Architektury systemów komputerowych

Lista zadań nr 13

Na zajęcia 10 czerwca 2021

UWAGA! W trakcie prezentacji należy być gotowym do zdefiniowania pojęć oznaczonych **wytłuszczoną** czcionką.

Zadanie 1. Zapoznaj się z [1, §5.11.2] i na podstawie [1, §5.7.1] wyjaśnij jak procesor przetwarza skoki warunkowe. W jakim celu procesor używa **predyktora skoków**? Co musi zrobić, jeśli skok zostanie źle przewidziany? Które skoki warunkowe warto zoptymalizować do instrukcji «cmov»? Posługując się narzędziem godbolt[1] z opcją «-O1» przepisz poniższy kod tak, żeby w linii 4 kompilator nie użył skoku warunkowego.

```
1 void merge1(long src1[], long src2[], long dest[], long n) {
2   long i1 = 0, i2 = 0;
3   while (i1 < n && i2 < n)
4     *dest++ = src1[i1] < src2[i2] ? src1[i1++] : src2[i2++];
5 }
```

Zadanie 2 (2). Pobierz ze strony przedmiotu archiwum «ask21_lista_13.tgz», rozpakuj je i zapoznaj się z kodem źródłowym w pliku «dictionary.c». Powtórz eksperyment z [1, §5.14], tj.: uruchom program, poczekaj na wyniki **profilowania**, a następnie na podstawie wydruku wskaż kandydata do optymalizacji. Zmień dokładnie jeden z argumentów przekazywanych do polecenia «make gprof»: SIZE=N, HASH=0/1/2, FIND=0/1/2, LOWER=0/1, QUICK=0/1. Odpowiednia opcja zmieni rozmiar tablicy mieszającej N lub ustali wersję implementacji jednej z kluczowych funkcji w programie. Wyjaśnij co zmieniło przekazanie danej opcji. Powtarzaj powyższe kroki tak długo, aż uzyskasz najkrótszy czas wykonania programu.

UWAGA: Należy być przygotowanym do szczegółowego wyjaśnienia wydruku programu profilującego!

Zadanie 3. Dla optymalnej konfiguracji z poprzedniego zadania uruchom polecenie «make callgrind». Następnie wczytaj plik wynikowy «callground.out» przy pomocy nakładki graficznej «kcachegrind». Wyświetl **graf wywołań funkcji**, a następnie zidentyfikuj wywołania biblioteki standardowej języka C, w których program spędza najwięcej czasu. Wyjaśnij do czego służą te funkcje. Jak można byłoby zoptymalizować miejsca, w których korzystamy z tych funkcji?

Zadanie 4. Przy pomocy polecenia «cat /proc/iomem» wydrukuj mapę **fizycznej przestrzeni adresowej** swojego komputera. Zidentyfikuj w niej pamięć operacyjną oraz pamięć należącą do urządzeń wejścia-wyjścia (np. karty graficznej). Następnie ściągnij repozytorium dwks/pagemap[2] i skompiluj program «pagemap2». Następnie zaprezentuj działanie mechanizmu **translacji adresów**. Przy pomocy polecenia «pagemap2 pid» wyświetl zawartość tablicy stron procesu o numerze «pid». Przetłumacz kilka adresów wirtualnych posługując się **numerem ramki** (ang. *page frame number*). Wskaż region pamięci w mapie fizycznej przestrzeni adresowej, gdzie trafił przetłumaczony adres. Zakładamy, że rozmiar strony to 4096 bajtów.

Wskazówka: Polecenia należy wykonywać z uprawnieniami administratora.

Zadanie 5. Przy pomocy polecenia «ps -e -o pid,rss,vsz,cmd» wydrukuj listę wszystkich procesów wraz z zadeklarowanym rozmiarem **wirtualnej przestrzeni adresowej** i zbioru rezydentnego, odpowiednio vsz i rss. Użyj zdobytych danych do obliczenia rozmiaru pamięci wirtualnej używanej przez wszystkie procesy. Na podstawie wydruku polecenia «free» określ bieżące użycie i całkowity rozmiar pamięci fizycznej. Wyjaśnij w jaki sposób **stronicowanie na żądanie** (ang. *demand paging*) i **współdzielenie pamięci** pozwalają systemowi operacyjnemu na oszczędne zarządzanie pamięcią fizyczną.

[1] https://godbolt.org
[2] https://github.com/dwks/pagemap

Zadanie 6. Ponownie przyjrzymy się wydrukowi z polecenia «`ps`» i «`free`». Wskaż bieżące użycie i całkowity rozmiar **pamięci wymiany** (ang. *swap*). Jakich technik system operacyjny używa do aproksymowania **zbioru roboczego** (ang. *working set*) procesu przy pomocy **zbioru rezydentnego** (ang. *resident set*). Kiedy system operacyjny używa wymiany do **pamięci drugorzędnej**? Co dzieje się z systemem, jeśli łączny rozmiar zbiorów roboczych wszystkich procesów jest większy niż rozmiar pamięci fizycznej?

Wskazówka: Pamiętaj, że pamięć wirtualna jest programową realizacją pamięci podręcznej dla stron.

Literatura

[1] „Computer Systems: A Programmer's Perspective"
Randal E. Bryant, David R. O'Hallaron; Pearson; 3rd edition, 2016