HyperScan vs PCRE
Raport z projektu

Autorzy:

1 Wydajności czasowe

1.1 Jak ilość regexów wpływa na czas wykonania się programu?

1.1.1 Podział wzorców regex

W celu analizy wpływu charakteru wyrażeń regularnych na wydajność silników Hyperscan i PCRE, zastosowane wzorce podzielono na cztery grupy różniące się złożonością oraz właściwościami obliczeniowymi.

- **Grupa 1 – literały (słowa)** Pojedyncze ciągi znaków bez metaznaków regex. Brak ograniczeń początku dopasowania, duża liczba potencjalnych trafień.

```
Marketing
Download
upon
needed
```

- **Grupa 2 – regexy literalne z granicami słowa** Długie literały ograniczone granicami słowa. Mała liczba pozycji startowych i bardzo niska częstość dopasowań.

```
(^|[^A-Za-z0-9_])uuro($|[^A-Za-z0-9_])
(^|[^A-Za-z0-9_])ipxlydil($|[^A-Za-z0-9_])
(^|[^A-Za-z0-9_])kbvu($|[^A-Za-z0-9_])
(^|[^A-Za-z0-9_])dvgsgo($|[^A-Za-z0-9_])
```

- **Grupa 3 – regexy strukturalne** Wzorce o określonej strukturze (klasy znaków, kwantyfikatory, alternacje). Szybkie odrzucanie niedopasowanych fragmentów tekstu.

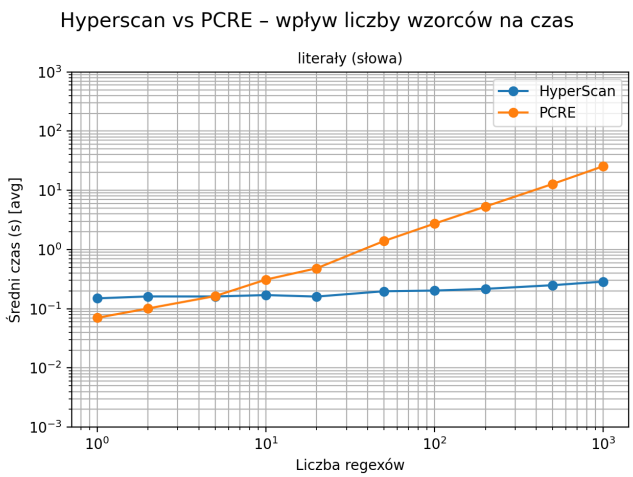
```
(sygjyojr|tqoeytlphoi)[ _-]?[0-9]{5}
vwqujeqbus[._-]?sxcds
(tzsv|oatwzycjn)[ _-]?[0-9]{3}
uafffats[ \t\r\n\f\v]?(id|code|ref)[ \t\r\n\f\v]?[0-9]+
```

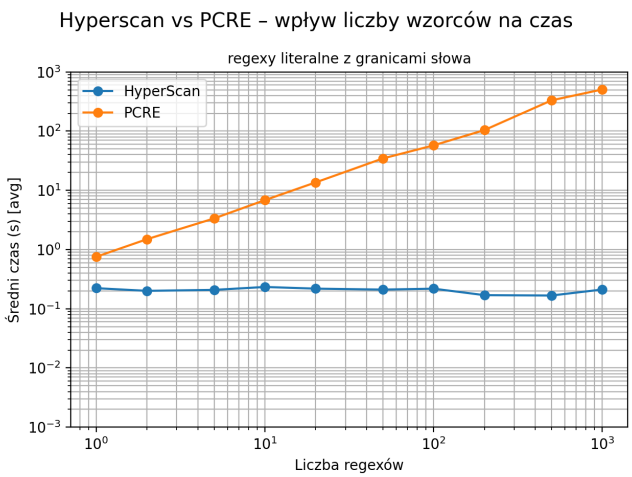
- **Grupa 4 – regexy z szerokim dopasowaniem** Wzorce zawierające wildcardy (`.*`, `.{0,200}`), alternacje i powtórzenia. Duża przestrzeń dopasowania i potencjalnie wysoki koszt obliczeniowy.

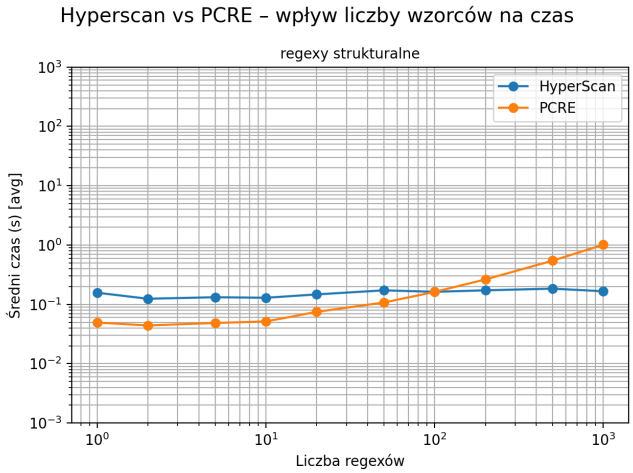
```
(?:zzblc|krnelhlomj|cbnrfmltbjnl)[^\n]{50,300}
(?:flnrhy|nkyoscvqvs|ggbciouzrgqc).{0,200}(?:id|code|ref)[=:]?\d+
(?:ohoceuukl\s*){2,6}ykhzzboa
(?:[A-Za-z0-9_-]+/)+yljpnfnsdo
```

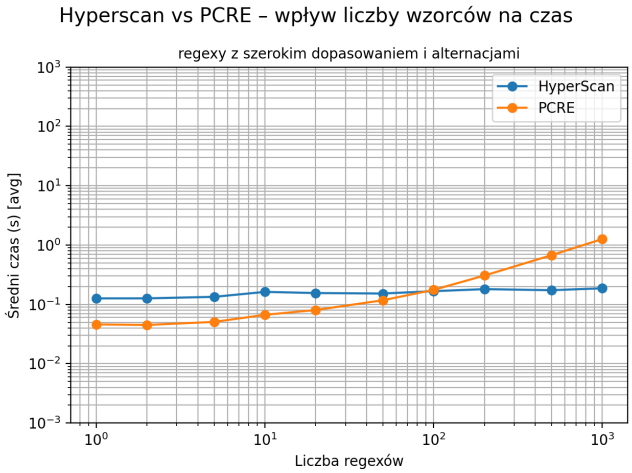
Podział ten umożliwia porównanie zachowania silników dla wzorców o rosnącej złożoności oraz analizę wpływu typu regexów na skalowanie czasowe.

1.1.2 Wyniki









1.1.3 Wnioski

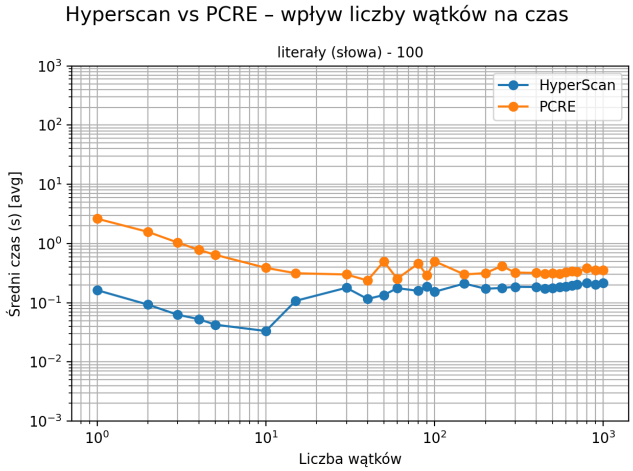
- Dla wszystkich typów regexów czas działania obu silników rośnie wraz z liczbą wzorców. Jednak wraz ze wzrostem liczby wzorców Hyperscan wyraźnie lepiej radzi sobie z szukaniem wzorców niż PCRE. Różnica pomiędzy silnikami rośnie wraz z liczbą regexów i dla dużych zbiorów staje się bazdzo znacząca.
- Najlepsze wyniki dla obu silników uzyskano w przypadku regexów strukturalnych oraz regexów z szerokim dopasowaniem, między którymi nie ma wyraźniej różnicy w zachowaniu żadnego z silników. W praktyce oznacza to, że obie te kategorie są przetwarzane w sposób bardziej stabilny, bez gwałtownych wzrostów czasu wraz z liczbą regexów.
- Hyperscan charakteryzuje się bardzo wysoką wydajnością we wszystkich analizowanych przypadkach - jego czas wykonania zmienia się tylko nieznacznie wraz ze wzrostem liczby i typu regexów.
- Silnik PCRE najlepiej radzi sobie z regexami strukturalnymi oraz z szerokim dopasowaniem, osiągając dla nich znacznie lepsze wyniki niż dla regexów prawie słownych, w których czas działania PCRE rośnie zauważalnie szybciej.

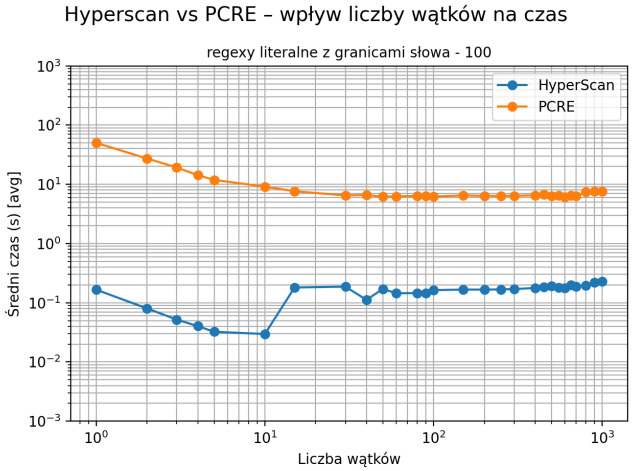
Uzyskane wyniki wskazują, że im większa jest liczba jednocześnie analizowanych wzorców, tym przewaga Hyperscana nad PCRE staje się wyraźniejsza, co czyni go szczególnie dobrze przystosowanym do zadań typu multi-pattern matching.

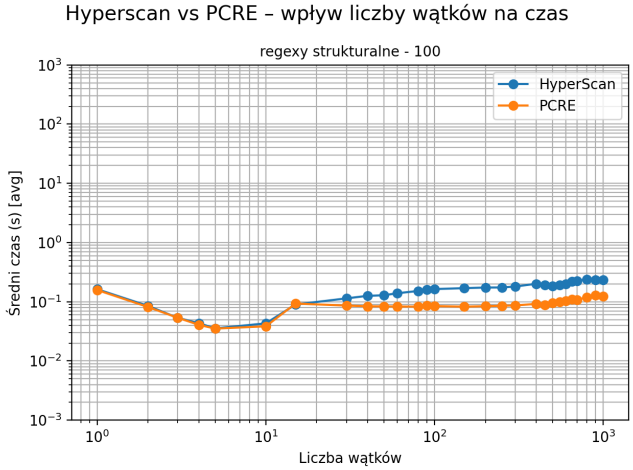
Dodatkowo "stały"czas HyperScana może wynikać z jużnie z pattern matchingu a poprostu otwierania, pobieraia itd plików

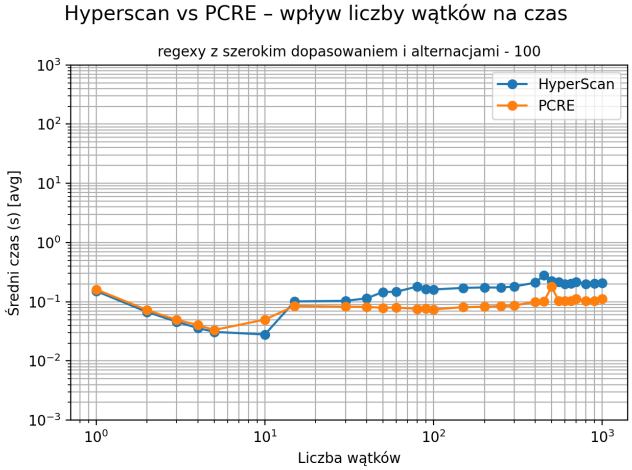
1.2 Jak ilość wątków wpływa na czas wykonania się programu?

1.2.1 Wyniki - 100 regexów

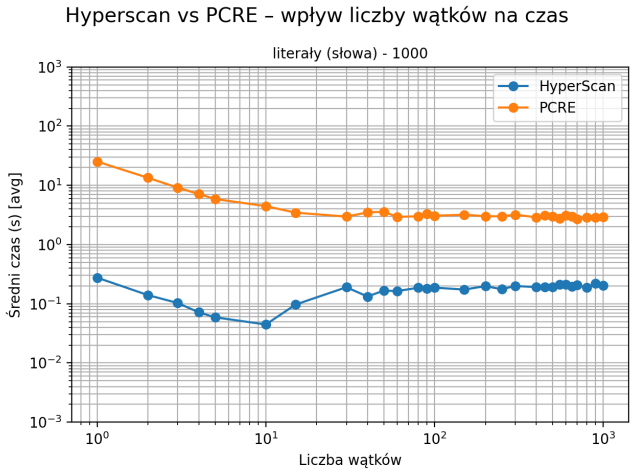


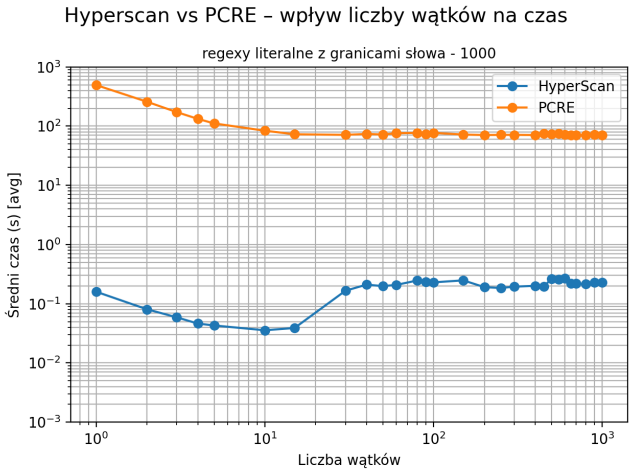


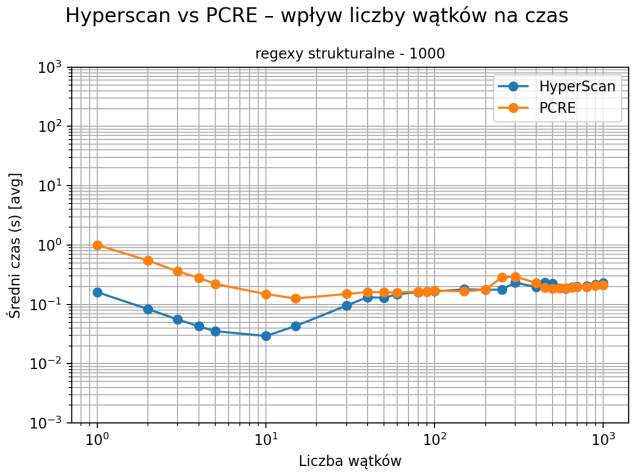


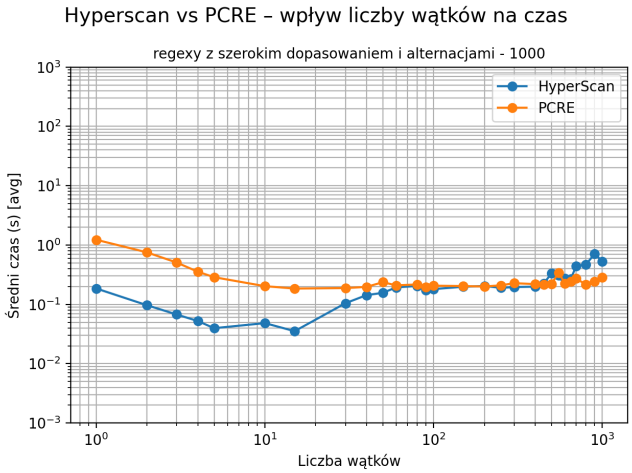


1.2.2 Wyniki - 1000 regexów







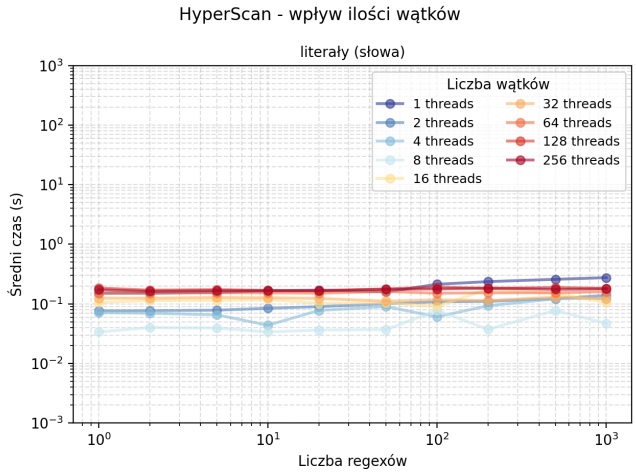


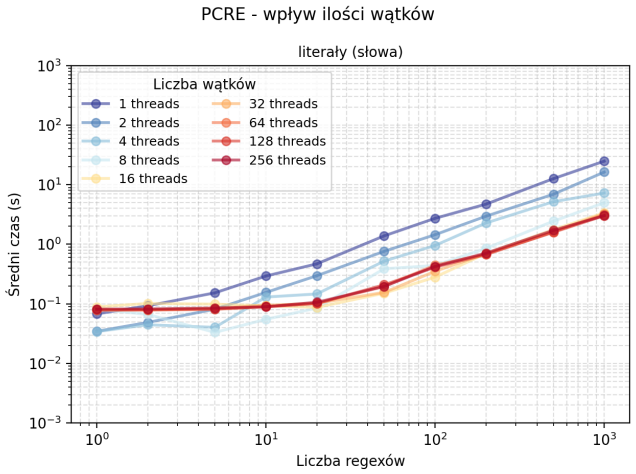
1.2.3 Wnioski

- Dla obu silników zwiększanie liczby wątków początkowo znacznie skraca czas wykoanania, co wskazuje na efektywne wykorzystanie równoległości.
- Dla PCRE liczba wątków, przy której osiągany jest najlepszy czas, zależy od typu regexów. Dla regexów słownych minimalny czas uzyskiwany jest przy większej liczbie wątków niż dla regexów strukturalnych i z szerokim dopasowaniem
- Dla regexów strukturalnych i z szerokim dopasowaniem przy dużej liczbie wątków (rzędu ≥ 100) czasy działania Hyperscana i PCRE stają się do siebie zbliżone.

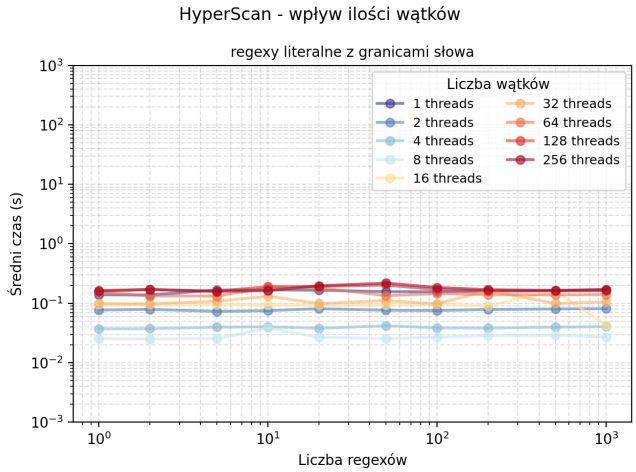
1.3 Jak ilość regexów vs ilość wątków wpływa na czas wykonania się programu?

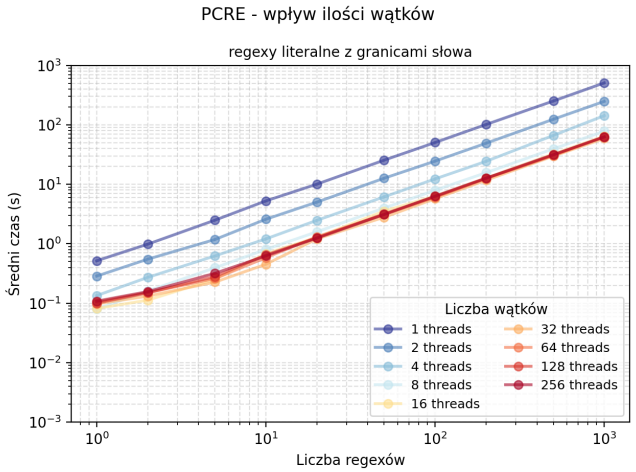
1.3.1 literały (słowa)



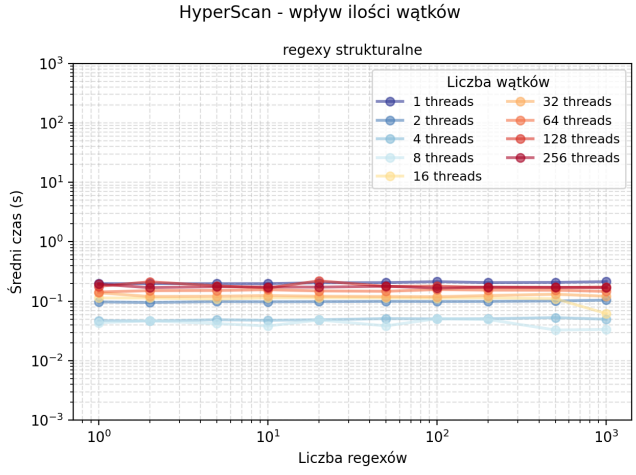


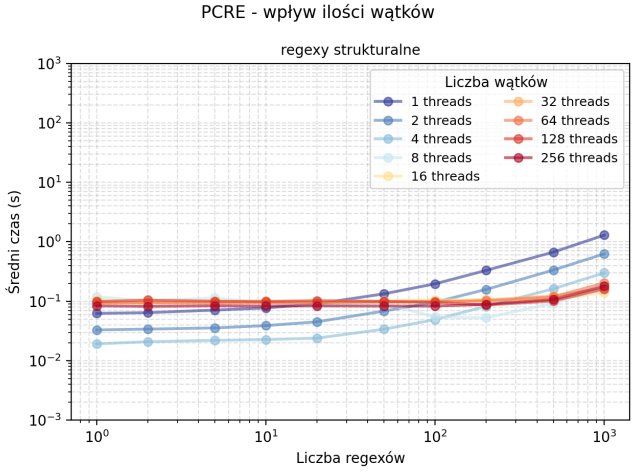
1.3.2 regexy literalne z granicami słów



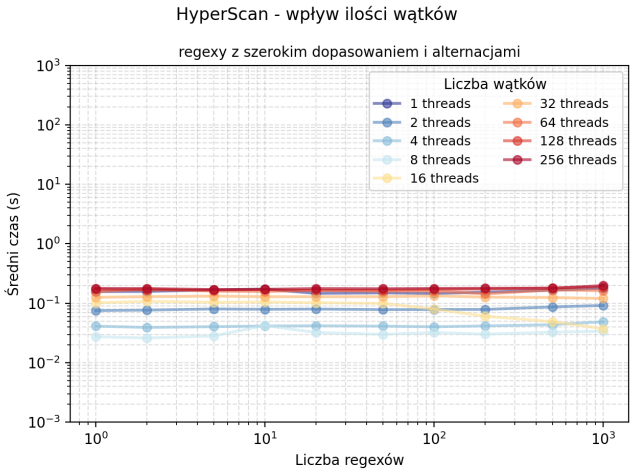


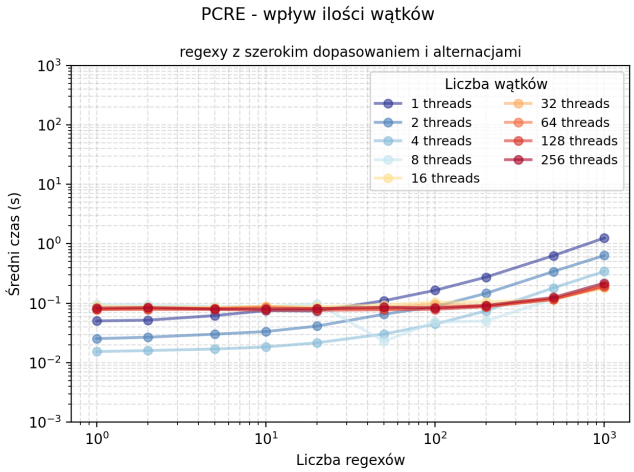
1.3.3 regexy strukturalne





1.3.4 regexy z szerokim dopasowaniem i alternacjami

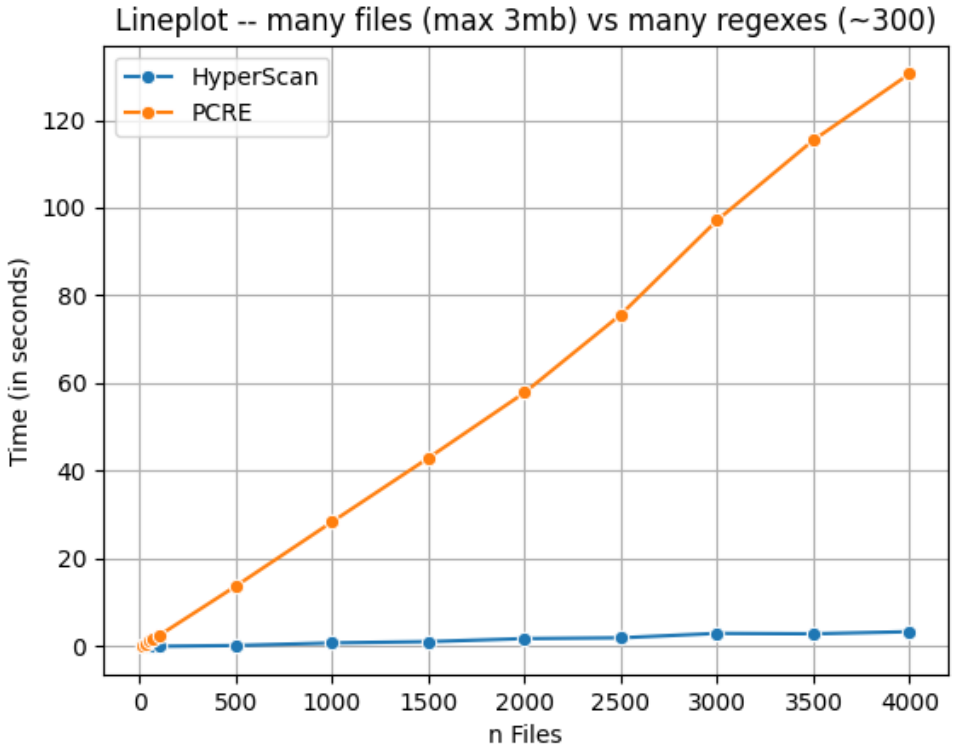




1.3.5 Wnioski

- Dla HyperScana zwiększenie ilości wątków dla ma niewielki wpływ na czas działania
- Dla PCRE większa liczba wątków rzeczywiście przyspiesza działanie, jednak nie kompensuje w pełni wzrostu czasu wynikającego ze zwiększania liczby regexów.
- Dla PCRE to przy ilu wątkach osiąga najlepszy czas zależy od regexów, dla regexów typu słownych sporo dalej osiąga najniższy czas w przeciwieństwie do regexów strukturalnych czy z szerokim dopasowaniem gdzie osiąga najniższ czas dla podobnej ilości wątków co HyperScan
- Podobnie jak w poprzedniej analizie, dla regexów strukturalnych i z szerokim dopasowaniem przy dużej liczbie wątków (rzędu ≥ 100) czasy wykonywania obu silników stają się zbliżone.

1.4 Jak ilość małych (≤ 3 MB) plików wpływa na czas wykonywania?



Pliki użyte w teście miały nie więcej niż 3 MB i zostały wygenerowane poleceniem:

```
base64 /dev/urandom | head -c 3MB > ...
```

Do testów wykorzystano 100 wątków, a każdy program miał 3 podejścia do problemu. Wynikiem przedstawionym na wykresie jest średnia arytmetyczna całkowitego czasu wykonania.

1.4.1 Przykłady regexów użytych w teście

```
^[0-9]+(\.[0-9]+)?\s?(l|ml)$
^Saldo:\s-?[0-9]+,[0-9]{2}\sPLN$
^Dawka:\s[0-9]+(mg|ml)$
```

1.4.2 Wnioski z wykresu

- Do około 100 plików, zarówno HS, jak i PCRE radziły sobie podobnie, z czasem wykonywania od 2 do 200 milisekund.
- Wraz ze wzrostem liczby plików czas wykonywania PCRE rośnie znacznie szybciej niż HS.
- Przyspieszenie PCRE jest zdecydowanie większe niż przyspieszenie HS.
- Czas dzialania HS dla 4000 plikow nie przekracza 4s, a czas dzialania PCRE przewyzsza 120 sekund.
- HS lepiej radzi sobie z łączeniem skomplikowanych regexów przy pracy na wielu plikach.