

Sztuka tworzenia wydajnego kodu : przewodnik po zaawansowanych technikach wykorzystywania sprzętu i kompilatorów / Fedor G. Pikus. – Gliwice, copyright © 2022

Spis treści

O autorze	9
O recenzencie	10
Przedmowa	11
Część I. Fundamenty wydajności	17
Rozdział 1. Wydajność i współbieżność – wprowadzenie	19
Dlaczego należy brać pod uwagę wydajność?	20
Dlaczego wydajność ma znaczenie?	22
Co rozumiemy przez wydajność?	24
Wydajność jako przepustowość	25
Wydajność jako pobór mocy	26
Wydajność w przypadku aplikacji czasu rzeczywistego	26
Wydajność w zależności od kontekstu	27
Ocenianie, szacowanie i przewidywanie wydajności	28
Poznanie zagadnienia dużej wydajności	30
Podsumowanie	32
Pytania	32
Rozdział 2. Pomiary wydajności	33
Wymagania techniczne	34
Przykład pomiaru wydajności	34
Testy porównawcze wydajności	42
Liczniki czasu biblioteki chrono języka C++	42
Liczniki czasu o dużej dokładności	43
Profilowanie wydajności	47
Narzędzie profilujące perf	49
Szczegółowe profilowanie przy użyciu narzędzia perf	51
Narzędzie profilujące pakietu Google Performance	54
Profilowanie z wykorzystaniem grafu wywołań	56
Optymalizacja i wstawianie	58
Profilowanie w praktyce	61
Mikrotesty porównawcze	63
Podstawy mikrotestów porównawczych	63
Mikrotesty porównawcze i optymalizacje kompilatora	66
Google Benchmark	69

Mikrotesty porównawcze to kłamstwo	73
Podsumowanie	76
Pytania	77
Rozdział 3. Architektura procesorów, zasoby i wydajność	78
Wymagania techniczne	79
Wydajność zaczyna się od procesora	79
Badanie wydajności za pomocą mikrotestów porównawczych	81
Wizualizacja obliczeń równoległych na poziomie instrukcji	86
Zależności od danych i potokowanie	89
Potokowanie i rozgałęzienia	93
Przewidywanie rozgałęzień	97
Profilowanie pod kątem nieudanego przewidywania rozgałęzień	99
Wykonywanie spekulatywne	102
Optymalizacja złożonych warunków	103
Wykonywanie obliczeń bez rozgałęzień	108
Odwijanie pętli	108
Operacja wyboru bez użycia rozgałęzień	109
Przykłady wykonywania obliczeń bez rozgałęzień	111
Podsumowanie	115
Pytania	116
Rozdział 4. Architektura i wydajność pamięci	117
Wymagania techniczne	117
Wydajność zaczyna się od procesora, ale na nim się nie kończy	118
Pomiar szybkości dostępu do pamięci	120
Architektura pamięci	121
Pomiar szybkości pamięci głównej i podręcznej	123
Szybkość pamięci — wartości	126
Szybkość operacji losowego dostępu do pamięci	127
Szybkość operacji dostępu sekwencyjnego do pamięci	130
Optymalizacje wydajności pamięci na poziomie sprzętowym	132
Optymalizowanie wydajności pamięci	135
Struktury danych efektywne z perspektywy pamięci	135
Profilowanie wydajności pamięci	139
Optymalizowanie algorytmów pod kątem wydajności pamięci	141
„Duch” w komputerze	146
Czym jest Spectre?	147
Przykład użycia ataku Spectre	149
Atak Spectre w pełni akcji	153
Podsumowanie	157
Pytania	157
Rozdział 5. Wątki, pamięć i współbieżność	158
Wymagania techniczne	158
Wątki i współbieżność	159

Czym jest wątek?	159
Wielowątkowość symetryczna	160
Wątki i pamięć	161
Programy ograniczane przez pamięć i współbieżność	165
Koszt synchronizacji pamięci	166
Dlaczego współużytkowanie danych jest tak kosztowne?	171
Współbieżność i kolejność	178
Potrzeba zapewnienia kolejności	178
Uporządkowanie pamięci i związane z nią bariery	180
Uporządkowanie pamięci w języku C++	186
Model pamięci	188
Podsumowanie	192
Pytania	193

Część II. Zaawansowana współbieżność **195**

Rozdział 6. Wydajność i współbieżność **197**

Wymagania techniczne	198
Co jest niezbędne do efektywnego korzystania ze współbieżności?	198
Blokady, alternatywy i ich wydajność	199
Programy z blokadą, pozbawione blokady oraz bez oczekiwania	202
Różne blokady w przypadku odmiennych problemów	204
Jaka jest faktyczna różnica między programem z blokadą i programem pozbawionym blokady?	208
Tworzenie bloków pod kątem programowania współbieżnego	211
Podstawy współbieżnych struktur danych	212
Liczniki i akumulatory	215
Protokół publikowania	220
Inteligentne wskaźniki używane w programowaniu współbieżnym	222
Podsumowanie	229
Pytania	229

Rozdział 7. Struktury danych odpowiednie w przypadku współbieżności **231**

Wymagania techniczne	232
Czym jest struktura danych bezpieczna wątkowo?	232
Najlepszy rodzaj bezpieczeństwa wątkowego	232
Rzeczywiste bezpieczeństwo wątkowe	234
Stos bezpieczny wątkowo	235
Projektowanie interfejsu pod kątem bezpieczeństwa wątkowego	235
Wydajność struktur danych chronionych przez muteks	238
Wymagania dotyczące wydajności w przypadku różnych zastosowań	239
Szczegółowa analiza wydajności stosu	244
Oszacowania wydajności w przypadku schematów synchronizacji	247
Stos bez blokady	250
Kolejka bezpieczna wątkowo	257

Kolejka pozbawiona blokady	258
Struktury danych spójne niesekwencyjnie	263
Zarządzanie pamięcią na potrzeby współbieżnych struktur danych	266
Lista bezpieczna wątkowo	268
Lista pozbawiona blokady	271
Podsumowanie	277
Pytania	278
Rozdział 8. Obsługa współbieżności w języku C++	279
Wymagania techniczne	279
Obsługa współbieżności w standardzie C++11	280
Obsługa współbieżności w standardzie C++17	281
Obsługa współbieżności w standardzie C++20	286
Podstawy dotyczące współprogramów	286
Składnia współprogramów w języku C++	291
Przykłady współprogramów	292
Podsumowanie	298
Pytania	299
Część III. Projektowanie i pisanie programów o dużej wydajności	301
Rozdział 9. Kod C++ o dużej wydajności	303
Wymagania techniczne	303
Czym jest efektywność języka programowania?	304
Zbędne kopiowanie	305
Kopiowanie i przekazywanie argumentów	305
Kopiowanie jako technika implementacji	307
Kopiowanie w celu przechowywania danych	309
Kopiowanie wartości zwracanych	310
Zastosowanie wskaźników w celu uniknięcia kopiowania	314
Metoda unikania zbędnego kopiowania	315
Nieefektywne zarządzanie pamięcią	316
Zbędne alokacje pamięci	317
Zarządzanie pamięcią w programach współbieżnych	320
Unikanie fragmentacji pamięci	322
Optymalizacja wykonywania warunkowego	325
Podsumowanie	327
Pytania	328
Rozdział 10. Optymalizacje kompilatora w kodzie C++	329
Wymagania techniczne	329
Kompilatory optymalizujące kod	330
Podstawy optymalizacji stosowanych przez kompilator	330
Wstawianie funkcji	333
Co tak naprawdę kompilator „wie”?	338

Przenoszenie informacji z fazy wykonywania do fazy kompilacji	344
Podsumowanie	347
Pytania	348
Rozdział 11. Zachowanie niezdefiniowane i wydajność	349
Wymagania techniczne	350
Czym jest zachowanie niezdefiniowane?	350
Dlaczego występuje zachowanie niezdefiniowane?	353
Zachowanie niezdefiniowane i optymalizacja kodu C++	355
Zastosowanie zachowania niezdefiniowanego do zapewnienia efektywnego projektu	364
Podsumowanie	367
Pytania	369
Rozdział 12. Projektowanie pod kątem wydajności	370
Wymagania techniczne	370
Interakcja między projektem i wydajnością	371
Projektowanie pod kątem wydajności	372
Zasada minimalnej ilości informacji	372
Zasada maksymalnej ilości informacji	374
Kwestie związane z projektowaniem interfejsu API	381
Projektowanie interfejsu API pod kątem współbieżności	381
Kopiowanie i wysyłanie danych	386
Projektowanie pod kątem optymalnego dostępu do danych	389
Kompromisy związane z wydajnością	391
Projekt interfejsu	392
Projektowanie komponentów	393
Błędy i zachowanie niezdefiniowane	394
Podejmowanie przemyślanych decyzji projektowych	395
Podsumowanie	398
Pytania	398
Odpowiedzi	399