Środowisko

	Laptop autora	**Środowisko students**
Rodzaj urządzenia	Laptop Lenovo Legion	Maszyna Students
System operacyjny	Linux	Linux
Dystrybucja	Ubuntu 20.04	Debian GNU (11)
Model procesora	AMD Ryzen 7 4800H with Radeon Graphics	Intel Xeon Processor (Skylake, IBRS)
Liczba gniazd	1	64
Liczba rdzeni na gniazdo	8	1
Liczba wątków na rdzeń	2	1
Architektura	x86_64	x86_64

Modyfikacje dołączonego kodu

- W chwili przygotowywania raportu w pliku *main.cpp* zakomentowano *sharedResults = true*, gdyż wówczas nie ukończono części dotyczącej współdzielenia plików.
- W *teams.cpp* zmieniono zaproponowane *ContestResult r* na *ContestResult result(inputSize, 0),* by ułatwić implementację umieszczania rezultatów.
- W *teams.hpp*, w części dotyczącej realizacji konkursu dla *TeamSolo* dodano *cout << "TeamSolo\n",* by ułatwić przetwarzanie wyników.
- Nie wykorzystano pliku *new_processes.hpp.*

Metodyka badań

Ze względu na różnorodność i obfitość danych, autor zdecydował się zawrzeć w suplemencie subiektywnie najbardziej znaczące zestawienia wyników. Doświadczenia powtórzono trzykrotnie na maszynie students (wymagane: *ulimit -s 128* przed uruchomieniem programu ze względu na ,,ograniczone zasoby") oraz na laptopie autora, jednak krotność eksperymentów uwzględnono jednak tylko w *Całkowitym czasie obliczeń na dwóch środowiskach.* Na pozostałych wykresach (w przypadku których zalecane jest powiększenie strony), natomiast w celu zachowania czytelności posługiwano się jedynie wynikami z pierwszych prób na maszynie students oraz na laptopie. W celu zachowania liczby stron zbliżonej do objętości ,,idealnego raportu", wyszczególniono wyniki tylko jednego konkursu uznanego przez autora za względnie wymagający i reprezentatywny – LongNumber – dla wszystkich zespołów oprócz tworzących procesy, gdyż destruktor umieszczonego w pamięci współdzielonej timera nie zwracał prawidłowo podsumowujących statystyk (w pozostałych zespołach dodano osobne timery). W statystykach dla poszczególnych konkursów nie wykorzystywano informacji z poszczególnych timerów dla danych wykonań, dodanych wewnątrz implementacji *runImpl* przez autora, zaś ,,średni czas" pochodzący z *TotalTimer*, co umożliwiło uwzględnienie również wyników dla procesów. Wszystkie wyniki, oprócz całkowitego czasu, zostały przeskalowane do milisekund. Zrezygnowano również z przedstawiania odchylenia standardowego.

Dane opracowano i zwizualizowano za pomocą własnego skryptu w Pythonie, dostępnego w publicznym reopzytorium wraz z częściowymi wynikami i danymi.[1]

Omówienie wyników

Przede wszystkim zwraca uwagę około dwukrotnie dłuższy czas wykonywania testów na maszynie students niż na własnym laptopie, co mogło wynikać zarówno z ,,mniejszych zasobów", sugerowanych wcześniejszymi problemami z uruchomieniem programu, jak i obciążeniem maszyny przez studentów intensywnie testujących programy.

W przypadku LongNumber zespoły TeamSolo oraz TeamNewProcesses, natomiast w pozostałych konkursach – TeamNewProcesses wyróżniają się zdecydowanie dłuższym czasem realizacji zadania. TeamSolo, jako jednowątkowe rozwiązanie, nie zaskakuje powolnością, gdyż praca nie może zostać rozdzielona równocześnie pomiędzy pozostałe wątki lub procesy. TeamNewProcesses wywołuje nowy proces dla każdego wywołania calcCollatz, jednak nie więcej niż getSize() procesów równocześnie. Wywołanie nowego procesu wymaga czasu dla rezerwowania nowych zasobów dla nowego procesu, dlatego kolejne wywoływanie nowych procesów, wraz z ograniczeniem ich liczebności, może sprzyjać spowolnieniu realizacji zadań. We wszystkich konkursach zadania realizuje najszybciej TeamAsync, prawdopodbnie dzięki zoptymalizowanemu przez bardziej doświadczonych programistów niż autor, asynchronicznemu wykonaniu,

1 https://github.com/EuphoricThinking/collatz_raport

umożliwionego przez uruchamiane na potrzeby zadania, niezależne wątki. Wraz ze złożonością danych wejściowych (większym rozmiarem danych), różnice w efektywności poszczególnych zespołów bardziej uwidaczniają się; w przypadku LongNumber rozwiązania wielowątkowe i tworzące stałą liczbę procesów charakteryzują się podobnymi rezultatami.

Wzrost złożoności i rozmiaru danych w podobny sposób wpływa na wzrost czasu wykonania zadań u wszystkich zespołów (wykresy zachowują podobny kształt). Nie zauważono znaczących różnic w szybkości TeamNewThreads, TeamConstThreads, TeamConstProcesses oraz TeamPool.

Wprawdzie równomierny podział pracy z pamięcią współdzieloną zwiększa skuteczność stosowania procesów, jednak pod względem teoretycznym procesy mogą okazać się nieopłacalne czasowo ze względu na czas tworzenia, konieczność rezerwowania nowych zasobów i potencjalne utrudnienia w komuniakcji (wątki mogą korzystać z zasobów tego samego procesu). Najkorzystniej przedstawiają się rozwiązania asynchroniczne, zoptymalizowane przez twórców i wspierane w bibliotece standardowej.

Przykładowe średni czas działania w milisekundach. Dla rozwiązań wielowątkowych liczono średnią z kilku wyników (podobnie w przypadku wykresów).

	Laptop		
	SameNumber	ShortNumber	LongNumber
TeamSolo	4598.8	2053.87	2253.37
ThreadNewThreads	4133.925	1860.555	1051.7087
TeamConstThreads	1842.3683	869.5477	1057.5487
TeamPool	1934.328	814.5598	919.4117
TeamNewProcesses	50309.8333	8798.7533	2170.0983
TeamConstProcesses	1869.3963	869.562	1065.3172
TeamAsync	718.805	234.757	260.581