

УДК 004.4  
ББК 32.973.202  
В76

**Восс М., Асенхо Р., Рейндерс Дж.**

В76 Параллельное программирование на C++ с помощью библиотеки TBB / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 674 с.: ил.

**ISBN 978-5-97060-864-7**

Эта книга представляет собой современное руководство для всех пишущих на C++ программистов, которые хотят научиться работать с библиотекой Threading Building Blocks (TBB). Написанная экспертами по TBB и параллельному программированию, она вобрала в себя их многолетний коллективный опыт разработки и преподавания параллельного программирования с помощью TBB. Излагаемый материал представлен в доступной форме. В книге имеются многочисленные примеры и рекомендации, которые помогут вам в полной мере овладеть TBB и задействовать всю мощь параллельных систем.

Книга начинается с описания базовых параллельных алгоритмов и средств распараллеливания, имеющихся в стандартной библиотеке шаблонов C++. Вы узнаете об основах управления памятью, работе со структурами данных и решении типичных проблем синхронизации. Затем эти идеи применяются к более сложным системам, на примере которых объясняются компромиссы во имя производительности, общеупотребительные паттерны параллельного программирования, управление потоками и накладные расходы, а также применение TBB к программированию гетерогенных систем и систем на кристалле.

УДК 004.4  
ББК 32.973.202

First published in English under the title «Pro TBB; C++ Parallel Programming with Threading Building Blocks» by Michael Voss, Rafael Asenjo and James Reinders, edition: 1.

This edition has been translated and published under licence from APress Media, LLC, part of Springer Nature.

APress Media, LLC, part of Springer Nature takes no responsibility and shall not be made liable for the accuracy of the translation.

Russian language edition copyright © 2020 by DMK Press. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-1-4842-4397-8 (англ.)  
ISBN 978-5-97060-864-7 (рус.)

Copyright © Intel Corporation, 2019  
© Оформление, издание, перевод,  
ДМК Пресс, 2020

# Содержание

<b>От издательства</b> .....	14
<b>Об авторах</b> .....	15
<b>Благодарности</b> .....	16
<b>Предисловие</b> .....	18
Мыслите параллельно .....	18
Что такое TBB .....	18
Структура книги и предисловия .....	18
Мыслите параллельно .....	19
Мотивы, стоящие за библиотекой TBB.....	19
Программирование с применением задач, а не потоков .....	20
Компонуемость: параллельное программирование необязательно должно быть запутанным.....	21
Масштабируемость, производительность и погоня за переносимой производительностью .....	22
Введение в параллельное программирование .....	23
Параллелизм вокруг нас .....	24
Конкурентность и параллелизм .....	24
Враги параллелизма.....	25
Терминология параллелизма .....	26
Сколько параллелизма в приложении? .....	33
Что такое потоки? .....	38
Что такое SIMD?.....	40
Безопасность в условиях конкурентности .....	41
Взаимное исключение и блокировки .....	41
Корректность .....	43
Взаимоблокировка .....	44
Состояния гонки.....	45
Нестабильность (недетерминированность) результатов .....	45
Уровень абстракции.....	46
Паттерны .....	46
Локальность и месть кешей .....	46
Аппаратное обоснование .....	47
Локальность ссылок .....	48
Строки кеша, выравнивание, разделение, взаимное исключение и ложное разделение .....	49
TBB помнит о кешах.....	53
Введение в векторизацию (SIMD).....	53
Введение в средства C++ (в объеме, необходимом для работы с TBB).....	55
Лямбда-функции.....	55
Обобщенное программирование.....	55
Контейнеры .....	56

Шаблоны .....	56
STL.....	56
Перегрузка .....	57
Диапазоны и итераторы .....	57
Резюме.....	58
Дополнительная информация .....	58

## ЧАСТЬ I

### Глава 1. Приступаем: «Hello, TBB!» ..... 60

Почему именно Threading Building Blocks? .....	60
Производительность: низкие накладные расходы, большое преимущество у C++ ..	61
Эволюция поддержки параллелизма в TBB и C++ .....	62
Недавние добавления в C++, относящиеся к параллелизму .....	63
Библиотека Threading Building Blocks (TBB) .....	63
Интерфейсы параллельного выполнения .....	64
Интерфейсы, не зависящие от модели выполнения .....	66
Использование строительных блоков в TBB .....	66
Да начнем же уже! .....	66
Получение библиотеки TBB.....	66
Получение кода примеров.....	67
Написание первого примера «Hello, TBB!» .....	67
Сборка простых примеров .....	70
Сборка в Windows в Microsoft Visual Studio .....	70
Сборка на платформе Linux из терминала .....	72
Более полный пример .....	74
Начинаем с последовательной реализации .....	75
Добавление уровня обмена сообщениями с помощью потокового графа .....	78
Добавление уровня разветвления–соединения с помощью <code>parallel_for</code> .....	80
Добавление уровня SIMD с помощью функции <code>transform</code> из Parallel STL .....	81
Резюме.....	84

### Глава 2. Обобщенные параллельные алгоритмы ..... 85

Функциональный параллелизм на уровне задач .....	88
Чуть более сложный пример: параллельная реализация быстрой сортировки .....	90
Циклы: <code>parallel_for</code> , <code>parallel_reduce</code> и <code>parallel_scan</code> .....	92
<code>parallel_for</code> : применение тела к каждому элементу диапазона .....	92
<code>parallel_reduce</code> : вычисление одного результата для всего диапазона .....	95
<code>parallel_scan</code> : редукция с промежуточными значениями .....	100
Как это работает? .....	102
Более сложный пример: линия прямой видимости .....	103
Варить до готовности: <code>parallel_do</code> и <code>parallel_pipeline</code> .....	105
<code>parallel_do</code> : применять тело, пока имеются элементы .....	106
<code>parallel_pipeline</code> : обработка несколькими фильтрами .....	113
Резюме.....	120
Дополнительная информация .....	120

### Глава 3. Потоковые графы .....122

Зачем использовать графы для выражения параллелизма? .....	123
Основы интерфейса потоковых графов в TBB.....	124
Шаг 1: создать объект графа .....	125

Шаг 2: создать узлы .....	126
Шаг 3: добавить ребра .....	128
Шаг 4: запустить граф .....	128
Шаг 5: ждать завершения выполнения графа .....	131
Более сложный пример потокового графа данных .....	131
Реализация примера в виде потокового графа TBV .....	133
Производительность потокового графа данных .....	134
Частный случай – графы зависимостей .....	136
Реализация графа зависимостей .....	138
Оценка масштабируемости графа зависимостей .....	143
Дополнительные сведения о потоковых графах в TBV .....	143
Резюме .....	144

## Глава 4. TBV и параллельные алгоритмы стандартной библиотеки шаблонов C++ .....

Какое отношение библиотека STL имеет к этой книге? .....	145
Аналогия для осмысления политик выполнения в Parallel STL .....	147
Простой пример – алгоритм <code>std::for_each</code> .....	148
Какие алгоритмы предоставляет реализация Parallel STL? .....	151
Как получить и использовать копию библиотеки STL, в которой применяется TBV .....	151
Алгоритмы в библиотеке Intel Parallel STL .....	152
Нестандартные итераторы открывают дополнительные способы использования .....	153
Некоторые наиболее полезные алгоритмы .....	156
<code>std::for_each</code> , <code>std::for_each_n</code> .....	156
<code>std::transform</code> .....	158
<code>std::reduce</code> .....	159
<code>std::transform_reduce</code> .....	160
Политики выполнения в деталях .....	162
<code>sequenced_policy</code> .....	162
<code>parallel_policy</code> .....	163
<code>unsequenced_policy</code> .....	163
<code>parallel_unsequenced_policy</code> .....	164
Какую политику выполнения использовать? .....	164
Другие способы ввести SIMD-параллелизм .....	165
Резюме .....	166
Дополнительная информация .....	166

## Глава 5. Синхронизация – почему ее нужно избегать и как это сделать .....

Сквозной пример: гистограмма изображения .....	167
Небезопасная параллельная реализация .....	170
Первая безопасная параллельная реализация: крупнозернистая блокировка .....	173
Варианты мьютексов .....	178
Вторая безопасная параллельная реализация: мелкозернистая блокировка .....	180
Третья потокобезопасная параллельная реализация: атомарные переменные .....	184
Улучшенная параллельная реализация: приватизация и редукция .....	188
Поточно-локальная память .....	189
Класс <code>enumerable_thread_specific</code> .....	190
Тип <code>combinable</code> .....	192

Самая простая параллельная реализация: шаблон редукции.....	194
Подведем итоги .....	196
Резюме.....	200
Дополнительная информация .....	200

## Глава 6. Структуры данных для конкурентного

<b>программирования</b> .....	201
Основы важнейших структур данных .....	202
Неупорядоченные ассоциативные контейнеры .....	202
Отображение или множество .....	203
Несколько значений.....	203
Хеширование .....	203
Неупорядоченность.....	204
Конкурентные контейнеры.....	204
Конкурентные неупорядоченные ассоциативные контейнеры .....	206
Конкурентные очереди: обычные, ограниченные и с приоритетами .....	212
Конкурентный вектор.....	220
Резюме.....	223

## Глава 7. Масштабируемое выделение памяти .....

Выделение памяти в современном C++ .....	224
Масштабируемое выделение памяти: что .....	225
Масштабируемое выделение памяти: почему .....	226
Избежание ложного разделения с помощью дополнения .....	227
Альтернативы масштабируемому выделению памяти: какие.....	229
К вопросу о компиляции.....	230
Самый популярный способ использования (библиотека прокси для C/C++): как.....	230
Linux: использование библиотеки прокси .....	231
macOS: использование библиотеки прокси .....	232
Windows: использование библиотеки прокси.....	232
Тестирование библиотеки прокси .....	233
Функции C: масштабируемые распределители памяти для C .....	234
Классы C++: масштабируемые распределители памяти для C++ .....	235
Распределители с сигнатурой <code>std::allocator&lt;T&gt;</code> .....	236
<code>scalable_allocator</code> .....	236
<code>tbb_allocator</code> .....	237
<code>zero_allocator</code> .....	237
<code>cached_aligned_allocator</code> .....	237
Поддержка пула памяти: <code>memory_pool_allocator</code> .....	238
Поддержка выделения памяти для массивов: <code>aligned_space</code> .....	238
Избирательная подмена <code>new</code> и <code>delete</code> .....	239
Настройка производительности: некоторые рычаги управления .....	242
Что такое большие страницы? .....	242
Поддержка больших страниц в TBB .....	242
<code>scalable_allocation_mode(int mode, intptr_t value)</code> .....	243
<code>TBBMALLOC_USE_HUGE_PAGES</code> .....	243
<code>TBBMALLOC_SET_SOFT_HEAP_LIMIT</code> .....	243
<code>int scalable_allocation_command(int cmd, void *param)</code> .....	244
<code>TBBMALLOC_CLEAN_ALL_BUFFERS</code> .....	244
<code>TBBMALLOC_CLEAN_THREAD_BUFFERS</code> .....	244
Резюме.....	244

<b>Глава 8. TBB и параллельные паттерны</b>	245
Параллельные паттерны и параллельные алгоритмы	245
Паттерны определяют классификацию алгоритмов, проектных решений и т. д.	247
Паттерны, которые работают	248
Параллелизм данных одерживает победу	249
Паттерн Вложенность	249
Паттерн Отображение	251
Паттерн Куча работ	252
Паттерны редукции (Редукция и Сканирование)	252
Паттерн Разветвление–соединение	253
Паттерн Разделяй и властвуй	256
Паттерн Ветви и границы	256
Паттерн Конвейер	257
Паттерн Событийно-управляемая координация (реактивные потоки)	258
Резюме	259
Дополнительная информация	259

## ЧАСТЬ II

<b>Глава 9. Столпы компонуемости</b>	261
Что такое компонуемость?	262
Вложенная композиция	263
Конкурентная композиция	265
Последовательная композиция	266
Благодаря каким особенностям библиотека TBB является компонуемой	268
Пул потоков TBB (рынок) и аренды задач	268
Диспетчер задач в TBB: заимствование работ, и не только	271
Соберем все вместе	277
Забегая вперед	281
Управление количеством потоков	281
Изоляция работ	281
Привязка задачи к потоку и потока к ядру	281
Приоритеты задач	281
Резюме	282
Дополнительная информация	282

<b>Глава 10. Использование задач для создания собственных алгоритмов</b>	283
Сквозной пример: вычисление последовательности	283
Высокоуровневый подход: <code>parallel_invoke</code>	285
Высший среди низших: <code>task_group</code>	287
Низкоуровневый интерфейс: часть первая – блокировка задач	289
Низкоуровневый интерфейс задач: часть вторая – продолжение задачи	293
Обход планировщика	299
Низкоуровневый интерфейс задач: часть третья – рециклинг задач	300
Контрольный список для интерфейса задач	302
И еще одно: FIFO-задачи (типа запустил и забыл)	303
Применение низкоуровневых средств на практике	304
Резюме	310
Дополнительная информация	311

<b>Глава 11. Управление количеством потоков</b>	312
Краткий обзор архитектуры планировщика TBB	313
Интерфейсы для управления количеством задач	314
Управление количеством потоков с помощью <code>task_scheduler_init</code>	314
Управление количеством потоков с помощью <code>task_arena</code>	315
Управление количеством потоков с помощью <code>global_control</code>	316
Сводка концепций и классов	316
Рекомендации по заданию количества потоков	317
Использование одного объекта <code>task_scheduler_init</code> в простом приложении	318
Использование нескольких объектов <code>task_scheduler_init</code> в простом приложении	320
Использование нескольких арен с разным числом слотов, чтобы подсказать TBB, куда направлять рабочие потоки	321
Использование <code>global_control</code> для управления количеством потоков, доступных для занятия слотов на аренах	324
Использование <code>global_control</code> с целью временно ограничить количество доступных потоков	326
Когда НЕ следует управлять количеством потоков	328
Что не так?	329
Резюме	330
 <b>Глава 12. Применение изоляции работы для обеспечения корректности и повышения производительности</b>	331
Изоляция работ для обеспечения корректности	332
Создание изолированного региона с помощью <code>this_task_arena::isolate</code>	336
Использование арен задач для изоляции: обоюдоострый меч	341
Не поддавайтесь искушению использовать арены задач для изоляции ради корректности	344
Резюме	347
Дополнительная литература	347
 <b>Глава 13. Привязка потока к ядру и задачи к потоку</b>	348
Создание привязки потока к ядру	349
Создание привязки задачи к потоку	351
Когда и как следует использовать средства привязки в TBB?	357
Резюме	358
Дополнительная информация	358
 <b>Глава 14. Приоритеты задач</b>	359
Поддержка невытесняющих приоритетов в классе задач TBB	359
Задание статических и динамических приоритетов	361
Два простых примера	362
Реализация приоритетов без поддержки со стороны задач TBB	365
Резюме	367
Дополнительная информация	368
 <b>Глава 15. Отмена и обработка исключений</b>	369
Как отменить коллективную работу	370
Отмена задач в деталях	371
Явное назначение TGS	373

Назначение TGC по умолчанию .....	375
Обработка исключений в TBB .....	379
Написание собственных классов исключений TBB .....	381
Соберем все вместе: компоновка, отмена и обработка исключений .....	384
Резюме .....	386
Дополнительная информация .....	387

## Глава 16. Настройка TBB-алгоритмов: зернистость,

## локальность, параллелизм и детерминированность .....

Зернистость задач: какой размер достаточен? .....	389
Выбор диапазонов и разбивателей для циклов .....	390
Обзор разбивателей .....	391
Выбирать ли степень детализации для управления зернистостью задач .....	392
Диапазоны, разбиватели и производительность кеша данных .....	395
Использование <code>static_partitioner</code> .....	402
Ограничение планировщика ради детерминированности .....	404
Настройка конвейеров в TBB: количество фильтров, режимы и маркеры .....	406
Сбалансированный конвейер .....	407
Несбалансированный конвейер .....	409
Конвейеры, локальность данных и привязка к потоку .....	410
В глубоких водах .....	411
Создание собственного типа диапазона .....	411
Класс <code>Pipeline</code> и фильтры, привязанные к потоку .....	414
Резюме .....	418
Дополнительная информация .....	418

## Глава 17. Потокосовые графы: дополнительные сведения .....

Оптимизация зернистости, локальности и степени параллелизма .....	419
Зернистость узла: какой будет достаточно? .....	420
Потребление памяти и локальность данных .....	428
Арены задач и потоковый граф .....	441
Рекомендации по работе с потоковыми графами: что полезно, а что вредно .....	444
Полезно: использовать вложенный параллелизм .....	444
Вредно: использовать многофункциональные узлы вместо вложенного параллелизма .....	444
Полезно: использовать узлы <code>join_node</code> , <code>sequencer_node</code> или <code>multifunction_node</code> для восстановления порядка в потоковом графе, когда это необходимо .....	445
Полезно: использовать функцию <code>isolate</code> для вложенного параллелизма .....	448
Полезно: использовать отмену и обработку исключений в потоковых графах .....	450
Полезно: задавать приоритеты для графа, в котором используется <code>task_group_context</code> .....	454
Вредно: создавать ребро между узлами разных графов .....	454
Полезно: использовать <code>try_put</code> для передачи информации между графами .....	456
Полезно: использовать <code>composite_node</code> для инкапсуляции группы узлов .....	458
Введение в Intel Advisor: Flow Graph Analyzer .....	462
Процесс проектирования в FGA .....	462
Процесс анализа в FGA .....	465
Диагностика проблем производительности с помощью FGA .....	467
Резюме .....	470
Дополнительная информация .....	470



## Глава 18. Дополнение потоковых графов асинхронными узлами

<b>узлами</b> .....	471
Пример из асинхронного мира .....	472
Зачем и когда использовать <code>async_node</code> ? .....	476
Более реалистичный пример .....	478
Резюме .....	486
Дополнительная информация .....	487

## Глава 19. Накачаные потоковые графы: узлы OpenCL

Пример «Hello OpenCL_Node» .....	489
Где исполняется наше ядро? .....	496
Возвращаясь к более реалистичному примеру из главы 18 .....	502
Дьявол кроется в деталях .....	509
Концепция NDRange .....	511
Поиграем со смещением .....	515
Задание ядра OpenCL .....	516
Еще о выборе устройства .....	517
Предупреждение по поводу порядка .....	520
Резюме .....	523
Дополнительная информация .....	524

## Глава 20. TBV в системах с архитектурой NUMA

Определение топологии платформы .....	527
Каковы затраты на доступ к памяти .....	530
Базовый пример .....	531
Мастерство размещения данных и привязки к процессору .....	533
Привлекаем <code>hwloc</code> и TBV к совместной работе .....	538
Более сложные альтернативы .....	543
Резюме .....	544
Дополнительная информация .....	545

## Приложение А. История и предшественники

Десятилетие «от птенца к орлу» .....	546
1. Революция TBV внутри Intel .....	546
2. Первая революция TBV в сфере параллелизма .....	547
3. Вторая революция TBV в сфере параллелизма .....	548
4. Птички TBV .....	549
Источники идей TBV .....	551
Модель ослабленного последовательного выполнения .....	552
Библиотеки, оказавшие влияние .....	552
Языки, оказавшие влияние .....	554
Прагмы, оказавшие влияние .....	554
Влияние обобщенного программирования .....	555
Учет кешей .....	555
Учет стоимости квантования времени .....	556
Литература для дополнительного чтения .....	557

<b>Приложение В. TBB в кратком изложении .....</b>	<b>560</b>
Отладка и условный код.....	560
Макросы ознакомительных средств .....	562
Диапазоны.....	562
Разбиватели .....	563
Алгоритмы .....	564
Алгоритм: parallel_do.....	564
Алгоритм: parallel_for .....	567
Алгоритм: parallel_for_each.....	569
Алгоритм: parallel_invoke .....	571
Алгоритм: parallel_pipeline.....	572
Алгоритм: parallel_reduce и parallel_deterministic_reduce .....	574
Алгоритм: parallel_scan.....	578
Алгоритм: parallel_sort.....	581
Алгоритм: pipeline .....	583
Потоковый граф.....	585
Потоковый граф: класс graph .....	586
Потоковый граф: порты и ребра .....	587
Потоковый граф: узлы .....	587
Выделение памяти.....	597
Контейнеры.....	602
Синхронизация .....	620
Поточно-локальная память (TLS) .....	626
Хронометраж .....	634
Группы задач: использование планировщика с заимствованием задач.....	635
Планировщик задач: точный контроль над планировщиком с заимствованием задач .....	636
Настройки плавающей точки .....	647
Исключения .....	649
Потоки .....	651
Parallel STL .....	652
<b>Глоссарий.....</b>	<b>655</b>
<b>Предметный указатель .....</b>	<b>668</b>