Program 1

Pomiar przeprowadziłem na swoim domowym komputerze. Wykorzystałem następujące nośniki pamięci:

1. Dysk wbudowany w komputer - Hitachi Travelstar 5K750 - EXT4
2. Dysk zewnętrzny - Samsung S2 Portable - FAT
3. Pamięć typu flash - Kingston DataTraveler 2.0 - FAT

Do przeprowadzenia pomiaru wykorzystałem poniższy program. Ilość operacji dyskowych wyznaczałem eksperymentalnie w taki sposób, by czas pomiaru był obrobinę niższego rzędu niż wiek wszechświata. W przypadku zapisów dane nie były generowane losowo gdyż funkcja rand() ma skłonność do zawieszania się na dłuższą chwilę (nawet kilka sekund) co psuło wyniki pomiarów. Zamiast tego na dysk zrzucałem dopiero co odczytany blok.

```
1  #include <sys/types.h>
2  #include <sys/stat.h>
3  #include <fcntl.h>
4  #include <unistd.h>
5  #include <stdlib.h>
6  #include <stdio.h>
7  #include <string>
8  #include <time.h>
9
10 using namespace std;
11
12 // Ustawienia pliku, który będzie testowany
13 const int rozmiar_bloku = 8 * 1024; // 8 KB
14 const int bloki = 1024 * 10; // plik z danymi o rozmiarze 80 MB
15 string plik = "test.dat"; // ścieżka do pliku
16
17 // Ustawienia testowania
18 const int ile_odczytow = 1000000; // Ile odczytów (w blokach)
19 const int ile_zapisow  = 10000;   // Ile zapisów
20 const int max_operacji = 10; // Ile bloków maksymalnie można odczytać
21
22 // Zmienne globalne
23 void *bufor = NULL; // Bufor wyrównanej pamięci o rozmiarze bloku
24
25 // Wypełnia bufor losowymi danymi
26 void losuj_bufor(int ile = 1)
27 {
28     int *tab = (int *) bufor;
```

```
29      for (int i = 0; i < rozmiar_bloku * ile / sizeof(int); ++i)
30          tab[i] = rand();
31  }
32
33  // Tworzy plik z danymi testowymi, na nim będą prowadzone operacje
34  void generuj_dane()
35  {
36      int fd = open(plik.c_str(), O_RDWR | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
37
38      for (int i = 0; i < bloki; ++i)    {
39          losuj_bufor();
40          write(fd, bufor, rozmiar_bloku);
41      }
42
43      close(fd);
44  }
45
46  void test(int fd, int odczyty, int zapisy, int ile = 1)
47  {
48      printf(" Ustawienia testu: %d odczytow %d zapisow %d blokow na raz\n",
49             odczyty, zapisy, ile);
50
51      posix_memalign(&bufor, rozmiar_bloku, rozmiar_bloku * ile);
52
53      struct timespec start, end;
54      clock_gettime(CLOCK_MONOTONIC_RAW, &start);
55
56      while (odczyty + zapisy > 0) {
57          int blok = rand() % (bloki - ile + 1);
58          int operacja = rand() % (odczyty + zapisy);
59
60          if (operacja < odczyty) {
61              lseek(fd, blok * rozmiar_bloku, 0);
62              read(fd, bufor, rozmiar_bloku * ile);
63
64              odczyty -= ile;
65          } else {
66              //generuj_dane(ile); // Rzecz dyskusyjna
67
68              lseek(fd, blok * rozmiar_bloku, 0);
69              write(fd, bufor, rozmiar_bloku * ile);
70
71              zapisy -= ile;
72          }
```

```
73      }
74
75      clock_gettime(CLOCK_MONOTONIC_RAW, &end);
76      uint64_t delta = (end.tv_sec - start.tv_sec) * 1000000 + (end.tv_nsec - start.tv_nsec) / 1000;
77
78      printf(" Czas testu: %u\n", delta);
79
80      free(bufor);
81  }
82
83  int main()
84  {
85      // Inicjalizacja
86      srand(time(0));
87      posix_memalign(&bufor, rozmiar_bloku, rozmiar_bloku);
88
89      printf("Generowanie danych testowych\n");
90      generuj_dane();
91      free(bufor);
92
93      // Testy operacji jednoblokowych
94      printf("1B - Test z buforowaniem systemu plików\n");
95      int fd = open(plik.c_str(), O_RDWR);
96      test(fd, ile_odczytow, ile_zapisow);
97      close(fd);
98
99      printf("1B - Test z bezpośrednim dostępem do dysku\n");
100     fd = open(plik.c_str(), O_RDWR | O_DIRECT);
101     test(fd, ile_odczytow / 1000, ile_zapisow / 1000);
102     close(fd);
103
104     // Testy operacji wieloblokowych
105     printf("WB - Test z buforowaniem systemu plików\n");
106     fd = open(plik.c_str(), O_RDWR);
107     test(fd, ile_odczytow, ile_zapisow, max_operacji);
108     close(fd);
109
110     printf("WB - Test z bezpośrednim dostępem do dysku\n");
111     fd = open(plik.c_str(), O_RDWR | O_DIRECT);
112     test(fd, ile_odczytow / 1000, ile_zapisow / 1000, max_operacji);
113     close(fd);
114
115     // Koniec pracy
116     bufor = NULL;
```

```
117 }
```

Wyniki wraz z przyjętymi parametrami obrazują poniższe tabele. Założyłem, że na każde 100 odczytanych bloków przypada 1 zapisany blok. Operacja do wykonania w danej chwili jest wybierana losowo.

Odczyt z buforowaniem systemu plików:

Nośnik	Odczyty	Zapisy	Po ile bloków	Czas testu (μs)	Średni czas operacji (μs)
Hitachi	1000000	10000	1	2067758	2.047
Samsung	1000000	10000	1	1640246	1.624
Kingston	1000000	10000	1	1650492	1.634
Hitachi	1000000	10000	10	1507583	1.492
Samsung	1000000	10000	10	1428341	1.414
Kingston	1000000	10000	10	1535337	1.520
Hitachi	1000000	10000	100	1447951	1.433
Samsung	1000000	10000	100	1418917	1.404
Kingston	1000000	10000	100	1354455	1.341

Odczyt z bezpośrednim dostępem do dysku:

Nośnik	Odczyty	Zapisy	Po ile bloków	Czas testu (μs)	Średni czas operacji (μs)
Hitachi	1000	10	1	14030002	13891.091
Samsung	1000	10	1	12211941	12091.030
Kingston	1000	10	1	49592789	49101.771
Hitachi	10000	100	10	12172208	1205.169
Samsung	10000	100	10	26652850	2638.896
Kingston	10000	100	10	47006670	4654.125
Hitachi	10000	100	100	4035586	399.562
Samsung	10000	100	100	15264079	1511.294
Kingston	10000	100	100	18797234	1861.112

W przypadku odczytu buforowanego wielkość odczytywanego obszaru nie wpływa przesadnie na uzyskane czasy. Obserwowana różnica jest prawdopodobnie efektem mniejszej ilości wywołań systemowych i mniejszego narzutu związanego z przełączaniem kontekstu przy dłuższych odczytach.

Zależności dotyczące odczytu bezpośredniego wydają się być nietrywialne. Pamięć flash ma zdecydowanie najgorsze osiągi, przy odczycie jednoblokowym aż 4 razy gorsze od pozostałych typów pamięci. Zwiększenie ilości odczytywanych bloków 10 razy daje odpowiednio 12-krotne i 2-krotne przyspieszenie. Inna rzecz, która rzuca się w oczy to ok. 4-krotna różnica między wbudowanym dyskiem a pamięciami USB przy odczycie 100-blokowym. To może być słabość systemu plików (FAT vs EXT4), ale też skutek uboczny dużej ilości danych przesyłanych przez kabel USB służących do samego zarządzania połączeniem (nie niosących żadnych istotnych informacji). Niezależnie od tego wszystkiego, różnica między dostępem buforowanym i bezpośrednim jest powalająca.

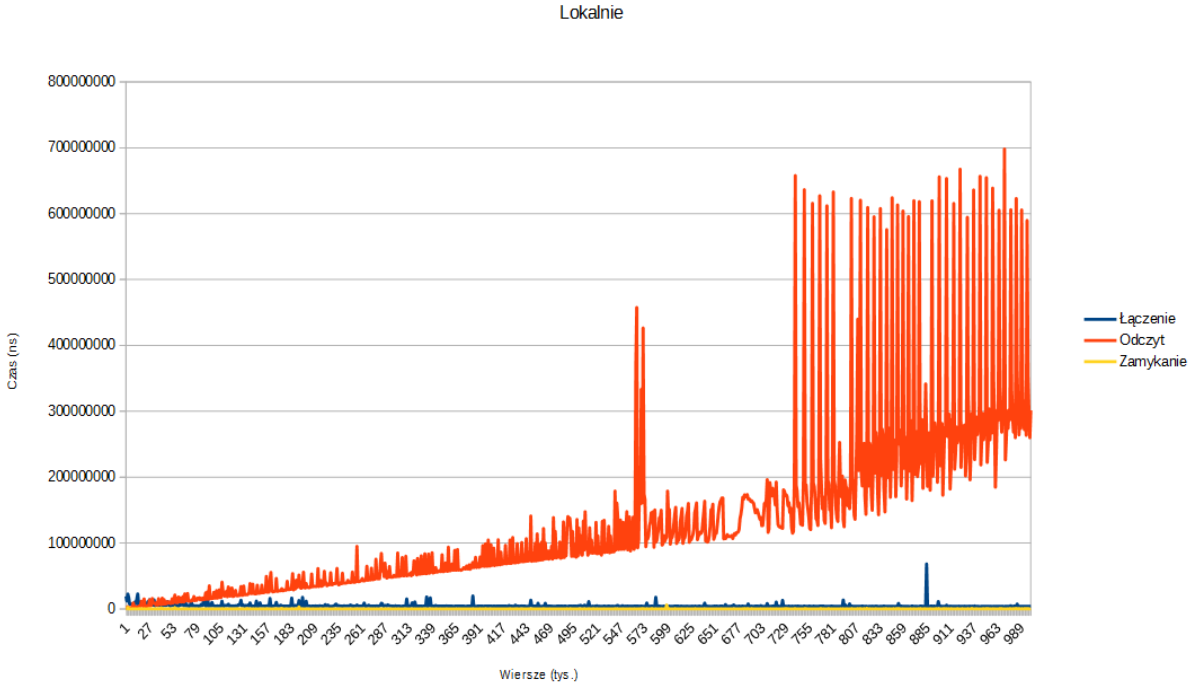
Program 2

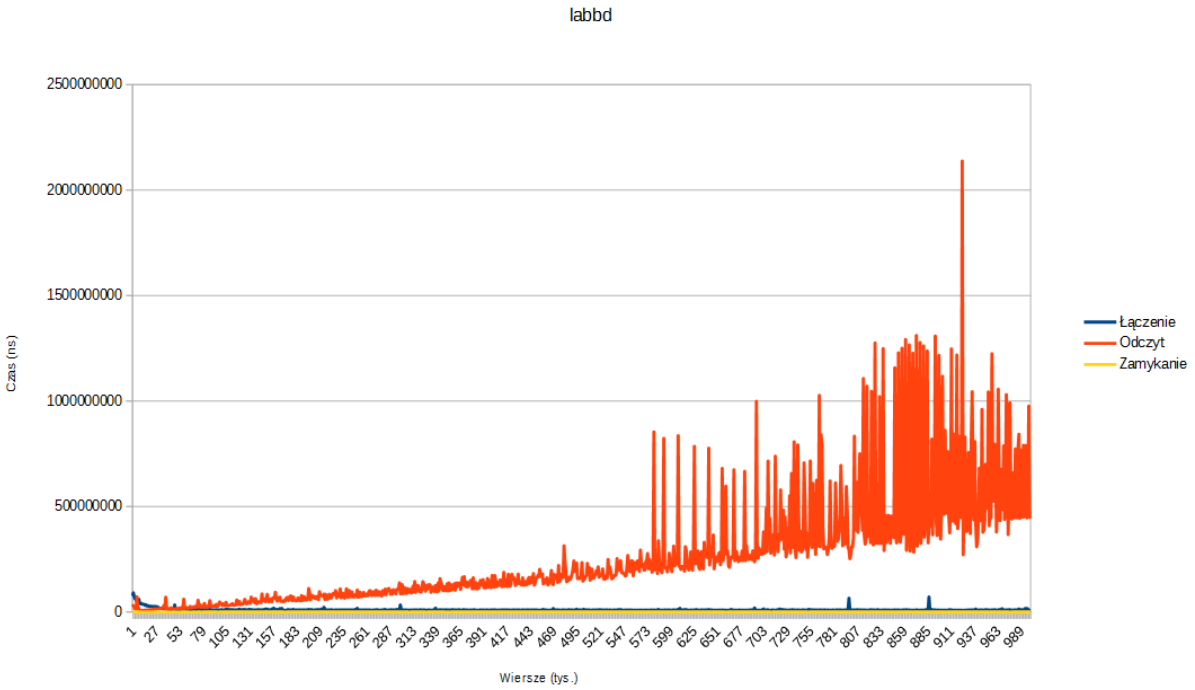
Testy szybkości nawiązywania połączenia wykonałem na bazie danych MySQL 5.1.51. Do testów lokalnych wykorzystałem domowy komputer, do testów zdalnych serwer labdb dostępny ze students. Testowałem zwyczajne pobieranie danych z tabel w zależności od ilości wierszy. Chodziło mi tutaj o stwierdzenie, powyżej jakiej ilości wierszy czas nawiązywania połączenia jest pomijalny w stosunku do wykonania reszty zapytania. W swoim programie napisanym w języku Java wykonuję 1000 iteracji testów, za każdym razem zwiększając obciążenie o 1000 wierszy. Oto kod programu:

```
1  import java.sql.*;
2
3  public class MysqlTest {
4      private static final int krok = 1000;
5      private static final int obroty = 1000;
6
7      public static void dodaj() throws SQLException
8      {
9          Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?user=root");
10         Statement statement = connection.createStatement();
11         statement.execute("CREATE TABLE IF NOT EXISTS test (id int) ENGINE = MEMORY;");
12
13         for (int i = 0; i < krok; ++i)
14             statement.execute("INSERT INTO test VALUES (42);");
15
16         connection.close();
17     }
18
19     public static void test(int iteracja) throws SQLException
20     {
21         long start = System.nanoTime();
22
23         Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/test?user=root");
24         long opened = System.nanoTime();
25
26         // Wykonanie zapytania
27         Statement statement = connection.createStatement();
28         statement.executeQuery("SELECT * FROM test;");
29         long done = System.nanoTime();
30
31         // Zamykanie połączenia
32         connection.close();
33         long closed = System.nanoTime();
34
```

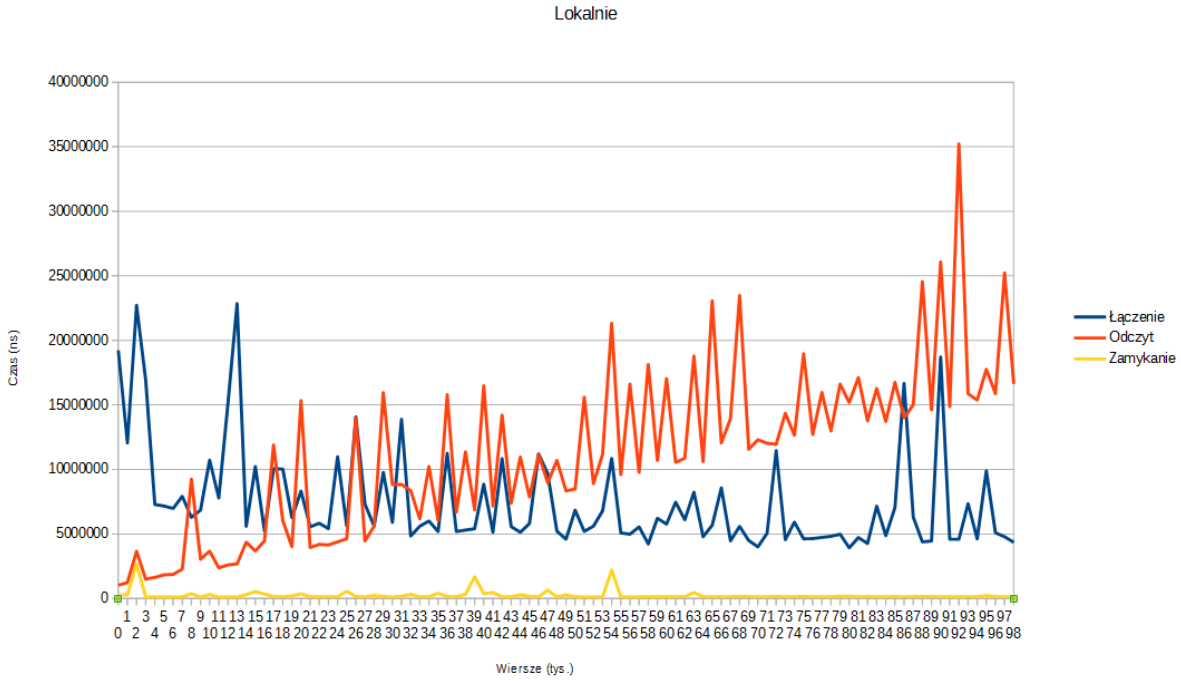
```
35          // Format danych:
36          // Iteracja; czas otwarcia; czas zapytania; czas zamykania
37          System.out.format("%d;%d;%d;%d;\n", iteracja, opened - start,
38                  done - opened, closed - done);
39      }
40
41      public static void main(String[] args) {
42          for (int i = 0; i < obroty; ++i) {
43              try {
44                  dodaj();
45                  test(i);
46              } catch(SQLException e) {
47                  System.out.println("Error" + e);
48              }
49          }
50      }
51
52  }
```

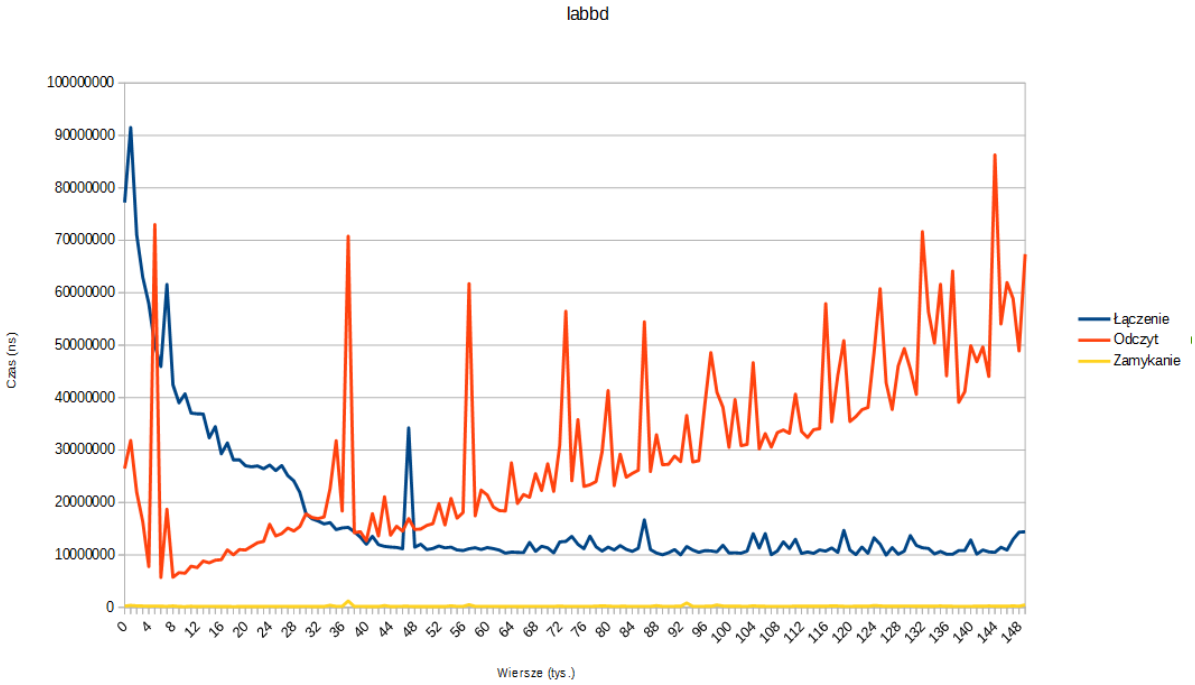
Ze względu na swoją objętość wyniki są zilustrowane na poniższych wykresach oraz dostępne w dołączonych plikach o nazwie wynikilocal.txt i wynikilabdb.txt.





Jak widać, czas łączenia jest znikomy w porównaniu do pobierania danych i niewiele z wykresu wynika. Zobaczmy teraz dokładniejsze wykresy:





Wykresy są dość nieregularne, zdarzają się bardzo duże odchylenia od typowego czasu odczytu. Na komputerze lokalnym wynoszą one zazwyczaj dwukrotność oczekiwanego czasu, na komputerze zdalnym też, ale tutaj maksimum wynosi czterokrotność. Oczekiwany czas odczytu miliona wierszy wynosi ok. 0,3 s na komputerze lokalnym i 0,6 s na labbd.

Ciekawsze są wyniki z mniejszego przedziału. W przypadku łączenia z zdalnym hostem użyłem dłuższej serii by uwypuklić ciekawą zależność. Czas pierwszego łączenia zajmuje aż 90 ms by spaść do 10 ms po wykonaniu 50-ciu iteracji testu. Wygląda to tak, jakby baza danych musiała się „rozgrzać". Być może ma to jakiś związek z dynamicznie regulowaną ilością procesów wykonujących zapytania, socketów, wątków albo jakimś cache. Czas łączenia z lokalnym hostem zajmuje jakieś 5 do 10 ms.

Odpowiadając na pytanie postawione we wstępie do rozdziału, wystarczy pobrać 50 wierszy z tabeli (właściwie to liczb typu integer) by czas łączenia z bazą danych przestał dominować. W ramach ciekawostki zmierzyłem też czas zamykania połączenia. Wygląda na to, że nie są stosowane tutaj żadne wyrafinowane protokoły.