Analiza i modelowanie wydajności obliczeń

Lab 1. Pomiar czasu w Linuxie/Unixie

Kolejne wykonywane kroki:

1. Utwórz nowy katalog *lab_01*, a w nim podkatalog, np. *c_solver*
2. W podkatalogu *c_solver* rozpakuj (*tar xvzf c_solver.tgz),* skompiluj (*make*) i uruchom (*./2dc_ac*) program z paczki *c_solver.tgz*
    1. Program domyślnie wykonuje jedną iterację solwera układu równań liniowych (algorytm jest bez znaczenia dla przeprowadzanych ćwiczeń). Czas wykonania programu można zmieniać przez zmianę liczby iteracji zadaną wartością zmiennej *Nstop* (linia 69 pliku *pp_main.c*) (i oczywiście rekompilację kodu). Dłuższe czasy wykonania mogą być przydatne do uzyskania sensownych rezultatów w punktach 3.1 i 3.2, domyślna wartość jest przygotowana do sprawnego przeprowadzania pomiarów w punktach 3.3, 3.4, 3.5, 3.6.
    2. **Przed przystąpieniem do pomiarów czasu wykonania przeczytaj zasady dokonywania pomiarów umieszczone na stronie przedmiotu jako "Zasady pomiaru czasu"**
        1. **Podstawową zasadą jest wielokrotne wykonanie pomiaru i odrzucenie przypadków z zaburzeniami**
3. Porównaj różne metody pomiaru czasu realizacji programu:
    1. pomiar czasu zegarkiem (stoperem)
    2. obserwacja danych prezentowanych przez polecenie *top* (w osobnym oknie, najlepiej z opcją *-d 1*) lub w dowolnym innym monitorze systemu
    3. wykorzystanie funkcji systemowych omawianych w następnych punktach – czas wypisywany przez aplikację *2dc_ac* w linijkach zaczynających się od *cputime* i *daytime*
        1. przeprowadzając pomiary w kolejnych punktach porównaj czasy uzyskane za pomocą innych narzędzi z czasami raportowanymi przez *2dc_ac* (dla tych samych uruchomień programu – zaobserwuj, które czasy są bardzo zbliżone)
    4. pomiar za pomocą komendy *time (time ./2dc_ac)* – można także użyć zazwyczaj dokładniejszej wersji */bin/time*
    5. pomiar z wykorzystaniem profilu wykonania kodu:
        1. upewnienie się, że kompilator dokona instrumentacji kodu: w pliku Makefile opcja kompilacji *-p* (lub *-pg*) do tworzenia pliku profilu (*mon.out* lub *gmon.out*)
        2. przeglądanie pliku profilu za pomocą polecenia *prof* (lub *gprof: gprof ./2dc_ac*)
    6. wykorzystanie liczników sprzętowych (*hardware counters*) udostępnianych przez polecenie *perf stat* (*perf stat ./2dc_ac*)
        1. *perf stat* zwraca czas wykorzystania procesora (*task clock* w milisekundach) oraz czas zewnętrzny (*elapsed time, wallclock time*) – czas zewnętrzny jest rozbity na czas użytkownika i czas systemowy (podobnie jak w poleceniu *time (/bin/time*)

    Zaobserwuj które z wyników są do siebie istotne zbliżone – wyciągnij wnioski dotyczące wiarygodności różnych form pomiaru czasu
    Zauważ, że mierzony czas może obejmować dodatkowe operacje związane z użyciem narzędzia (np. w przypadku *gprof*) oraz być dokonywany na zewnątrz kodu (jak *time* i *perf stat)* – przez co obejmuje także uruchomienie programu lub wewnątrz programu (jak w przypadku użycia wewnętrznego pomiaru za pomocą funkcji systemowych)
4. Przeprowadź kilka eksperymentów stosując *gprof* z różnymi opcjami optymalizacji (symbol *OPT* w pliku *Makefile*, wartości np. -O0, -O1, -O2, -O3 – znaczenie poziomów optymalizacji omawiane będzie na kolejnych wykładach i laboratoriach, **uzyskanie nowego pliku binarnego po zmianie symbolu *OPT* wymaga rekompilacji: *make clean; make***). Zaobserwuj, jak zmienia się czas wykonania (raportowany przez aplikację) oraz profil wykonania, produkowany przez *gprof.* (plik *gmon.out* jest tworzony przy każdorazowym uruchomieniu programu – w celu porównania różnych opcji optymalizacji należy uruchamiać konkretne wersje i bezpośrednio potem gprof, np. zapisując wyniki do różnych plików (*./2dc_ac ; gprof ./2dc_ac > wyniki_O0.txt*)

5. Pobierz paczkę "*time_measurements.tgz*" **na serwer Honorata lub na maszynę lokalną** do nowego podkatalogu, np. *pomiar_czasu*
6. Rozpakuj paczkę, skompiluj program w podkatalogu *time_measurements* korzystając z *make* (program korzysta z katalogu *../utd_time_unix*)
7. **Przeanalizuj funkcje pomiaru czasu w pliku *../utd_time_unix/uts_time.c* i ich sposób wykorzystania**
8. Uruchom program, przeanalizuj otrzymane wyniki
9. Dokonaj możliwych zmian, rozszerzeń, uzupełnień, mających na celu opanowanie technik pomiaru czasu wykonywania wybranych fragmentów kodu:
    1. **przeanalizuj udziały samej procedury generowania liczb losowych oraz późniejszej transformacji do zadanego przedziału w czasie wykonania**
        1. zmodyfikuj kod, tak aby były w nim 4 pętle:
            1. samo generowanie liczb losowych (sum+= rand())
            2. generowanie i rzutowanie do odpowiedniego przedziału
            3. generowanie i zapisywanie do zaalokowanej tablicy
            4. wczytywanie z tablicy i rzutowanie do odpowiedniego przedziału
        2. porównaj otrzymane czasy realizacji wszystkich 4 pętli i spróbuj wyciągnąć wnioski dotyczące czasu wykonywania rozmaitych operacji
            1. czy pojawia się sytuacja, kiedy wykonywanie pewnych operacji może być traktowane jako nie wpływające na całkowity czas wykonania (wykonanie tych operacji odbywa się jakby w tle wykonywania innych)?

---------- 4.0 ----------

Dalsze kroki:

1. Przetestuj czasy wykonania badanych pętli dla kodu skompilowanego przy użyciu kompilatora Intela *icc* (kompilator będzie często wykorzystywany w dalszych laboratoriach)
   (w aktualnej wersji kompilatora konieczne może być usunięcie polskich liter z napisów i włączenie opcji *-no-multibyte-chars*)

   Na serwerze Honorata w celu korzystania z kompilatora icc należy wykonać polecenie:
   ```
   source /opt/intel/oneapi/setvars.sh intel64
   [ Na potrzeby kolejnych laboratoriów najlepiej umieścić powyższą linijkę w pliku ~/.bashrc ]
   ```

2. Porównaj efekty zastosowanej w kodzie procedury transformacji do zadanego przedziału przy generowaniu liczb losowych z efektami często stosowanego przekształcenia z użyciem operacji % (modulo) (c = *a+ rand()%(b-a+1)* !) dla różnych ilości wygenerowanych liczb – porównaj czasy realizacji i losowość (np. przez porównanie średniej z wartością oczekiwaną przy liczbie prób zmierzającej do nieskończoności)
3. Dokończ zadania z Lab 0.

Sprawozdanie:

1. Zrealizowane kroki – w tym:
    1. opis wyników pomiarów czasu wykonania w p. 3 – wnioski dla każdej z metod (np. dotyczące dokładności możliwej do uzyskania, innych, poza czasem wykonania, informacji dostarczanych przez poszczególne metody pomiaru czasu, itp.)
    2. wydruki (z opisem i wnioskami) fragmentu wyników zwracanych przez "*flat profile*" `gprof` dla kilku najważniejszych funkcji kodu `2dc_ac` i dla co najmniej 2 wybranych opcji optymalizacji
    3. opis procedur zwracających czas CPU i zegara zewnętrznego
2. Wnioski:
    1. wpływ poziomu optymalizacji na czas wykonania poszczególnych funkcji, czy optymalizacja podobnie wpływa na czas wykonania różnych funkcji?
    2. czasy realizacji wybranych operacji, różnice we wskazaniach zegarów różnych typów dla różnych rodzajów operacji, uzyskiwanie liczb losowych z zadanego przedziału, wykorzystywanie zasobów systemowych do pomiaru czasu wykonania fragmentów kodu.