Wydział Informatyki Politechniki Białostockiej Tworzenie Aplikacji Rozproszonych	
Sprawozdanie nr 1 Zadanie 1: HTCondor i DAGMan PS 1 Alan Bednarczyk Sławomir Romanowski	Prowadzący: mgr inż. Daniel Reska

Zadanie

Badanie skalowalności i podziału na podzadania

Przybliżanie wartości PI metodą MonteCarlo

Wartość Pi można przybliżyć probabilistycznie losując punkty (czyli 2 liczby będące ich współrzędnymi *x* i *y*) w kwadracie o boku 2*R* i sprawdzają, czy znajdą się one we wpisanym w ten kwadrat kole o promieniu *R*. Stosunek pola koła do pola kwadratu to:

(**PI** *R*^2) / (2*R*)^2 = **PI** / 4.

Jeżeli zatem obliczy się stosunek liczby wylosowanych punktów, które "wpadają" do koła do całkowitej liczby wylosowanych punktów (zakładając losowanie z rozkładu jednostajnego) i pomnoży się wynik przez 4, uzyska się przybliżenie **PI**.

Czym więcej wylosowanych punktów, tym lepsze przybliżenie.

Realizacja algorytmu w systemie HTCondor z użyciem mechanizmu DAGman

Napisz program realizujący zadaną liczbę (jako argument uruchomienia albo wczytywaną ze standardowego wejścia albo wczytywana z pliku o ustalonej nazwie) iteracji losowań punktów wg opisu powyżej. Ważne jest, aby generator liczb pseudolosowych mógł być inicjowany podaną wartością (wybór j.w.) - w przeciwnym wypadku przy jednoczesnym uruchomieniu wszystkie programy mogą generować te same sekwencje liczb pseudolosowych. Liczby losowań i "trafień" w koło powinny być zwracane jako wynik: zapisywane do pliku albo na standardowe wyjście. Program taki powinien być uruchamiany wielokrotnie z różnymi parametrami.

Napisz też program, który przetwarza wyniki z wielu uruchomień pierwszego programu i wylicza przybliżenie wartości PI.

Następnie przy użyciu mechanizmu HTCondor DAGMan zorganizuj prosty proces obliczeniowy zakładający wielokrotne uruchomienie pierwszego programu (losującego) a dopiero po ich zakończeniu uruchomienie drugiego programu agregującego wyniki. Różne parametry wejściowe mogą być przekazywane do programów losujących w pliku uruchomieniowym Condora (submit) w różny sposób: atrybuty input, arguments, initial_dir

(niektóre opcje wymagają transferu plików) z użyciem symboli $(Process) i/lub $(Cluster) oraz różnymi parametrami polecenia "Queue" (link).

Przeprowadź eksperyment z użyciem "rozsądnej" liczby iteracji (losowania punktów) algorytmu Monte Carlo obliczającego przybliżoną wartość PI. "Rozsądna ilość" oznacza dobranie ilości punktów tak, aby program nie działał zbyt długo (maks. kilka minut), ale jednocześnie aby czas obliczeń nie był porównywalny z czasem narzutu uruchamiania zadań w systemie HTCondor (w ostatniej instalacji ten narzut powinie być mały). Orientacyjnie: na komputerach w sali 224 w czasie ok. 1 minuty powinno dać się uruchomić algorytm na ilości punktów równej 3.750.000.000 (3.75 mld, implementacja jednowątkowa w C). Przeprowadź kilka prób w różnych konfiguracjach liczby uruchomionych programów i liczby iteracji (np. 10 programów po 10 mld iteracji vs 100 programów po 1 mld iteracji) i sformułuj wnioski.

Następnie zbadaj:

- jak problem się skaluje zwiększając liczbę uruchomień programu losującego (stała całkowita liczba iteracji),
- jak wpływa liczba iteracji na dokładność obliczenia (rosnąca całkowita liczba iteracji).

Badanie skalowalności

Uruchom zadanie ze stałą docelową liczbą iteracji w 1, 2, 4, 8, 16, 32 uruchomieniach programu losującego (w miarę wzrostu liczby uruchomień programu maleje liczba iteracji na uruchomienie aby zachować stałą całkowitą liczbę iteracji) . Pomiary całkowitego czasu zegarowego (*Wall-clock time*), całkowitego czasu procesora (*CPU time*), czasu działania poszczególnych procesów roboczych oraz przyspieszenia***** - wszystko w zależności od liczby węzłów roboczych umieść w tabelach i na wykresach. Wyniki krótko skomentuj (w szczególności odnieś się do idealnej liniowej skalowalności/przyspieszenia).

Badanie dokładności obliczeń

Uruchom zadanie z rosnącą całkowitą liczbą iteracji (czas nie będzie tu badany ale szybciej będzie użyć wielu uruchomień programu losującego). Następnie zbadaj:

1. jak liczba iteracji wpływa na dokładność obliczeń (różnica między teoretyczną wartością PI a wartością obliczoną).
2. czy dokładność ta jest stabilne dla zadanej liczby iteracji - tzn. czy kolejne uruchomienia dla stałej całkowitej liczby iteracji dają zbliżoną dokładność.

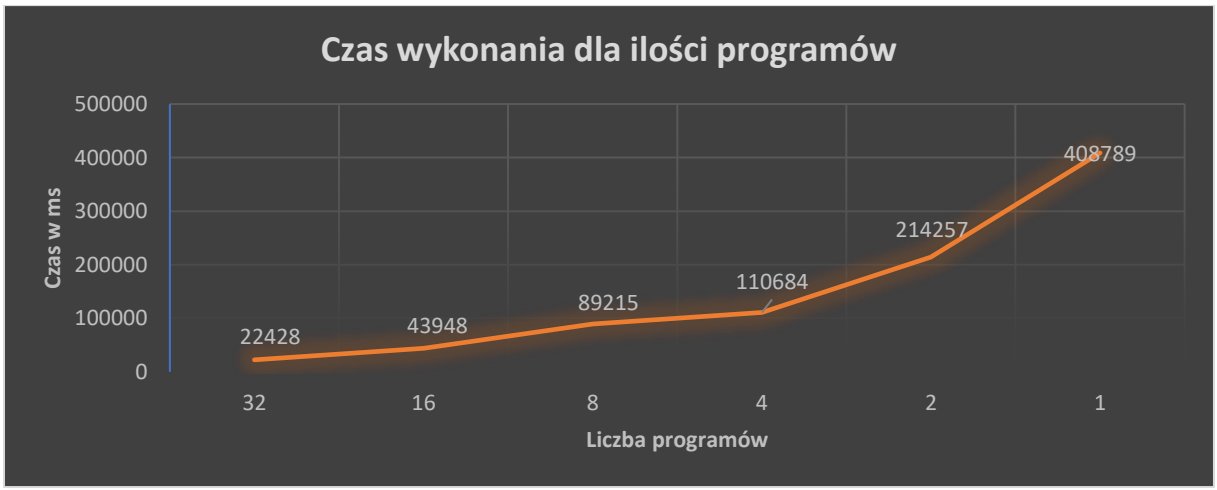
Uwaga: aby zapewnić unikalność sekwencji losowań zadbaj aby przy każdym uruchomieniu programu losującego użyć unikalnej wartości ziarna generatora liczb pseudolosowych.

Rozwiązanie:

Badanie skalowalności

Badanie przeprowadzono na łącznej liczbie iteracji 20 miliardów mierząc czas wykonywania każdego z programów.

Programy	Iteracje	Czas wykonania (ms)	Wartość Pi
32	625000000	22428	3.1415903944
16	1250000000	43948	3.1415894598
8	2500000000	89215	3.1415838814
4	5000000000	110684	3.1415925836
2	10000000000	214257	3.1415841434
1	20000000000	408789	3.1415934976

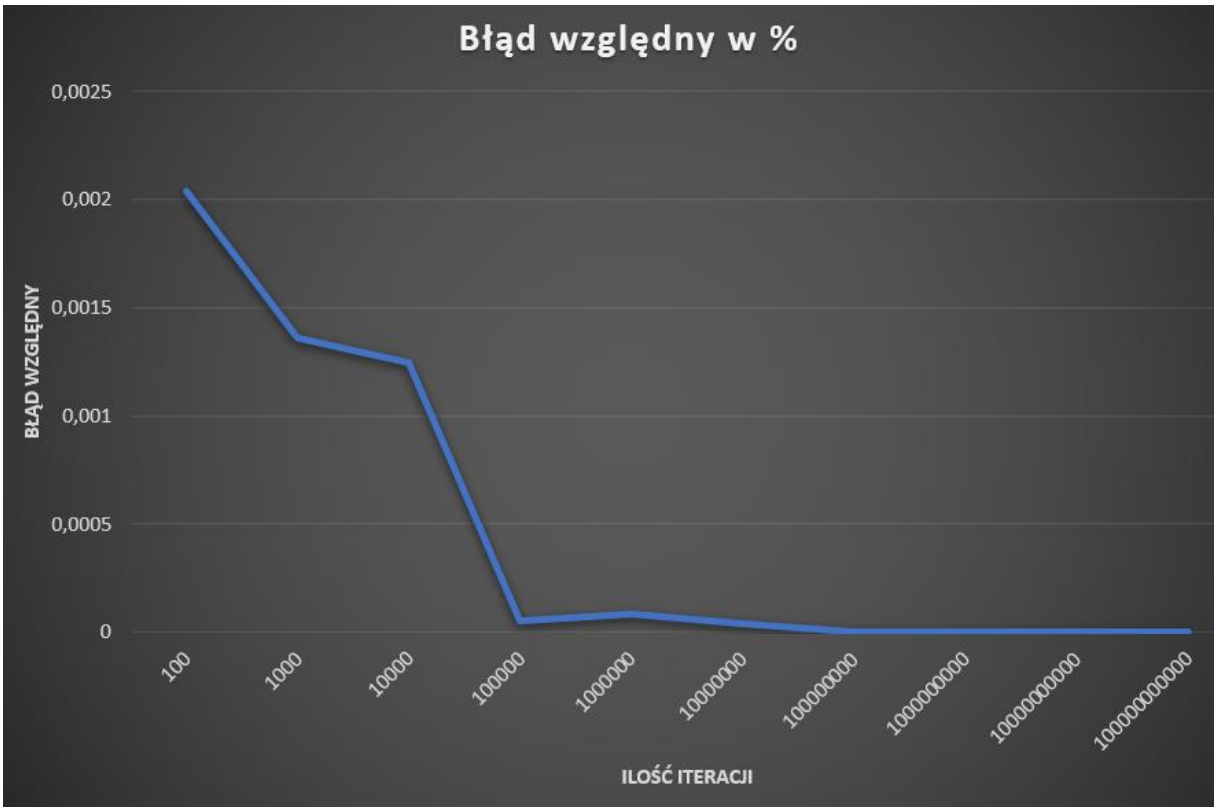


Wyniki przeprowadzonego badania wykazały, że oddelegowanie części zadań na inne węzły skraca czas ich wykonywania. Wraz z dwukrotnym wzrostem liczby węzłów – czas wykonania maleje **prawie** dwukrotnie, więc występuje przyspieszenie zbliżone do przyspieszenia liniowego, które jest przyspieszeniem idealnym.

Badanie dokładności obliczeń

Wartość pi do 15 miejsc po przecinku to 3,141592653589793. Wyniki i wspomnianą wartość zestawiono, by obliczyć wartość błędu względnego.

Liczba iteracji na każdy program	Uzyskany wynik	Błąd względny w %	Programy
100	3,148	0,20395217065741100000%	30
1000	3,145866667	0,13604606160497500000%	30
10000	3,13768	0,12454363188426500000%	30
100000	3,141434667	0,00502887995168091000%	30
1000000	3,141862533	0,00859053904495572000%	30
10000000	3,141719053	0,00402342879798445000%	30
100000000	3,141592311	0,00001091558193359540%	30
1000000000	3,141596013	0,00010694395837672700%	30
10000000000	3,14159161	0,00003322655847021850%	30
100000000000	3,141591479	0,00003738767527627430%	30



Przeprowadzone badania pozwalają wywnioskować, iż zwiększanie liczby iteracji prowadzi do skuteczniejszego wyliczania wartości π – błąd względny zmniejsza się tak jak pokazano to na wykresie powyżej, jednak przy coraz większej ilości iteracji mogą pojawić się nieoczekiwane rezultaty – pojawia się niewielki wzrost błędu względnego. Ponowienie obliczeń dla tych samych ilości iteracji pozwoliło na uzyskanie nieznacznie różniących się wyników w porównaniu do wyników pierwszego podejścia, dlatego można wysunąć wniosek, iż wzrost ilości iteracji zwiększa skuteczność obliczeń, jednak należy uważać na możliwe odstępstwa, co jasno pokazuje powyższy wykres.