Sprawozdanie - Dynamiczna Alokacja Pamięci

Kamil Kuczaj

6 marca 2016

1 Wstęp

Podanym zadaniem był pomiar czasu wykonywania alokacji pamięci dla tablicy dynamicznej elementów typu *int*. Należało wykonać pomiary zapisu: 10^1, 10^3, 10^5, 10^6, 10^9 danych oraz stworzyć specjalną klasę, która będzie zarządzać zapisem danych oraz alokacją pamięci. Poniżej listing pliku nagłówkowego klasy.

```
class pojemnik  {
 private:
  int *elementy;
  unsigned long rozmiar;
  unsigned long indeks;

 public:
  pojemnik(int x=10);
  ~pojemnik();
  bool czy_pelne();
  void zapisz(int dana);
  int zwieksz_rozmiar();
  unsigned long zwroc_rozmiar();
  void wypisz(); // w celu debuggowania
};
```

Rysunek 1: Listing pliku *pojemnik.h*

2 Specyfikacja komputera

Wersja kompilatora *g++*	4.8.4
System	Ubuntu 14.04.4
Procesor	Intel Core i5 2510M 2.3 GHz
Pamięć RAM	8 GB DDR3 1600 MHz
Rozmiar zmiennej *int*	4 bajty

3 Wybrane metody alokacji pamięci i wyniki

1. Zwiększanie rozmiaru tablicy o 1 element

Rozmiar	Ilość elementów
10	1 μs
1 000	$1,63$ ms
100 000	$18,483$ s
1 000 000	$2070,91$ $s = 34,5min$
1 000 000 000	zbyt długo

2. Zwiększanie rozmiaru tablicy o 100 elementów

Rozmiar	Ilość elementów
10	1 μs
1 010	33 μs
100 010	$206,995$ ms
1 000 010	$19,7127$ s
1 000 000 000	zbyt długo

3. Zwiększanie rozmiaru tablicy o 10000 elementów

Rozmiar	Ilość elementów
10	1 μs
10 010	14 μs
100 010	$3,097$ ms
1 000 010	$227,339$ ms
1 000 000 000	zbyt długo

4. Podwajanie rozmiaru tablicy

Rozmiar	Ilość elementów
10	2 μs
1 280	25 μs
163 840	$2,381$ ms
1 310 720	$20,657$ ms

Przy większych ilościach danych następował wyciek pamięci, gdyż potrzebna była zbyt duża ilość pamięci danych:

$$1310720 * 2^{10} = 1342177280 \ (> 10^9)$$

$$\frac{1310720 * 2^{10}}{2^{20}} * 4 = 5\ 120\ MB$$

Program więc potrzebował 5 GB pamięci podręcznej, czego mój system nie mógł zapewnić.

5. Potrajanie rozmiaru tabliicy

Rozmiar	Ilość elementów
10	1 μs
2 430	17 μs
196 830	$1,375$ ms
1 771 470	$15,127$ ms

   Przy większych ilościach danych następował wyciek pamięci, gdyż potrzebna była zbyt duża ilość pamięci danych tak jak w poprzednio wyjaśnionym przypadku.

6. Zwiększanie rozmiaru tablicy o 50%

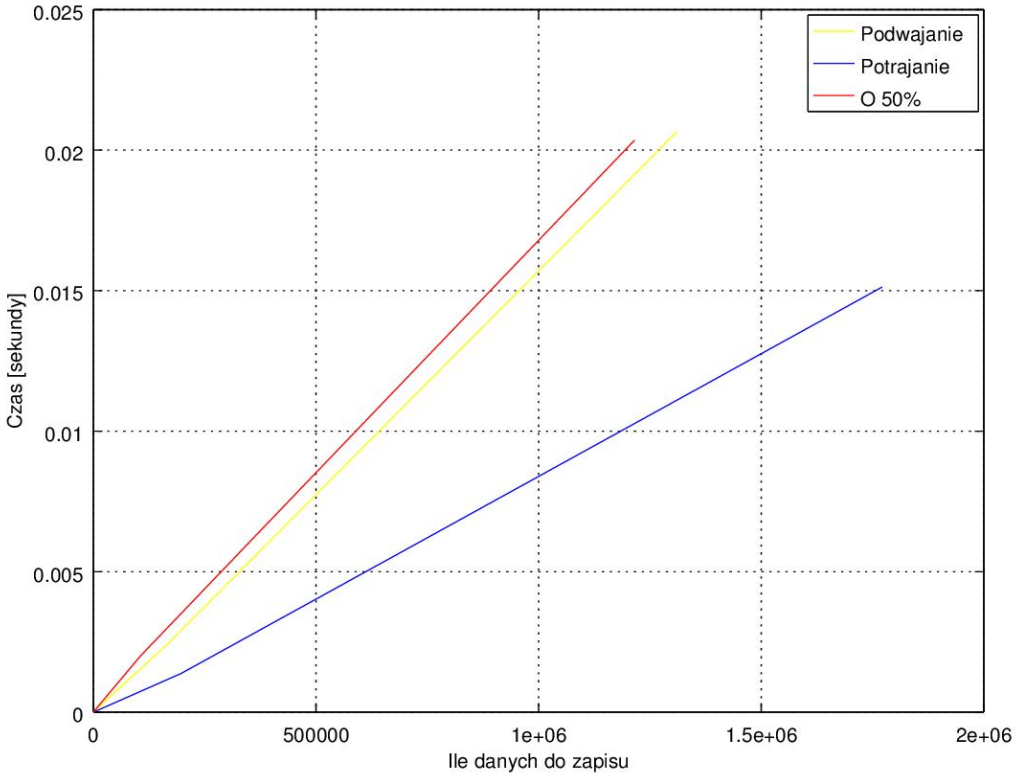
Rozmiar	Ilość elementów
10	1 μs
1 234	29 μs
106 710	$2,017$ ms
1 215 487	$20,354$ ms

   Przy większych ilościach danych następował wyciek pamięci, gdyż potrzebna była zbyt duża ilość pamięci danych tak jak w poprzednio wyjaśnionym przypadku.

4 Wnioski

Choć najbardziej optymalną metodą zwiększania rozmiaru tablicy jest jej podwajanie, to przy bardzo dużych danych może się skończyć brakiem pamięci fizycznej wynikającej z ograniczeń sprzętowych (zbyt mała ilość pamięci RAM komputera). To samo tyczy się potrajania rozmiaru tablicy oraz zwiększania jej rozmiaru o 50%. Niemniej jednak są to bardzo wydajne metody.

Żadna z metod dodawania elementów, nawet w ilościach takich jak 10 000 nie dorównują wspomnianym powyżej metodom. Poniżej zamieściłem wykres ilustrujący szybkość zapisu trzech najwydajniejszych metod.



Rysunek 2: Wyraźnie widać, że najlepszą metodą w eksperymencie okazało się potrajanie rozmiaru tablicy