan-Harry:/home/harry/master/hyperscan-ue2/build_master# ./bin/hsbench -h
Usage: hsbench [OPTIONS...]

Options:

  -h               Display help and exit.
  -G OVERRIDES     Overrides for the grey box.
  -e PATH          Path to expression directory.
  -s FILE          Signature file to use.
  -z NUM           Signature ID to use.
  -c FILE          File to use as corpus.
  -n NUMBER        Repeat scan NUMBER times (default 20).
  -N               Benchmark in block mode (default: streaming).
  -V               Benchmark in vectored mode (default: streaming).
  -T CPU,CPU,...   Benchmark with threads on these CPUs.
  -i DIR           Don't compile, load from files in DIR instead.
  -w DIR           After compiling, save to files in DIR.
  -d NUMBER        Set SOM precision mode (default: 8 (large)).
  -E DISTANCE      Match all patterns within edit distance DISTANCE.

  --per-scan       Display per-scan Mbit/sec results.
  --echo-matches   Display all matches that occur during scan.


root@dpdk-hyperscan-Harry:/home/harry/master/hyperscan-ue2/build_master# 
```

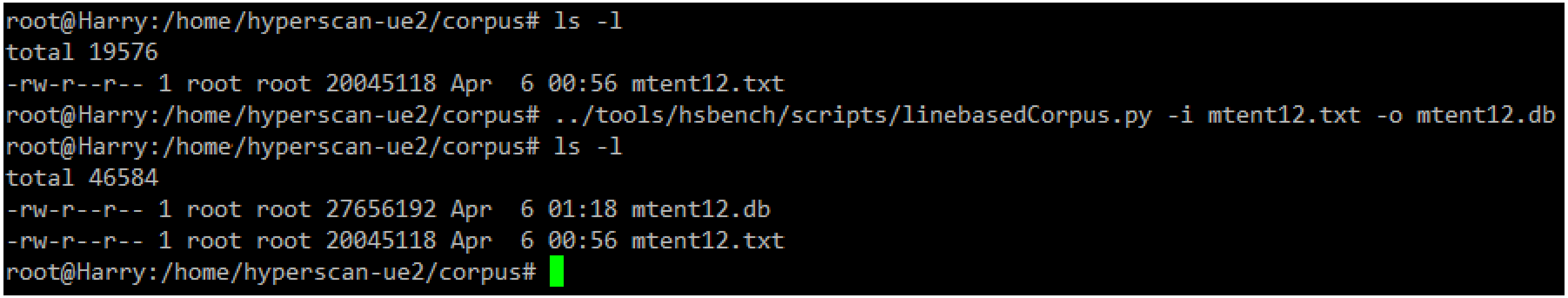
hsbench input



```
1:/Twain/
2:/(?i)Twain/
3:/[a-z]shing/
4:/Huck[a-zA-Z]+|Saw[a-zA-Z]+/
5:/\b\w+nn\b/
6:/Tom|Sawyer|Huckleberry|Finn/
7:/(?i)Tom|Sawyer|Huckleberry|Finn/
8:/.{0,2}(Tom|Sawyer|Huckleberry|Finn)/
9:/.{2,4}(Tom|Sawyer|Huckleberry|Finn)/
10:/Tom.{10,25}river|river.{10,25}Tom/
11:/[a-zA-Z]+ing/
12:/\s[a-zA-Z]{0,12}ing\s/
13:/([A-Za-z]awyer|[A-Za-z]inn)\s/
14:/["'][^"']{0,30}[?!\.]["']/


"signatures_pcre_bench/demo_pcre/pcre/sample1" 15L, 383C      15,0-1         All
```



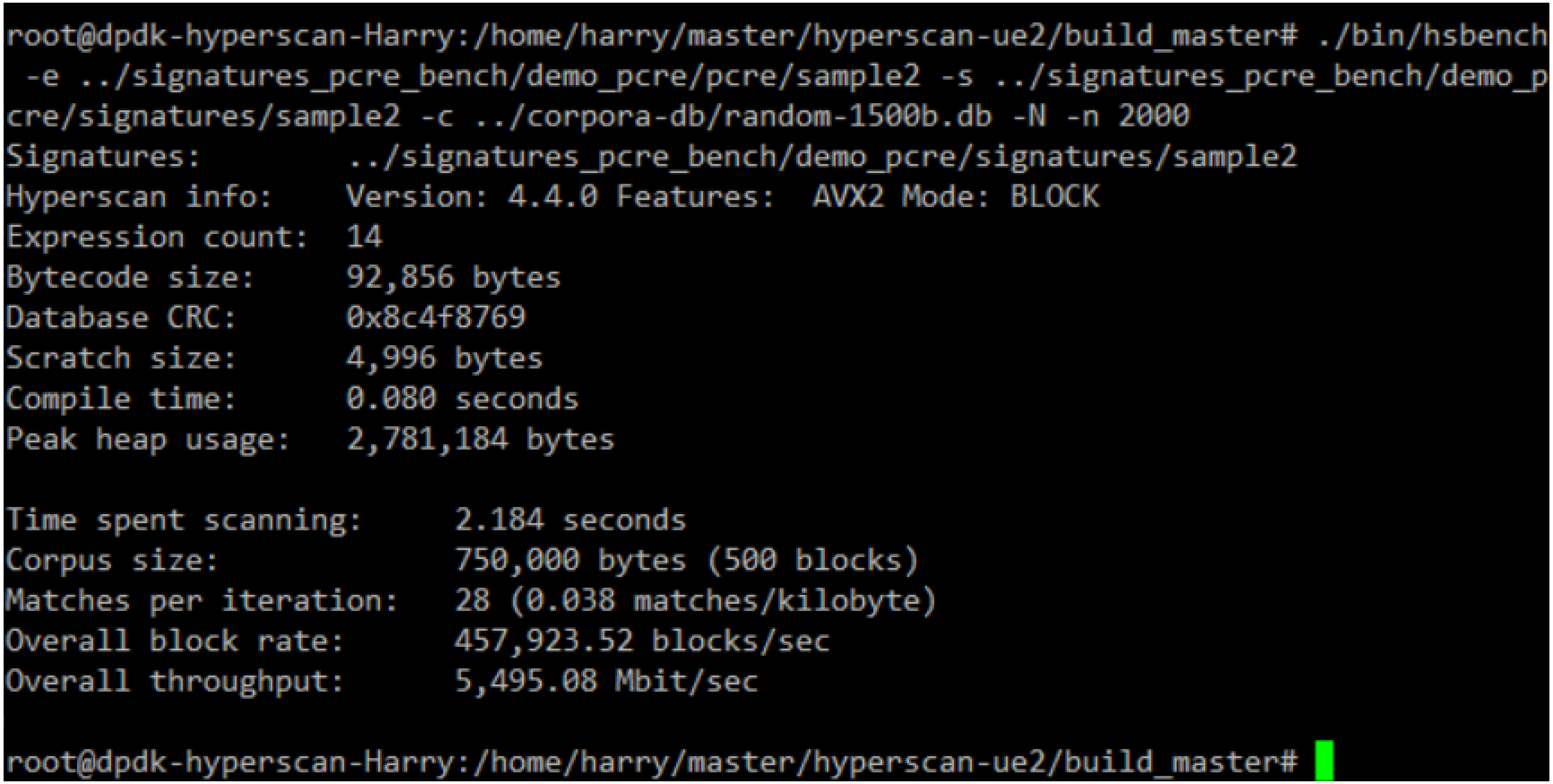
```
1
2
3
4
5
6
7
8
9
10
11
12
13
14


"signatures_pcre_bench/demo_pcre/signatures/sample1" 15L, 34C    15,0-1         All
```



```
root@Harry:/home/hyperscan-ue2/corpus# ls -l
total 19576
-rw-r--r-- 1 root root 20045118 Apr  6 00:56 mtent12.txt
root@Harry:/home/hyperscan-ue2/corpus# ../tools/hsbench/scripts/linebasedCorpus.py -i mtent12.txt -o mtent12.db
root@Harry:/home/hyperscan-ue2/corpus# ls -l
total 46584
-rw-r--r-- 1 root root 27656192 Apr  6 01:18 mtent12.db
-rw-r--r-- 1 root root 20045118 Apr  6 00:56 mtent12.txt
root@Harry:/home/hyperscan-ue2/corpus#
```

hsbench output

# ./bin/hsbench -e e_sample2 -s s_sample2 -c random-1500b.db -N -n 2000



```
root@dpdk-hyperscan-Harry:/home/harry/master/hyperscan-ue2/build_master# ./bin/hsbench
 -e ../signatures_pcre_bench/demo_pcre/pcre/sample2 -s ../signatures_pcre_bench/demo_p
cre/signatures/sample2 -c ../corpora-db/random-1500b.db -N -n 2000
Signatures:        ../signatures_pcre_bench/demo_pcre/signatures/sample2
Hyperscan info:    Version: 4.4.0 Features:  AVX2 Mode: BLOCK
Expression count:  14
Bytecode size:     92,856 bytes
Database CRC:      0x8c4f8769
Scratch size:      4,996 bytes
Compile time:      0.080 seconds
Peak heap usage:   2,781,184 bytes

Time spent scanning:     2.184 seconds
Corpus size:             750,000 bytes (500 blocks)
Matches per iteration:   28 (0.038 matches/kilobyte)
Overall block rate:      457,923.52 blocks/sec
Overall throughput:      5,495.08 Mbit/sec

root@dpdk-hyperscan-Harry:/home/harry/master/hyperscan-ue2/build_master# 
```

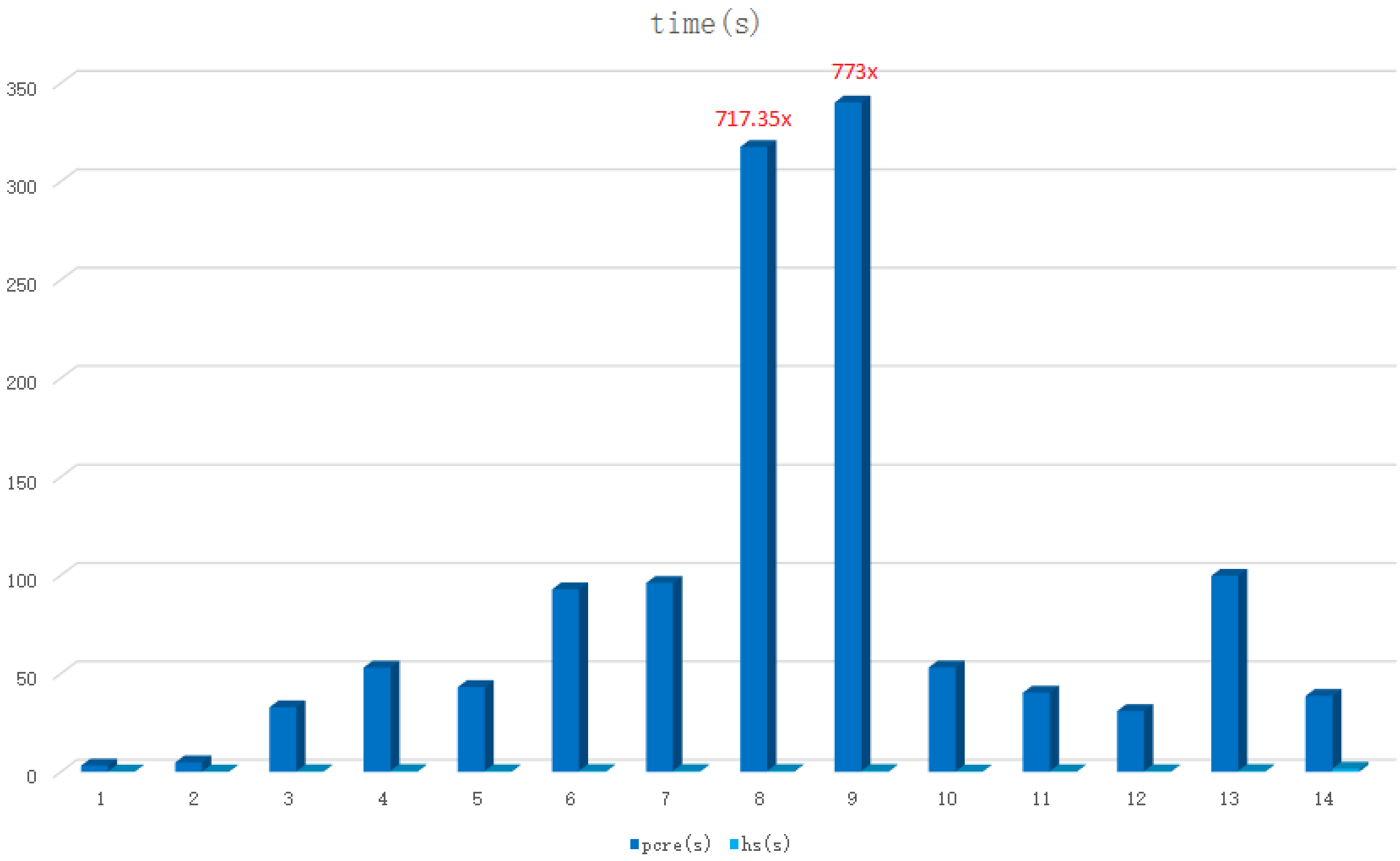
Performance comparison

Pick following 14 regexes for performance testing:

running on 1 core of Intel Xeon E5-2699 @ 2.30GHz, using 750000Byte random corpus, repeat 2000 times.

id	signature	pcre(s)	hs(s)	pcre->hs
1	Twain	3.175	0.19	16.73x
2	(?i)Twain	4.77	0.198	24.12x
3	[a-z]shing	32.778	0.276	118.6x
4	Huck[a-zA-Z]+|Saw[a-zA-Z]+	53.027	0.451	117.57x
5	\b\w+nn\b	43.099	0.291	148.14x
6	Tom|Sawyer|Huckleberry|Finn	92.815	0.445	208.45x
7	(?i)Tom|Sawyer|Huckleberry|Finn	96.017	0.448	214.41x
8	.{0,2}(Tom|Sawyer|Huckleberry|Finn)	317.573	0.443	717.35x
9	.{2,4}(Tom|Sawyer|Huckleberry|Finn)	340.121	0.44	773x
10	Tom.{10,25}river|river.{10,25}Tom	53.122	0.185	287.51x
11	[a-zA-Z]+ing	40.22	0.279	143.92x
12	\s[a-zA-Z]{0,12}ing\s	30.907	0.289	107.01x
13	([A-Za-z]awyer|[A-Za-z]inn)\s	99.669	0.437	202.86x
14	[″’][ˆ″’]{0,30}[?!\.][″’]	38.623	1.614	23.94x

Performance comparison



Multi-pattern matching

If matching rules 1 by 1.

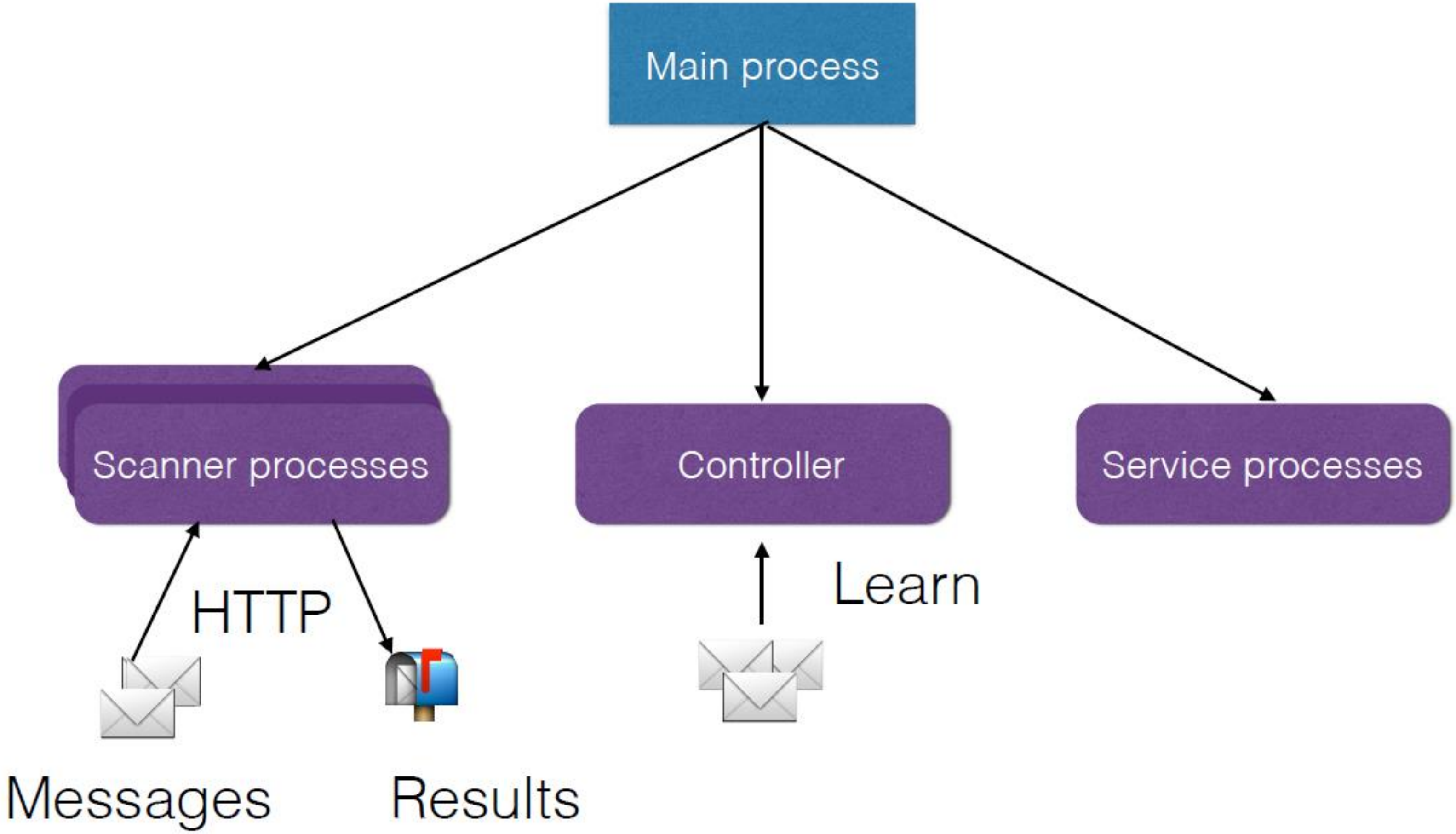
PCRE total scan time: 1245.916s

Hyperscan total scan time: 5.986s

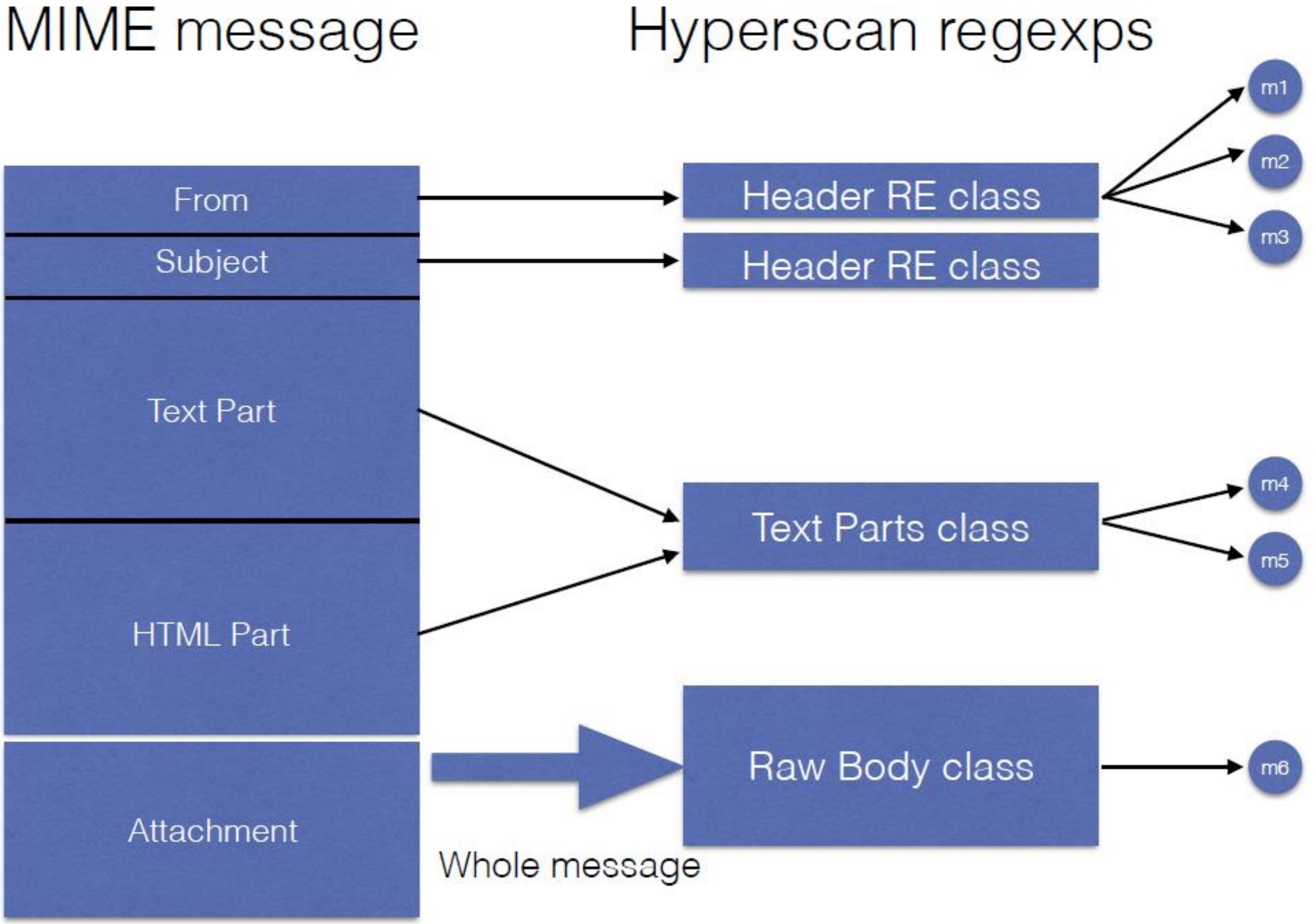
Hyperscan can compile and match all 14 rules just once, the performance will be even better.

Hyperscan scan time: 2.184s (< 5.986s)

Rspamd


Main process
Scanner processes
Controller
Service processes
HTTP
Learn
Messages
Results

MIME (Multipurpose Internet Mail Extensions)



Problem in Rspamd

Rspamd

spam filtering system

previously using PCRE for regex pattern matching

need to quickly process large regex set

Naïve sulotion: Join all expressions with '|'

/abc/ + /cde/ + /efg/ => /(abc)|(cde)|(efg)/

Not all regex sets can be merged by '|':

/foo.*bar/s	- single match
/[a-f]{6,10}/i	- case insensitive
/^GET\s.*HTTP/m	- multi-line

Hyperscan can compile all kinds of regex sets to one bytecode.

Hyperscan in Rspamd

Intergrated with Hyperscan:

Rspamd 1.0 -> Rspamd 1.1 (Jan 2016)

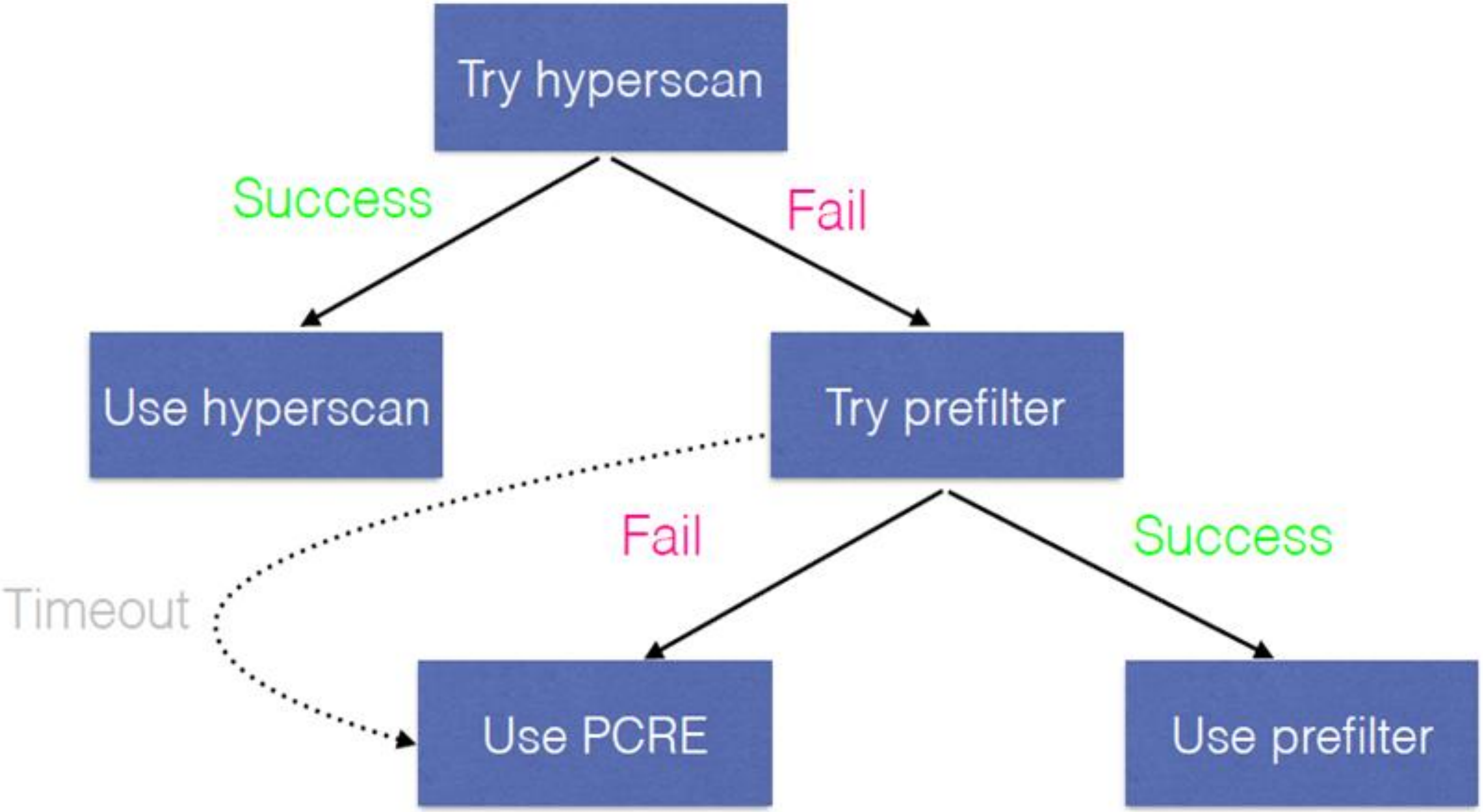
Key to this integration:

Multi-pattern matching, allowing Rspamd to quickly determine which of a large set of regex patterns match in a given message with a single scan call, rather than scanning for each pattern one at a time;

High performance, using Hyperscan's optimized regex engine;

Prefiltering support, which allows the use of Hyperscan as a quick prefilter even when the pattern makes use of PCRE constructs (such as backreferences) that Hyperscan does not natively support. If a prefiltered pattern produces matches, PCRE is used to confirm them**.**

Hyperscan(Prefilter) Compile in Rspamd



Try single pattern

Try Hyperscan

```
hs_compile(pats[i], flags[i], …)
```

Use Hyperscan

```
hs_pats[n] = pats[i]

hs_flags[n] = flags[i]

hs_ids[n++] = i
```

Try Prefilter

```
hs_compile(pats[i], flags[i] | HS_FLAG_PREFILTER, …)
```

Use Prefilter

```
hs_pats[n] = pats[i]

hs_flags[n] = flags[i] | HS_FLAG_PREFILTER

hs_ids[n++] = i
```

Compile & match multi-patterns

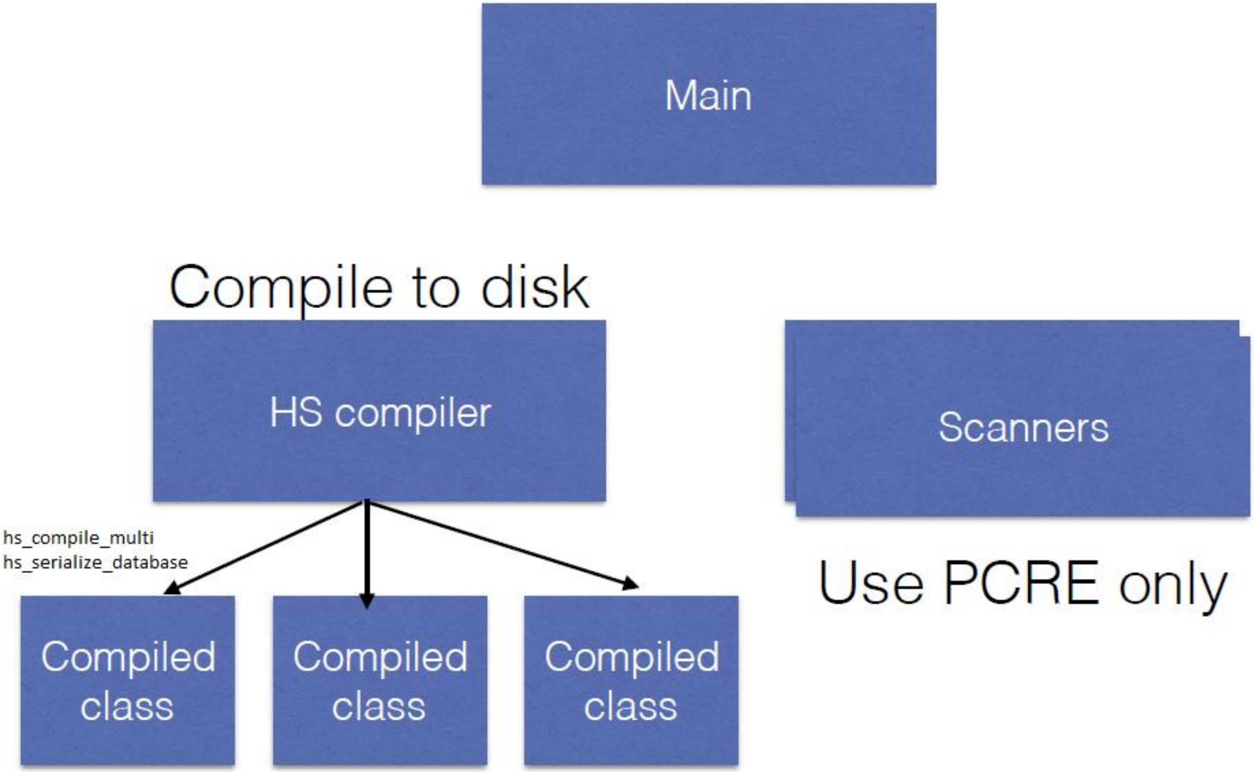
HS Compiler

```
hs_compile_multi(hs_pats, hs_flags, hs_ids, n, …, &hs_db, …)

hs_serialize_database(hs_db, &hs_serialized, &serialized_len)
```

Scanner

```
hs_deserialize_database(ptr, len, &hs_db)

hs_alloc_scratch(hs_db, &hs_scratch)

hs_scan(hs_db, data, len, 0, hs_scratch, cb, cb_data)
```

Hyperscan in Rspamd



Main

Compile to disk

HS compiler

Scanners

hs_compile_multi
hs_serialize_database

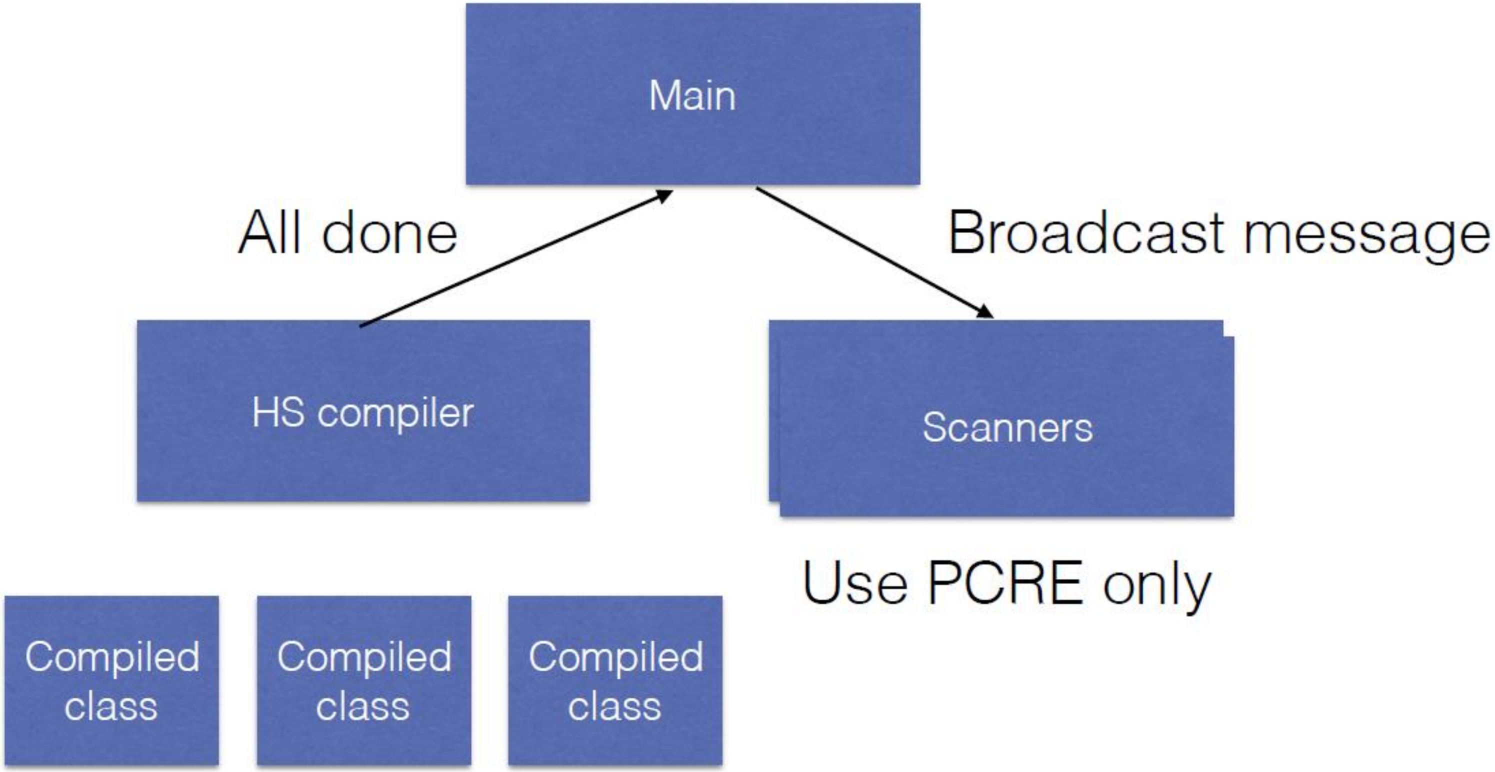
Compiled class

Compiled class

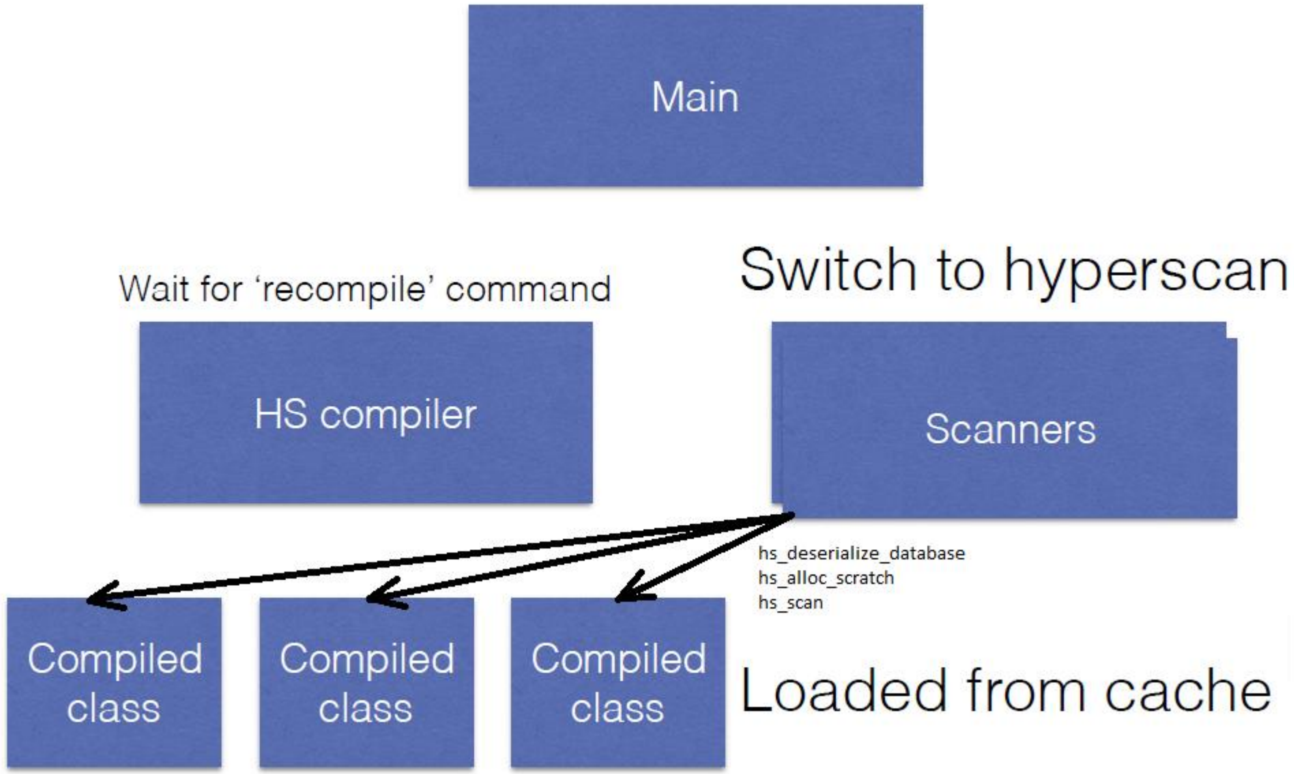
Compiled class

Use PCRE only

Hyperscan in Rspamd



Hyperscan in Rspamd



Main

Wait for ‘recompile’ command

HS compiler

Switch to hyperscan

Scanners

hs_deserialize_database
hs_alloc_scratch
hs_scan

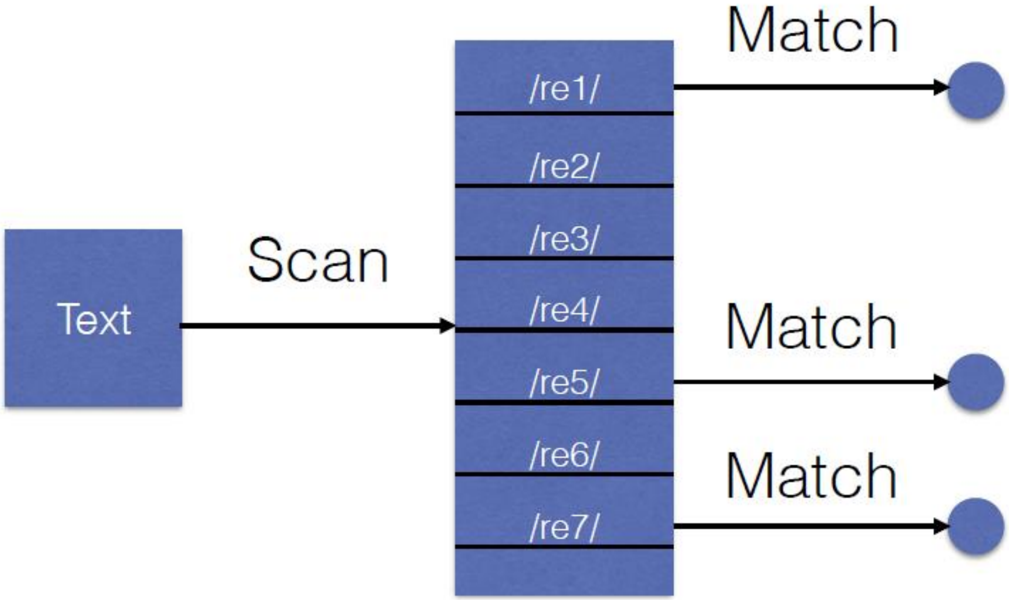
Compiled class

Compiled class

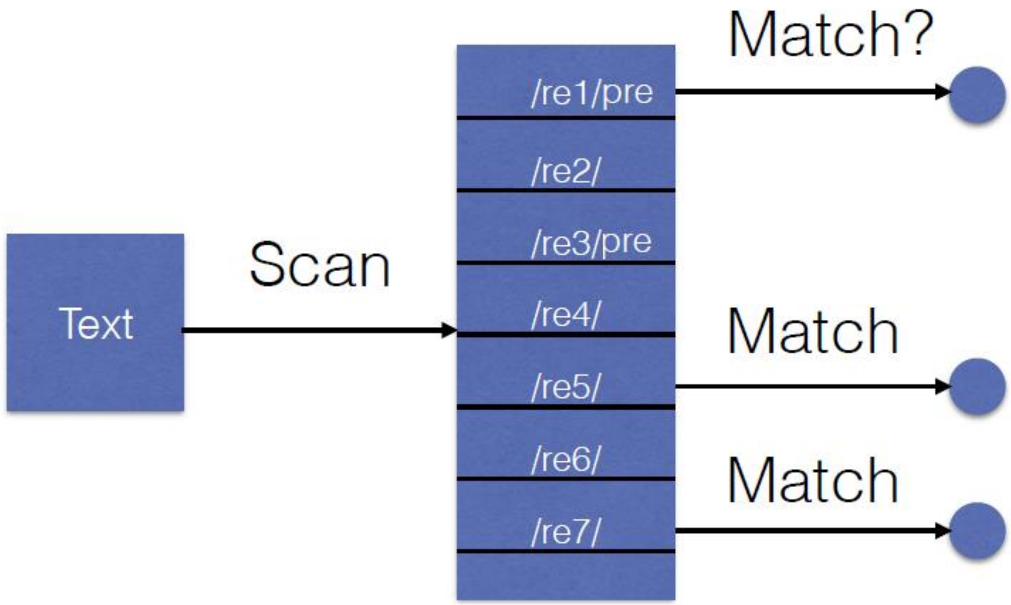
Compiled class

Loaded from cache

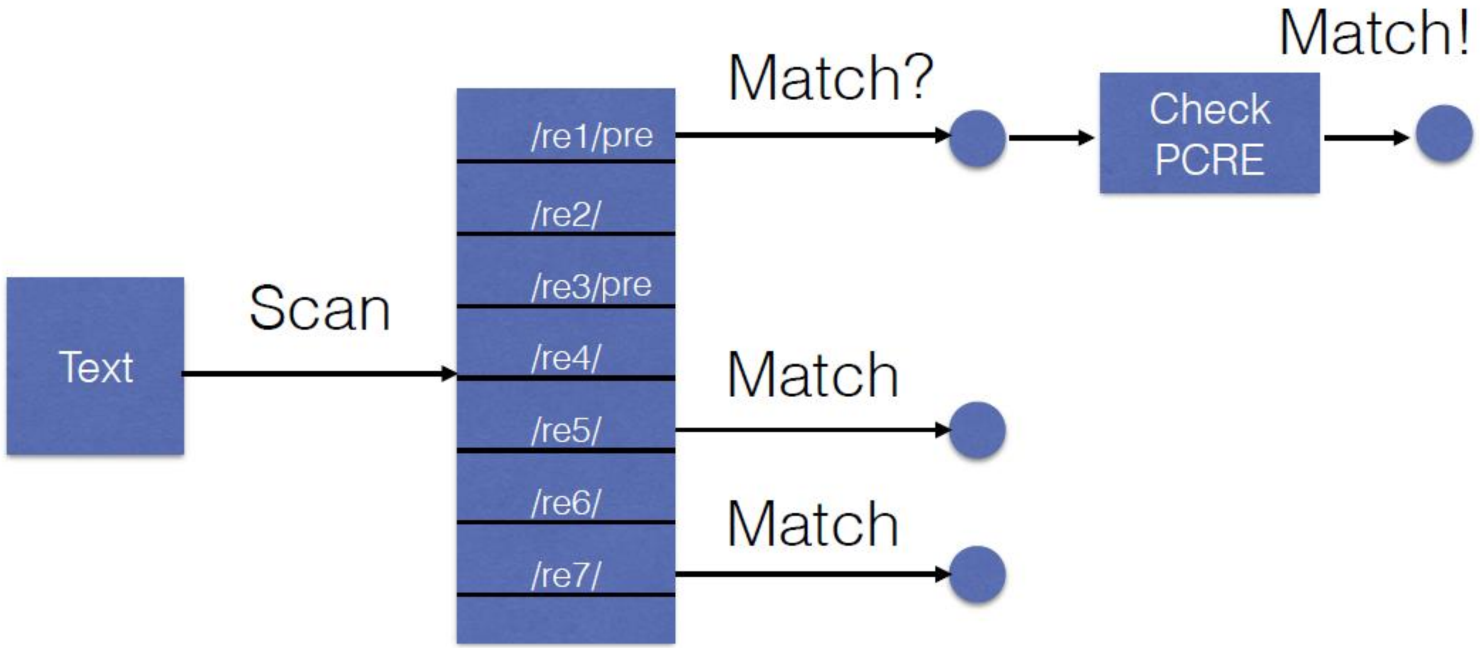
Hyperscan runtime



Hyperscan + Prefilter runtime



Hyperscan + Prefilter + PCRE confirm



Result

Before(only PCRE):

len: 610591, time: **882.251ms**

regexp statistics: **4095** pcre regexps scanned, **694M** bytes scanned using pcre

After(Hyperscan + Prefilter + PCRE):

len: 610591, time: **309.785ms**

regexp statistics: **34** pcre regexps scanned, **8.41M** bytes scanned using pcre, **9.56M** bytes scanned total

Conclusion

Why replacing PCRE with Hyperscan?

Better performance

Multi-pattern matching

Prefilter

Reuse pre-compiled bytecode

Easy coding

Links

Hyperscan

https://01.org/hyperscan

Rspamd

https://www.rspamd.com

Hyperscan intergrated in Rspamd

https://01.org/node/4020

intel®