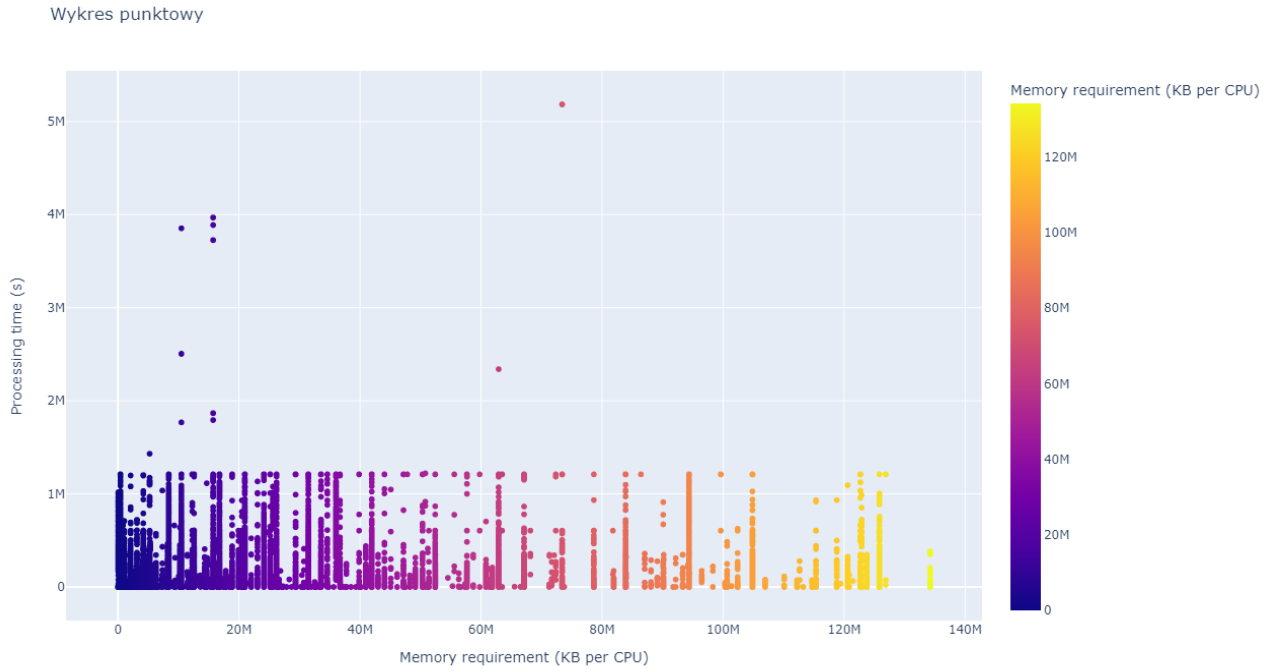
M05. Zarządzanie zasobami w klastrach i chmurach obliczeniowych

Andrzej Szczap
s458010

1. Analiza danych z pliku zapat.csv

Analiza pliku zapat.csv wskazuje, że wraz ze wzrostem zapotrzebowania na czas dla pojedynczego zadania, maleje ilość zadań. Podobnie jest w przypadku pamięci, najwięcej jest zadań które nie przekraczają pewnej wartości zapotrzebowania na pamięć. Powyżej tego poziomu ilość zadań znacząco spada.

Przedstawia to następujący histogram.



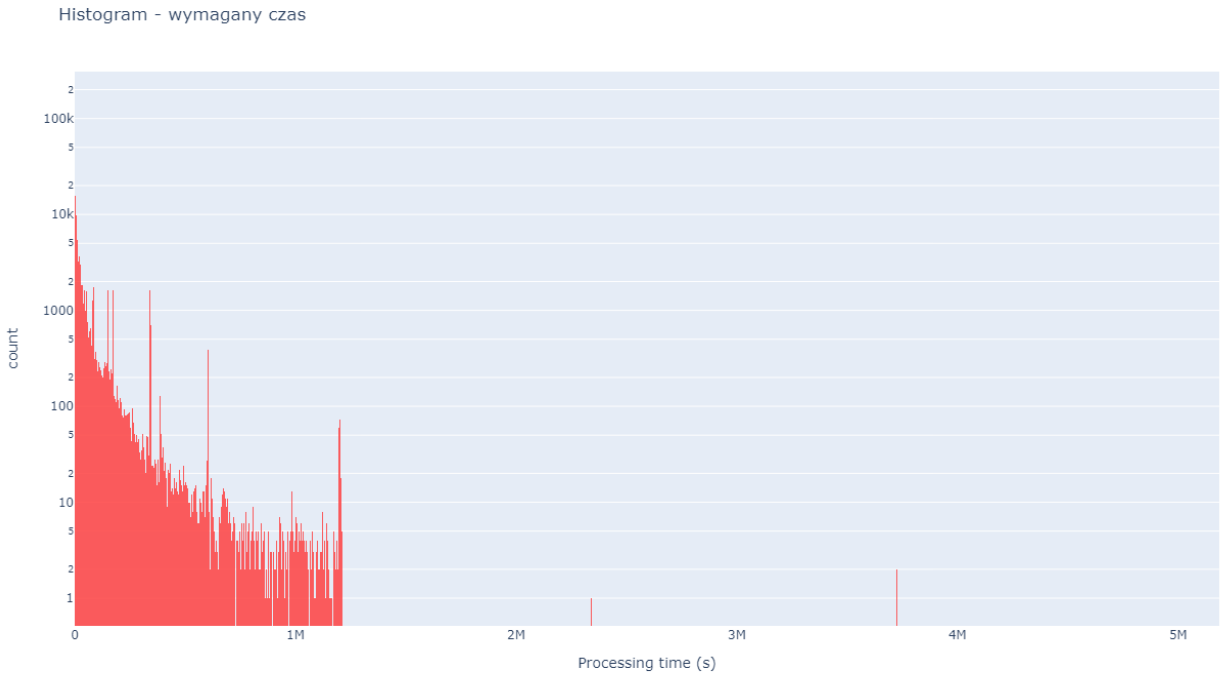
Jak więc widzimy większość zadań nie przekracza pewnych wartości, jest to około 95% zadań.

Większość zadań nie przekracza ram czasowych od 0 do około 1.3M sekund czasu wykonywania.

Naszymi dostępnymi zasobami jest 16 rdzeni CPU, 134GB pamięci RAM.

Zatem możemy założyć że głównym ograniczeniem będzie tutaj ilość dostępnych procesorów.

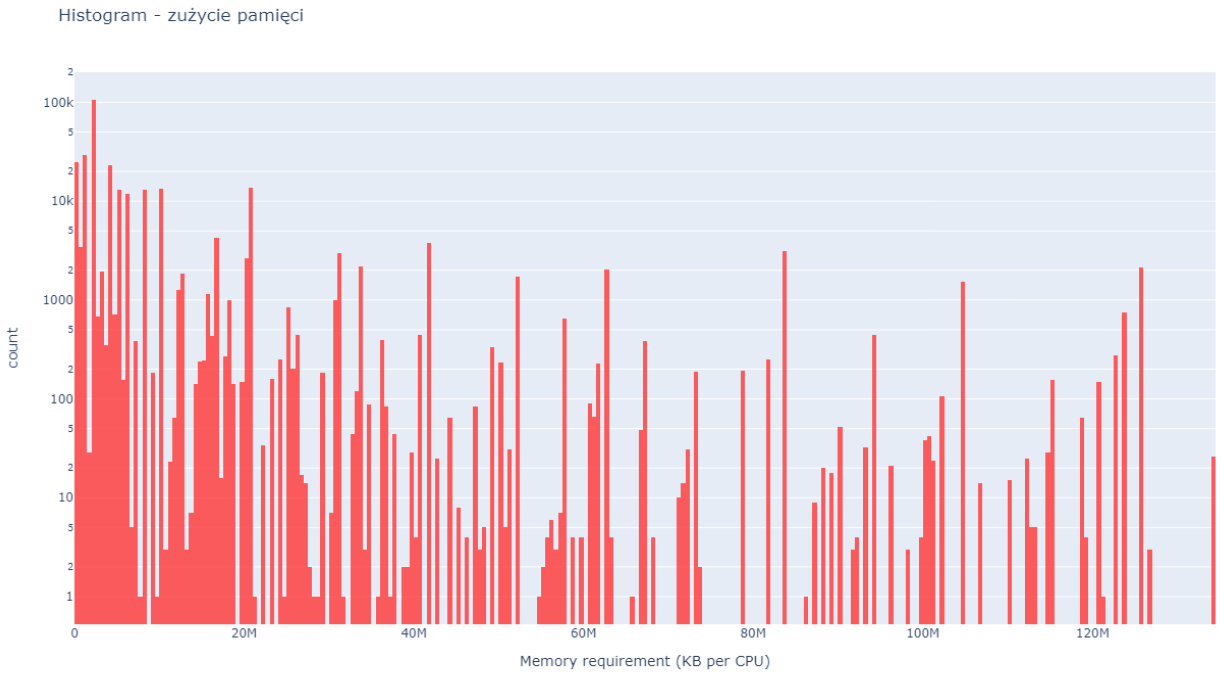
Kolejny histogram wymaganego czasu dla ilości zadań



Z analizy wymaganej pamięci operacyjnej dla zadań można wyciągnąć wniosek,

że większość zadań mieści się w górnym ograniczeniu 134GB, a około 95% zadań nie przekracza połowy dostępnych zasobów.

Możemy to dostrzec z następnego histogramu.



Obecne wnioski są następujące.

Najważniejsze dla naszego uszeregowania będzie właściwe korzystanie z dostępnych procesorów.

Pamięć operacyjna jest ważna, lecz mamy jej wystarczającą ilość i cenniejsze dla uszeregowań będzie właściwe zarządzanie procesorami.

2. Szeregowanie

Strategia ALFA

Algorytm ALFA polega na posortowaniu zadań na wstępie według długości ich czasu wykonywania.

Następnie dopóki mamy dostępne zadania, szukamy procesor który najszybciej będzie wolny, czyli będzie mieć najmniejszą wartość CMAX.

Jeśli kolejne zadanie do uszeregowania spełnia wymagania, czyli wymagana pamięć zadania mieści się w przedziale do 134GB odjąć już używana pamięć, to dodajemy zdanie do szeregu.

Jeżeli zadanie nie spełnia wymagań to szukamy kolejnego zadania które spełnia warunek.

Gdy żadne zadanie nie spełnia warunku, to lista zadań jest ponownie sortowana według ich

czasu zakończenia.

Gdy zdania się kończą to pamięć jest zwalniana i możliwe jest dodanie większych zadań.

Jest tutaj wykorzystana postać znormalizowana na pamięć, wartość mi = 0 gdy zadanie nie potrzebuje pamięci, oraz mi=1 gdy wymaga całą dostępną pamięć czyli 134GB.

Takie uszeregowanie sprawia że na początku wykonujemy zadania które mają najmniejsze wymagania i do pamięci i do czasu wykonywania.

Strategia BETA

Na początku sortujemy zadania według podzielonej wartości czasu wykonywania przez potrzebną pamięć dla każdego zadania.

Następnie jeśli zadanie spełnia wymogi, czyli mieści się w dostępnej wartości pamięci operacyjnej,

Jest przypisywane dla procesora który ma najszybszy wolny czas, wartość CMAX.

Natomiast jeśli zadanie nie spełnia warunku, szukane jest zadanie które da się zmieścić.

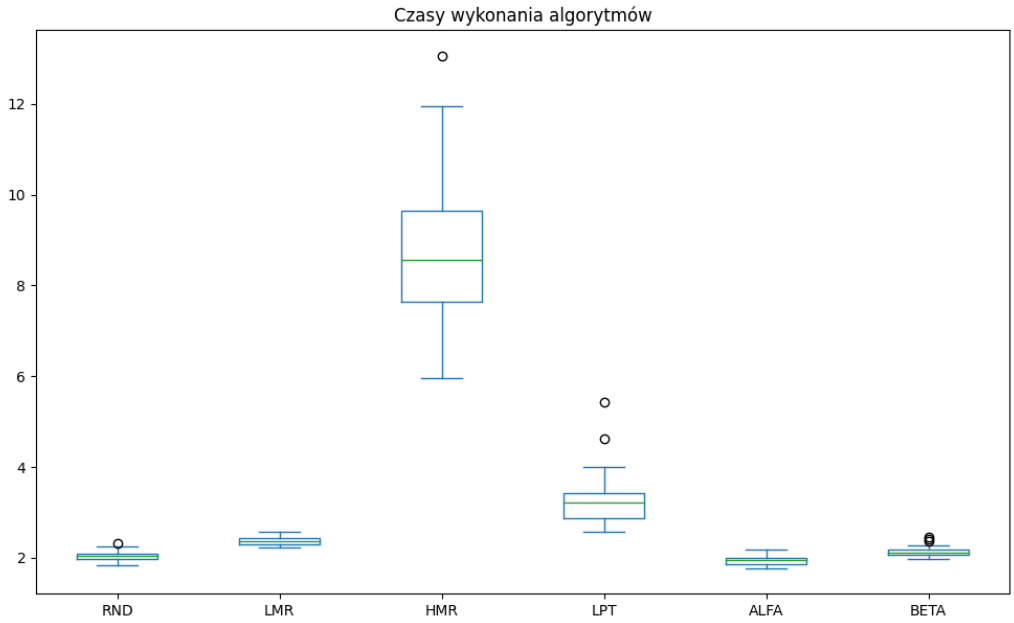
Jeśli nie znajdziemy takiego zadania które da się zmieścić w dostępnych dla nas zasobach, sortujemy zadania według czas zakończenia.

Gdy uszeregowane zadania zwolnią pamięć dodajemy nowe zadania do uszeregowania.

Takie rozmieszczenie zadań powinno skutkować wykorzystywaniem dużej wartości pamięci operacyjnej dla wszystkich procesorów.

3. Wykresy skrzynkowe

Analiza czasów wykonywania algorytmów



Z wykresu możemy wywnioskować że RND, LMR, ALFA i BETA szeregują zadania w podobnym czasie.

Znacząco wyróżnia się HMR który ma wysoki przedział, oraz LPT który ma 2 skrajne wartości.

Algorytm RND

Zadania są przydzielane w sposób losowy, nie są porządkowane na wejściu.

Najczęściej zadanie będzie porządkowane do pierwszego wolnego procesora ze względu na to,

że mało jest zadań które przekraczają pewne wartości potrzebnej wartości operacyjnej.

Algorytm LMR

Różni się od RND tym że dane wejściowe są porządkowane według zapotrzebowania na pamięć operacyjną.

Następnie zadania są przydzielane w podobny sposób co w RND.

Algorytm HMR

Dane wejściowe są porządkowane według zapotrzebowania na pamięć operacyjną malejąco.

Czas pracy algorytmu spowodowany jest tym, że często sprawdza czy dla obecnie dostępnych zasobów da się jeszcze zmieścić jakieś zadanie.

Algorytm LPT

Dane wejściowe są porządkowane według największego zapotrzebowania na pamięć operacyjną.

Takie rozwiązanie powoduje, że najpierw wykonują się zadania ciężkie.

Mniejsze zadania są szeregowane w późniejszym etapie, algorytm często sprawdza dostępne zasoby,

wydłużając czas pracy.

Algorytm ALFA

W zestawieniu wypadł bardzo podobnie do RND oraz trochę lepiej od BETA.

Wstępne sortowanie danych według czasu wykonywania,

następnie szereguje zadania dla najszybszych procesorów, o najniższym CMAX.

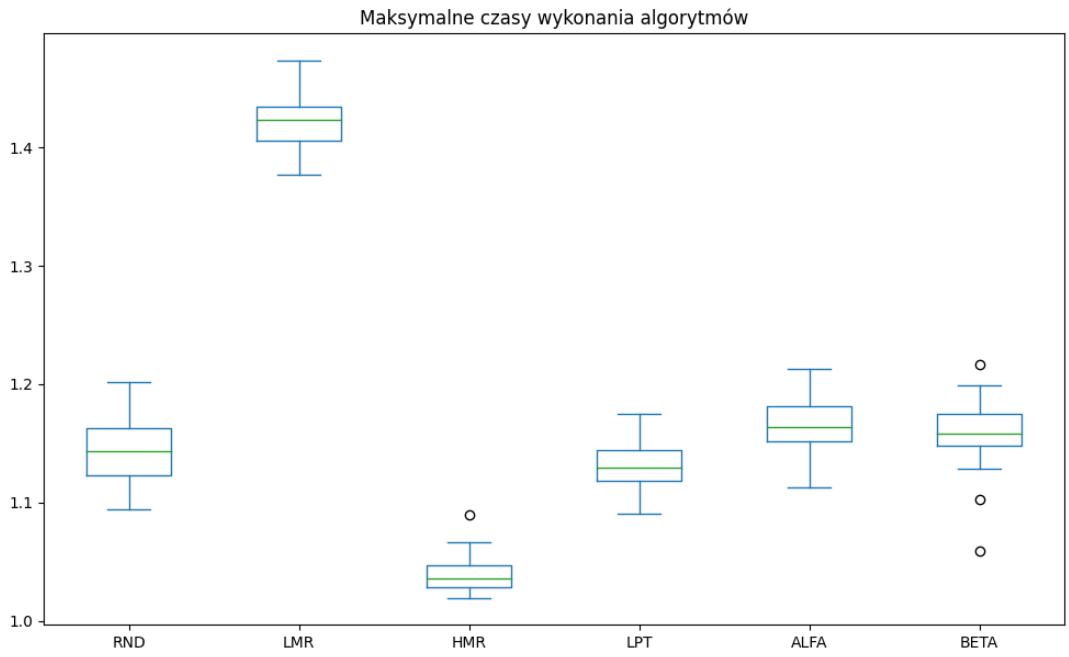
Okazuje się to być dobrą strategią.

Algorytm BETA

Osiąga podobny wynik do ALFA, choć operacja szeregowania zadań na wejściu przez podzielenie czasu wykonywania przez zapotrzebowanie na pamięć operacyjną go spowalnia.

Dalszy proces nie ma dużego wpływu na wynik w szeregowaniu.

Analiza maksymalnych czasów wykonywania



Algorytm RND

RND szereguje w sposób bardzo prymitywny, nie porządkuje danych wejściowych, dlatego przedział CMAX dla instancji jest tak duży. Sporo tutaj zależy od tego jakie wartości się trafią z wczytanego pliku.

Algorytm LMR

Ma najgorszy czas maksymalny dla uszeregowań, zadania na wejściu są porządkowane według rosnącego zapotrzebowania na pamięć operacyjną, następnie zadania są szeregowane dla procesorów.

Skutek tego jest taki że procesory są zapychane małymi zadaniami co wydłuża maksymalny czas zakończenia.

Algorytm HMR

Uzyskał najlepszy wynik w zestawieniu. Prawdopodobnie przez to że dane wejściowe są porządkowane według malejącego zapotrzebowania na pamięć operacyjną.

Więc duże zadania są dodawane w pierwszej kolejności a potem reszta wolnej pamięci jest wykorzystywana przez mniejsze zadania.

Algorytm LPT

Uzyskany wynik, prawdopodobnie jest skutkiem wstępnego szeregowania zadań według malejącego czasu wykonywania. Następnie zadania są dodawane do procesorów które mają najmniejszy CMAX, nie jest skomplikowany, porządkuje w prosty sposób.

Algorytm ALFA

ALFA uwzględnia dużo wartości w swoim przejściu co wydłuża czas. Mimo uszeregowania danych wejściowych według rosnącego czasu wykonywania nie wypadł dobrze.

Algorytm BETA

Wypada minimalnie lepiej niż ALFA. Algorytm szereguje dane wejściowe według podzielonej wartości czasu wykonywania przez potrzebną pamięć operacyjną. Następnie w procesie szeregowania dodawane są zadania o niskich kosztach pamięci operacyjnej. Dla BETA zdarzyły się wartości znacząco odbiegające od uśrednionych wartości.