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.

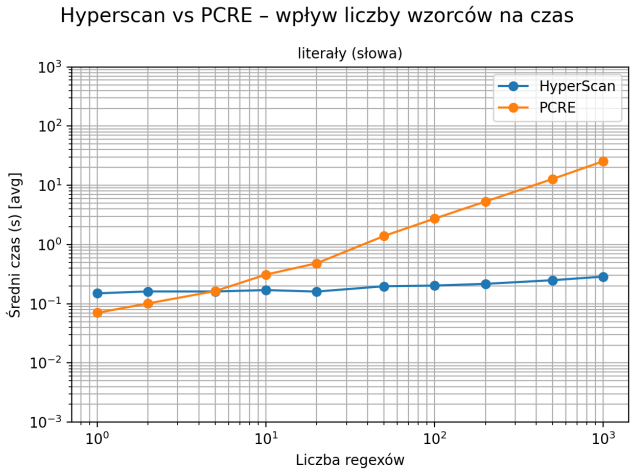
```
(sygjyojr|tqoeytlphoi)[ _-]?[0-9]{5}
vwqujeqbus[._-]?sxcds
(tzsv|oatwzycjn)[ _-]?[0-9]{3}
uafffats[ \t\r\n\f\v]?(id|code|ref)[ \t\r\n\f\v]?[0-9]+
```

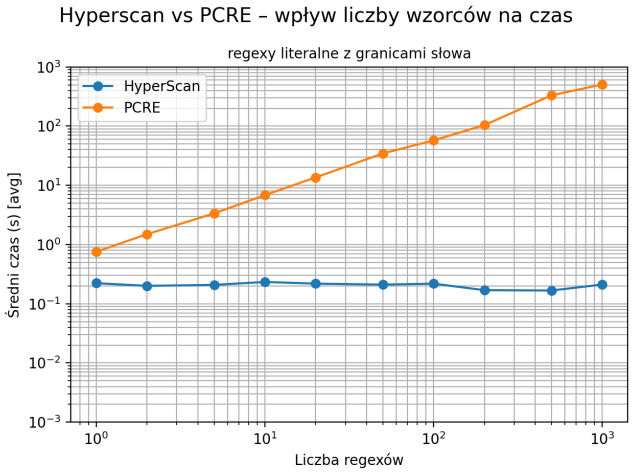
- **Grupa 4 – regexy z szerokim dopasowaniem** Wzorce zawierające wildcardy (`.*`, `.{0,200}`), alternacje i powtórzenia. Duża przestrzeń dopasowania i potencjalnie wysoki koszt obliczeniowy.

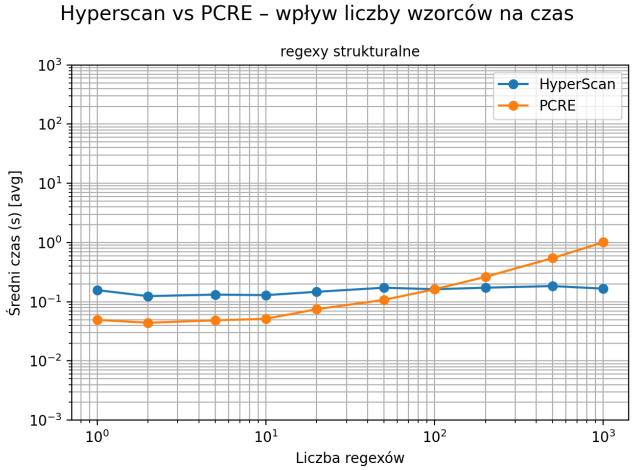
```
(?:zzblc|krnelhlomj|cbnrfmltbjnl)[^\n]{50,300}
(?:flnrhy|nkyoscvqvs|ggbciouzrgqc).{0,200}(?:id|code|ref)[=:]?\d+
(?:ohoceuukl\s*){2,6}ykhzzboa
(?:[A-Za-z0-9_-]+/)+yljpnfnsdo
```

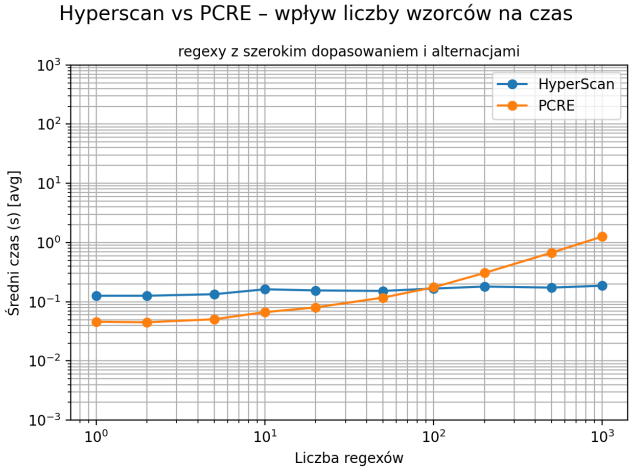
Podział ten umożliwia porównanie zachowania silników dla wzorców o rosnącej złożoności oraz analizę wpływu typu regexów na skalowanie czasowe.

1.1.2 Wyniki









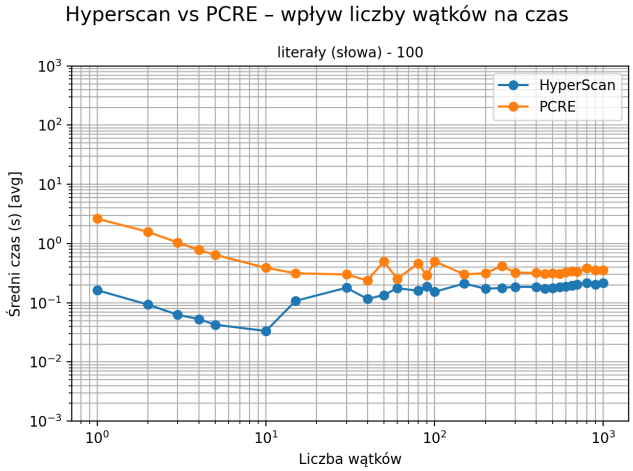
1.1.3 Wnioski

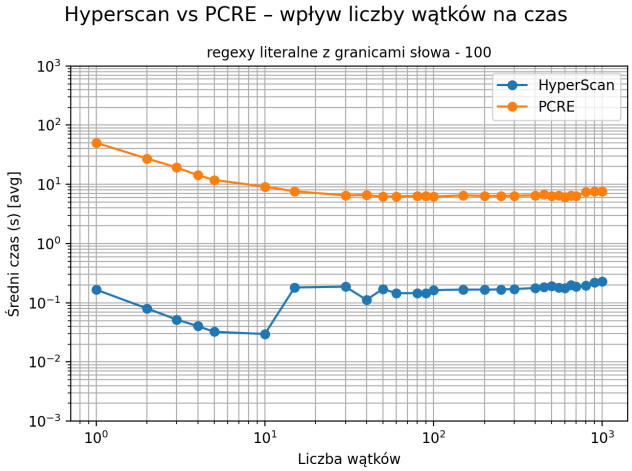
- Wraz ze wzrostem liczby wzorców Hyperscan wyraźnie lepiej radzi sobie z szukaniem wzorców niż PCRE. Różnica pomiędzy silnikami rośnie wraz z liczbą regexów i dla dużych zbiorów osiąga kilka rzędów wielkości.
- Najlepsze wyniki dla obu silników uzyskano w przypadku regexów strukturalnych. Wzorce o jasno określonej strukturze umożliwiają szybkie odrzucanie niedopasowanych fragmentów tekstu, co przekłada się na niski i stabilny czas wykonania.
- Hyperscan charakteryzuje się bardzo wysoką wydajnością we wszystkich analizowanych przypadkach, jednak najlepsze rezultaty uzyskuje dla regexów literalnych z granicami słowa oraz dla regexów strukturalnych.
- Silnik PCRE zdecydowanie najlepiej radzi sobie właśnie z regexami strukturalnymi, osiągając dla nich znacznie lepsze wyniki niż dla pozostałych kategorii wzorców. W pozostałych grupach czas działania PCRE rośnie zauważalnie szybciej.
- Krótkie literały bez ograniczeń początku dopasowania oraz regexy z szerokim dopasowaniem i alternacjami stanowią dla Hyperscana większe obciążenie, choć nawet w tych przypadkach jego wydajność pozostaje znacząco wyższa niż w przypadku PCRE.

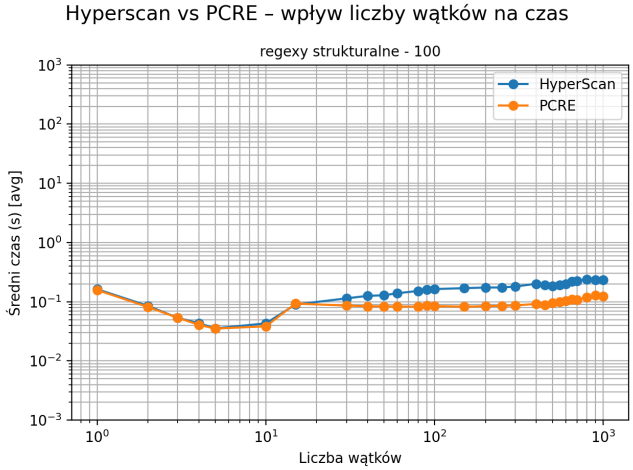
Uzyskane wyniki wskazują, że im większa jest liczba jednocześnie analizowanych wzorców, tym przewaga Hyperscana nad PCRE staje się wyraźniejsza, co czyni go szczególnie dobrze przystosowanym do zadań typu multi-pattern matching.

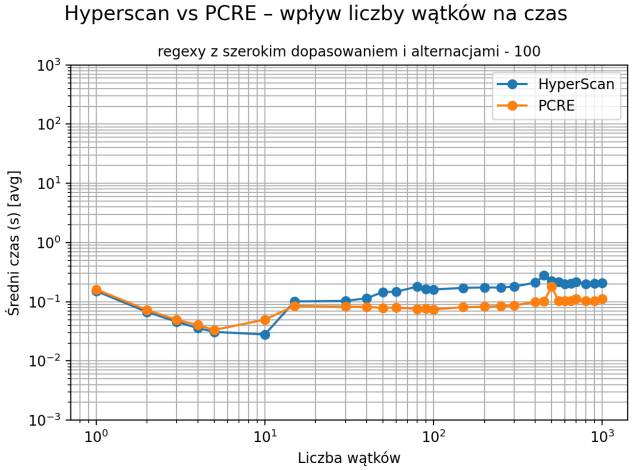
1.2 Jak ilość wątków wpływa na czas wykonania się programu?

1.2.1 Wyniki - 100 regexów

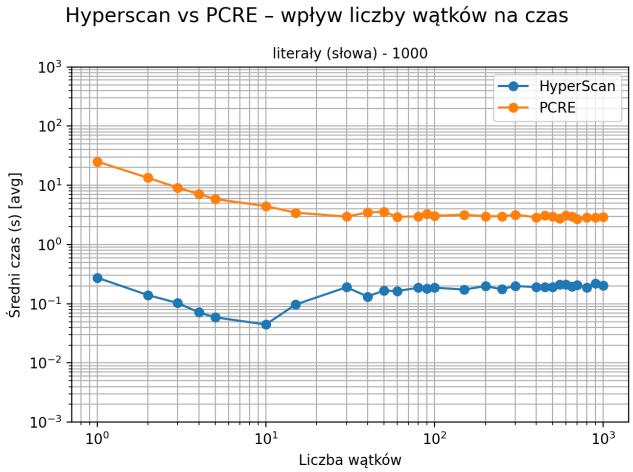


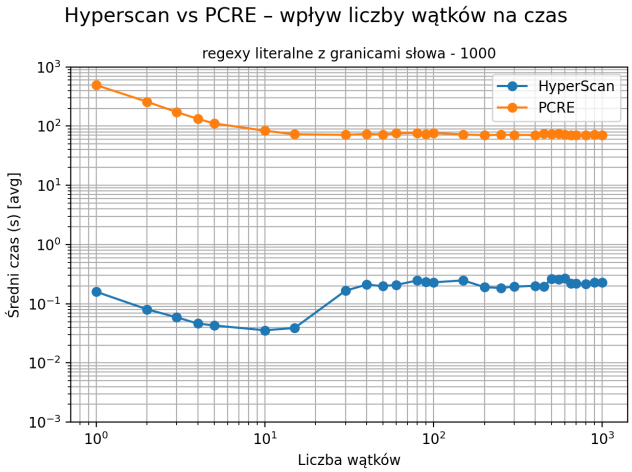


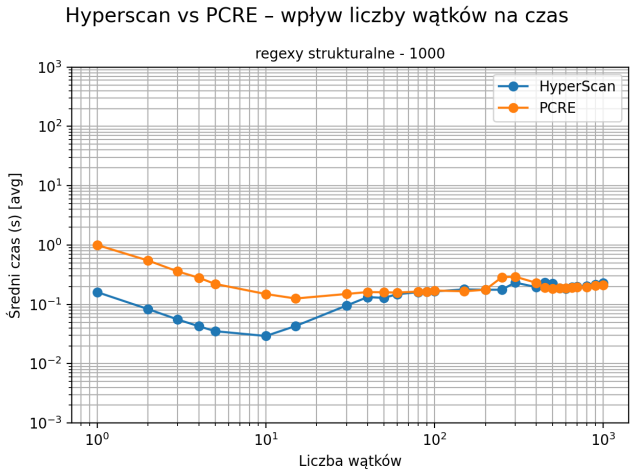


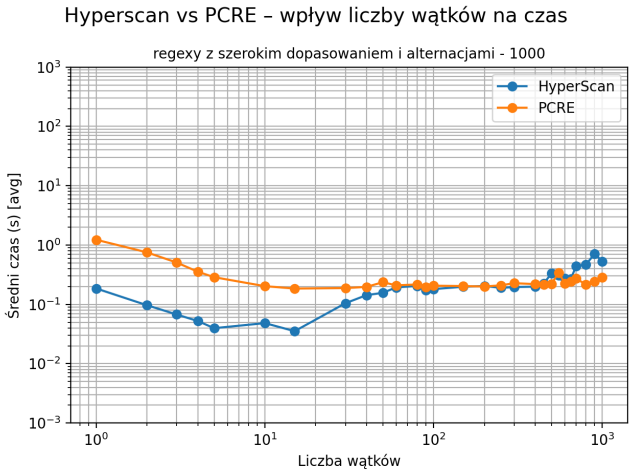


1.2.2 Wyniki - 1000 regexów



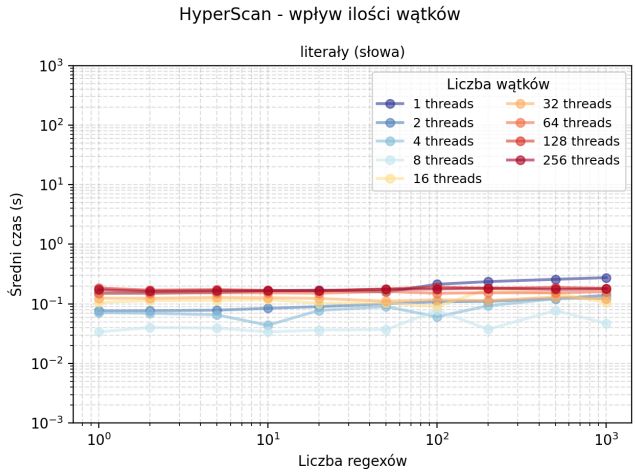


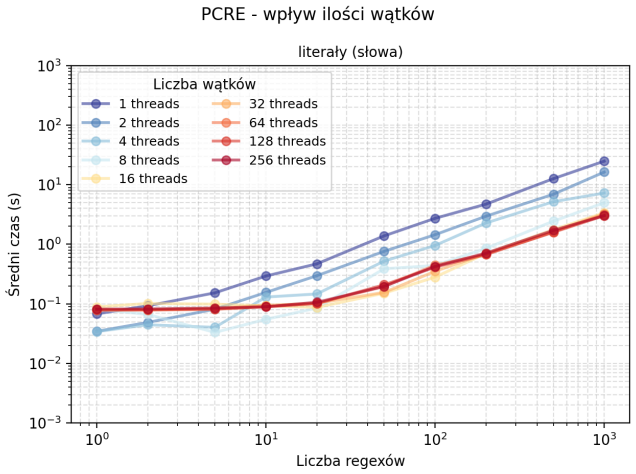




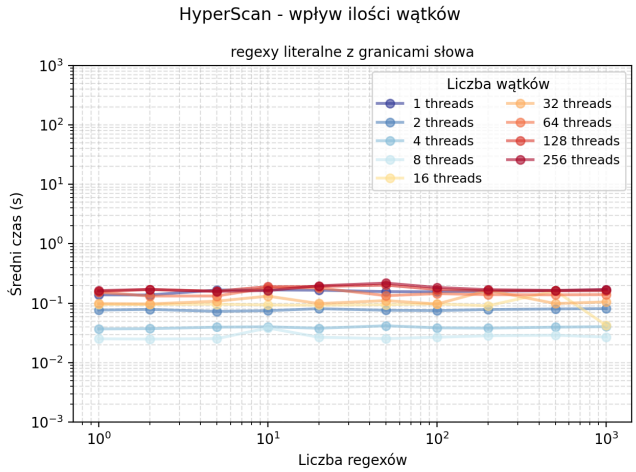
1.3 Jak ilość regexów vs ilość wątków wpływa na czas wykonania się programu?

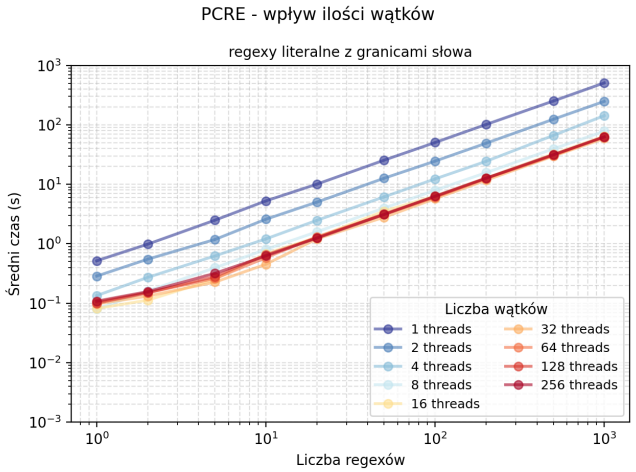
1.3.1 literały (słowa)



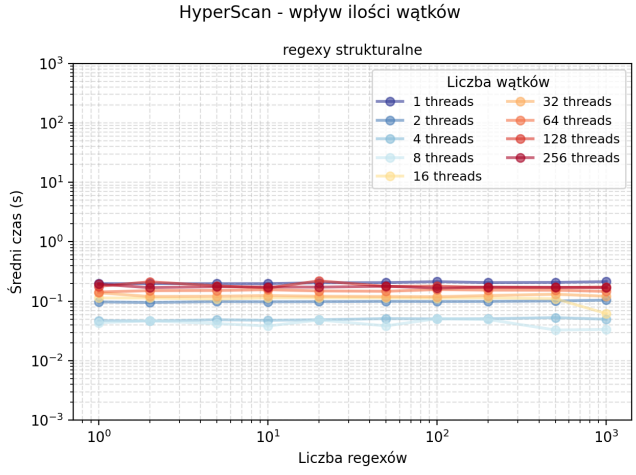


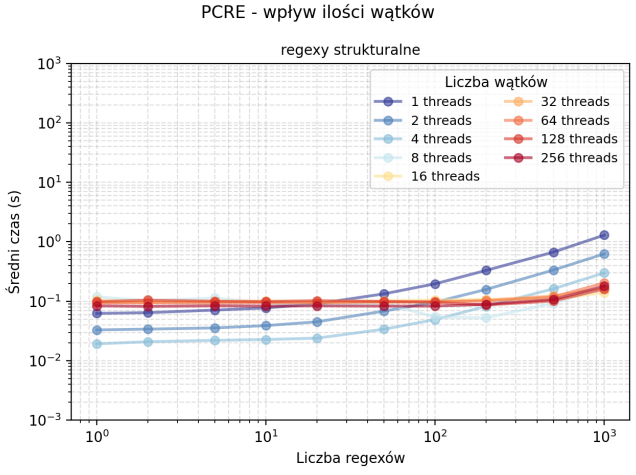
1.3.2 regexy literalne z granicami słów



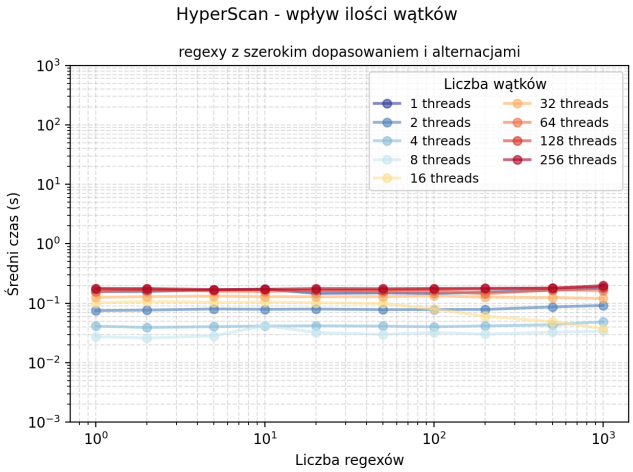


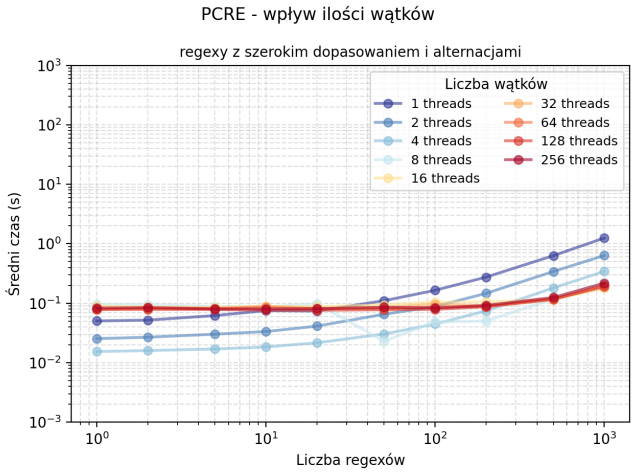
1.3.3 regexy strukturalne



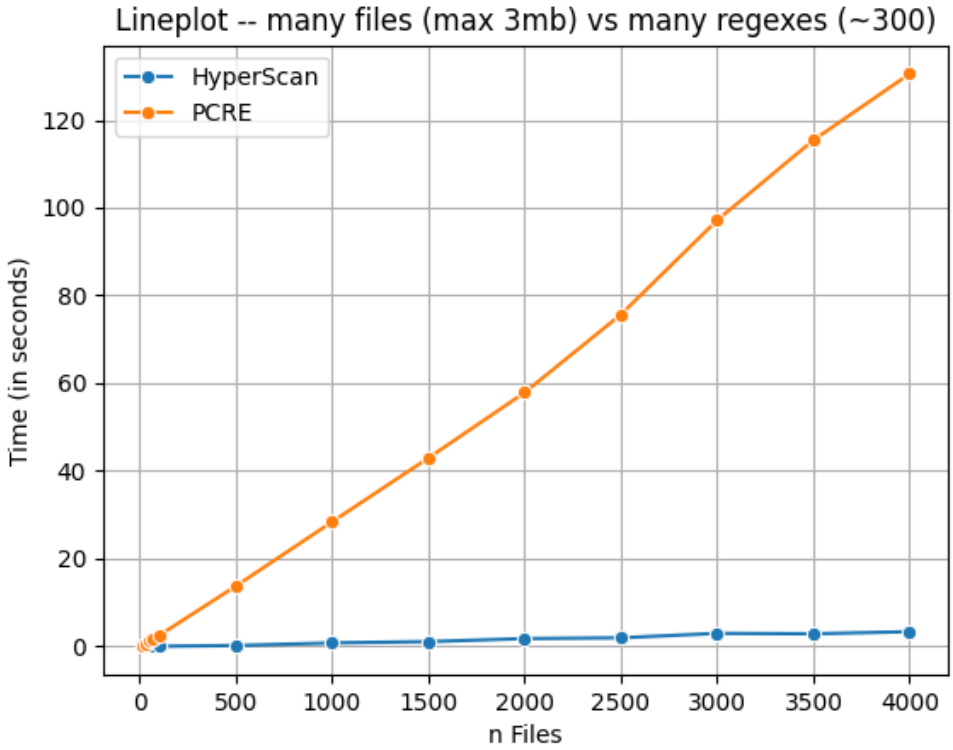


1.3.4 regexy z szerokim dopasowaniem i alternacjami





1.4 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.