Teoria Współbieżności

Porównanie wzorca projektowego Active Object z mechanizmem monitora dla problemu producenta i konsumenta.

Jakub Synowiec

25.05.2015r.

1. Cele zadania:

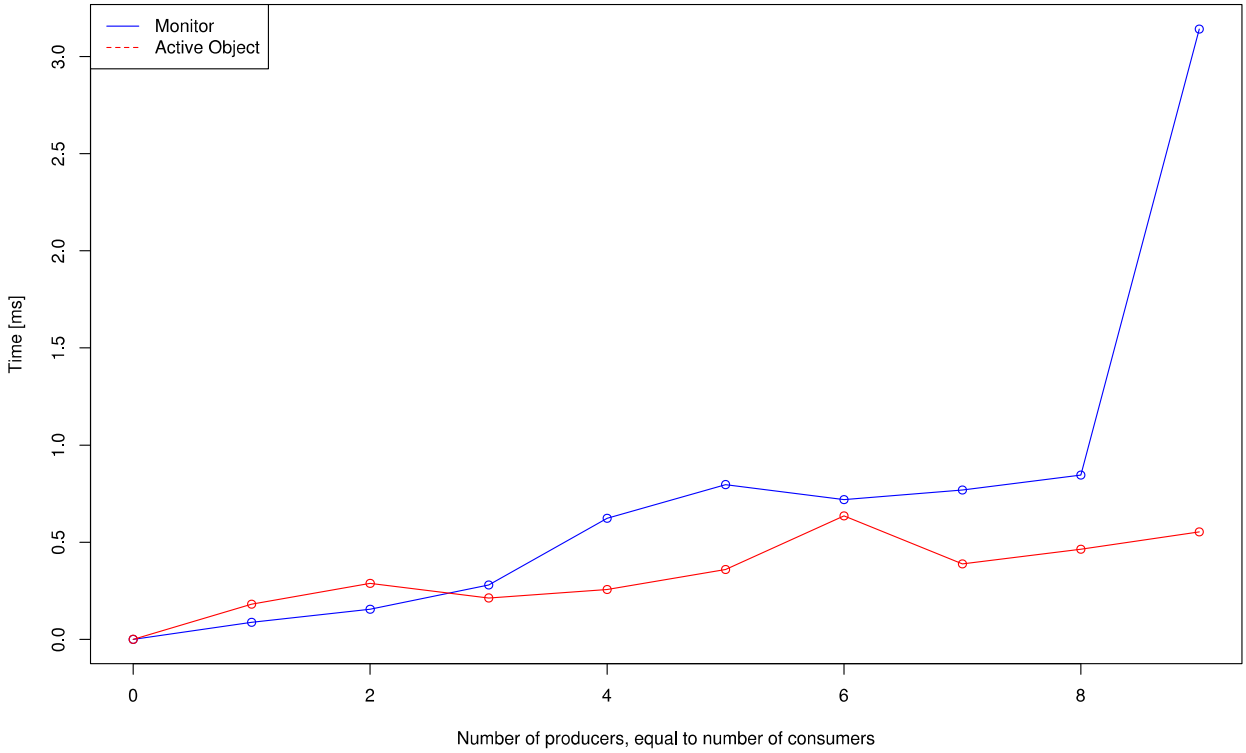
Porównanie szybkości działania wzorca projektowego Active Object oraz mechanizmu monitora w technologii Java dla problemu producenta i konsumenta z asynchronicznym pobieraniem i wstawianiem różnych wielkości porcji danych.

2. Wstęp:

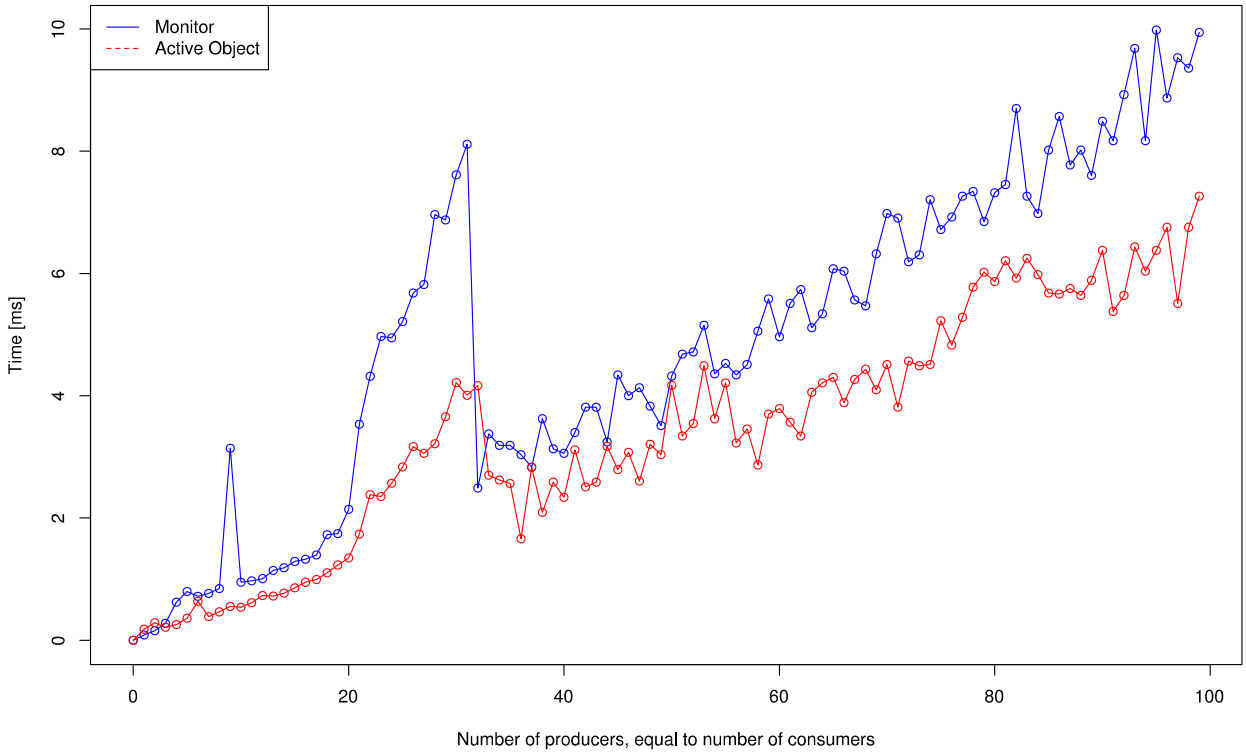
Porównanie zostało dokonane na laptopie z procesorem czterordzeniowym Intel Core i7 4710MQ, z częstotliwością pracy rdzenia 2.5 GHz, 12GB pamięci ram. System operacyjny to Ubuntu 14.04 x64. Użyta w doświadczeniu wersja Javy to: Java(TM) SE Runtime Environment (build 1.8.0_45-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode) . Pomiar danych został zapisany do pliku typu CSV. Za pomocą języka R dane z tego pliku zostały pobrane, policzone zostały średnie dla odpowiedniej ilości producentów i konsumentów z wielu prób, oraz zostały wyrysowane trzy wykresy zawierające funkcje czasu pracy programu w zależności od ilości producentów bądź konsumentów na różnych przedziałach wartości.

3. Przebieg ćwiczenia:

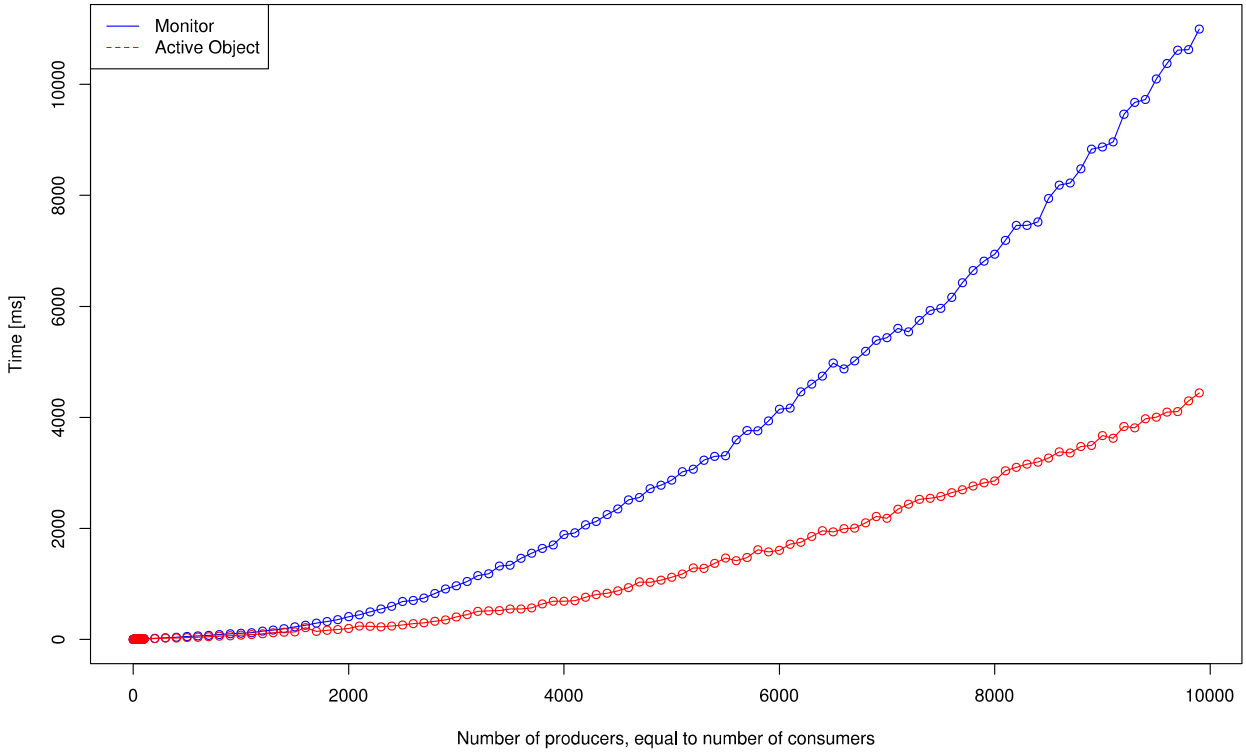
Kod źródłowy implementacji Active Object oraz odpowiedniego programu z użyciem mechanizmu monitora został umieszczony w załączniku. Metoda porównywania wydajności powyższych mechanizmów wymagała jasnego określenia warunków końca działania pojedyńczej próby. Określamy więc próbę jako cykl wstawień do bufora, bądź cykl pobrań z bufora (odpowiednio dla producenta i konsumenta) losowej porcji elementów tak, aby sumaryczna ilość dodanych elementów we wszystkich porcjach wynosiła N, zarówno dla producenta i konsumenta; gdzie N – liczba zadana przy określeniu problemu. Czas mierzony w doświadczeniu jest czasem pomiędzy początkiem tworzenia wątków będących producentami bądź konsumentami, a końcem działania ostatniego z tych wątków. Wielkość bufora na elementy wstawiane lub pobierane oraz wielkość kolejki oczekujących wątków w implementacji Active Object jest obliczana na podstawie odpowiednio informacji o sumarycznej ilości pobranych elementów (która jest jednocześnie maksymalną) oraz ilości wątków producenta i konsumenta. Zmienną w problemie jest liczba wątków producenta, będąca równej liczbie wątków konsumenta. Dla tak zadanego problemu i zmiennych wygenerowano trzy wykresy:



Rys 1. Przedział 0 – 10 wątków



Rys 2. Przedział 0 – 100 wątków



Rys 3. Przedział 0 – 10000 wątków

Warto zauważyć, że dla przedziału 0 – 100 wątków przeprowadzono testy dla każdej całkowitej liczby wątków z tego przedziału, natomiast powyżej tego przedziału testowano szybkość implementacji co 100 wątków. Dla mniejszych przedziałów wykonano również większą liczbę pomiarów, przez co po uśrednieniu wyniku uzyskano mniejszy błąd pomiaru.

4. Wnioski:

a) Active Object jest szybszym rozwiązaniem zadanego problemu dla większej liczby procesów, mechanizm monitora wypadł szybciej jedynie dla 2, 4 wątków oraz dla szczególnych przypadków. Od 60 wątków producenta i 60 wątków konsumenta widać już znaczącą różnicę na korzyść Active Object.
b) Wraz ze wzrostem wątków widać, że przyrost w czasie działania jest większy dla mechanizmu monitora niż Active Object. Dla około 6000 procesów konsumenta i 6000 procesów producenta czas działania dla implementacji Active Object jest dwukrotnie krótszy od mechanizmu monitora.