Вопросы по курсу "Параллельное программирование"

1 Лекция 1: introduction to multithreading

Ключевые понятия: concurrency, parallelism, agents, threads, scheduler, Amdahl's law, race condition, deadlock, wait-for graph

1.1 Конкурентность (concurrency) и параллелизм (parallelism). Системы разделения времени (time-sharing), многозадачность (multitasking). Процессы и потоки. Вытесняющая многозадачность и кооперативная многозадачность. Планировщик ОС: квант планирования, смена контекста, метрики эффективности планирования, стратегии планирования, алгоритм round-robin. Задачи, легко поддающиеся параллелизации (embarrassingly parallel problems). Закон Амдала.

1.2 Недетерминизм многопоточного исполнения, состояние гонки (race condition), гонка данных (data race). Word tearing. Видимость (visibility). Wait-for graph. Состояние тупика (deadlock). Инверсия приоритетов.

2 Лекция 2: basic concepts

Ключевые понятия: mutex, acquisition order, reentrancy, fairness, data locking, code locking, signalling, condition variable, lost signal, spurious wakeup

2.1 Неформальный подход к потоковой безопасности (thread-safety), примеры. Ключевые свойства многопоточного алгоритма: безопасность/корректность (safety), живность (liveness), производительность.

Взаимное исключение. Критическая секция (mutex). Admission policy. Реентерабельность (reentrancy), честность (fairness), голодание (starvation).

Блокировка кода (code locking), блокировка данных (data locking). Расщепление мьютексов (lock splitting). Упорядочивание критических секций для предотвращения состояния тупика (lock ordering, deadlock prevention).

Проблема lock convoy. Задача об обедающих философах (dining philosophers problem).

2.2 Неформальный подход к потоковой безопасности (thread-safety), примеры. Ключевые свойства многопоточного алгоритма: безопасность/корректность (safety), живность (liveness), производительность.

Передача сигнала от одного потока другому (signalling). Условная переменная (condition variable).

Пример API: `wait/notify/notifyAll`. Пример API: `await/signal/signalAll`. Проблема потерянного сигнала (lost signal), устаревания условия (predicate invalidation), внезапного пробуждения (spurious wakeup). Честность (fairness). Голодание (starvation).

3 Лекция 3: advanced synchronization primitives

Ключевые понятия: monitor, latch, barrier, thundering herd, semaphore, read-write lock, thread pool, executor, producer-consumer, fork-join, load balancing

3.1 Взаимное исключение. Критическая секция (mutex). Admission policy. Реентерабельность (reentrancy), честность (fairness), голодание (starvation).Передача сигнала от одного потока другому (signalling). Условная переменная (condition variable).

Мониторы в языке Java (built-in monitor). Описание API. Ключевое слово `synchronized`. Admission policy. Реентерабельность (reentrancy), честность (fairness), владение (ownership).

Блокировка кода (code locking), блокировка данных (data locking). Расщепление мьютексов (lock splitting). Упорядочивание критических секций для предотвращения состояния тупика (lock ordering, deadlock prevention).

Проблема потерянного сигнала (lost signal), устаревания условия (predicate invalidation), внезапного пробуждения (spurious wakeup), грохочущего стада (thundering herd), инверсии приоритета (priority inversion), голодания (starvation), конвоя (lock convoy).

3.2 Взаимное исключение. Критическая секция (mutex). Admission policy. Реентерабельность (reentrancy), честность (fairness), голодание (starvation).

Передача сигнала от одного потока другому (signalling). Условная переменная (condition variable).

CountDownLatch.

Проблема потерянного сигнала (lost signal), устаревания условия (predicate invalidation), внезапного пробуждения (spurious wakeup), грохочущего стада (thundering herd), инверсии приоритета (priority inversion), голодания (starvation), конвоя (lock convoy).

3.3 Взаимное исключение. Критическая секция (mutex). Admission policy. Реентерабельность (reentrancy), честность (fairness), голодание (starvation).

Передача сигнала от одного потока другому (signalling). Условная переменная (condition variable).

CyclicBarrier.

Проблема потерянного сигнала (lost signal), устаревания условия (predicate invalidation), внезапного пробуждения (spurious wakeup), грохочущего стада (thundering herd), инверсии приоритета (priority inversion), голодания (starvation), конвоя (lock convoy).

3.4 Взаимное исключение. Критическая секция (mutex). Admission policy. Реентерабельность (reentrancy), честность (fairness), голодание (starvation).

Передача сигнала от одного потока другому (signalling). Условная переменная (condition variable).

Semaphore.

Проблема потерянного сигнала (lost signal), устаревания условия (predicate invalidation), внезапного пробуждения (spurious wakeup), грохочущего стада (thundering herd), инверсии приоритета (priority inversion), голодания (starvation), конвоя (lock convoy).

3.5 Взаимное исключение. Критическая секция (mutex). Admission policy. Реентерабельность (reentrancy), честность (fairness), голодание (starvation).

Передача сигнала от одного потока другому (signalling). Условная переменная (condition variable).

ReadWriteLock. Предпочтение читающего или пишущего потока. Усиление (read-to-write promotion) и ослабление (write-to-read downgrade).

Проблема потерянного сигнала (lost signal), устаревания условия (predicate invalidation), внезапного пробуждения (spurious wakeup), грохочущего стада (thundering herd), инверсии приоритета (priority inversion), голодания (starvation), конвоя (lock convoy).

3.6 Взаимное исключение. Критическая секция (mutex). Admission policy. Реентерабельность (reentrancy), честность (fairness), голодание (starvation).

Передача сигнала от одного потока другому (signalling). Условная переменная (condition variable).

Thread pools: `ThreadFactory`, `Executor`, `Future`, `ExecutorService`, `fixedThreadPool`, `singleThreadExecutor`, `cachedThreadPool`.

Проблемы, характерные для пулов потоков: взаимные блокировки, утечки памяти, голодание, искажение владения (ownership, thread-local).

3.7 Взаимное исключение. Критическая секция (mutex). Admission policy. Реентерабельность (reentrancy), честность (fairness), голодание (starvation).

Передача сигнала от одного потока другому (signalling). Условная переменная (condition variable).

Шаблон "производитель-потребитель" (producer-consumer). Балансировка нагрузки (load balancing). Проблемы балансировки нагрузки для push-based протоколов. Шаблоны разделения работы: work arbitrage, work dealing, work stealing. Fork-join модель, её преимущества и недостатки.

4 Лекция 4: advanced synchronization concepts

Ключевые понятия: interruption, cancellation, partitioning, privatization, replication, thread-local, ownership

4.1 Отмена задач (cancellation, interruption). Прерывание задач (task cancellation) и прерывание потоков (thread cancellation). Односторонняя отмена задач (forced cancellation). Кооперативная отмена задач с помощью cancellation token. Кооперативная отмена задач с помощью cancellation points. Прерываемые (interruptible) и непрерываемые (uninterruptible) методы. Политики распространения отмены задач. Упорядочение асинхронных событий: отмена, сигнал условной переменной, срабатывания таймера (interruption, notification, timeout ordering).

4.2 Техники декомпозиции многопоточных программ. Конечный автомат. Перенос функций (function shipping). Выделенный поток (designated thread). Разбиение потоков на группы. Разбиение данных на кластеры. Приватизация данных (privatization). Репликация (replication). Группировка однотипных событий (batching). Владение (ownership) и привязка к потоку исполнения (thread confinement).

5 Лекция 5: additional topics of practical concurrency

Ключевые понятия: documenting protocols and classes, checking concurrent invariants, stress testing, execution trace analysis, estimating required testing effort, static and dynamic checks, scheduling randomization, model checking

5.1 Документирование многопоточных инвариантов. Проверка инвариантов потоковой безопасности для различных структур данных. Assertions. Препятствия для написания надежных многопоточных программ.

5.2 Тестирование многопоточных программ. Юнит-тестирование, стресс-тестирование. Построение и анализ трасс исполнения. Проектирование тестов, исполняемых в многопоточной среде. Анализ вероятности негативного сценария, оценка требуемых вычислительных ресурсов для тестирования.

5.3 Инструменты анализа многопоточных программ. Статический анализ и поиск дефектов. Проверки времени исполнения. Мониторинг и observability API. Контроль порядка исполнения конкурентной программы (scheduling control): с помощью тестов (mocking, inheritance), с помощью языковых средств (aspects, bytecode transformation), с помощью операционной системы (chaos mode execution).

6 Лекция 6: theoretical foundations of concurrency

Ключевые понятия: timeline, events, precedence, 2-thread mutual exclusion, deadlock freedom, starvation freedom, N-thread mutual exclusion, sequential objects and specifications, concurrent objects, linearizability

6.1 Формализация многопоточного исполнения: линия времени, атомарное событие (event), трасса исполнения потока (trace model), временной интервал. Частичный порядок, отношение предшествования (precedence), его свойства.

Формальное определение взаимного исключения (mutual exclusion), отсутствия тупика (deadlock freedom), отсутствия голодания (starvation freedom).

Алгоритмы **LockOne**, **LockTwo**, алгоритм Петерсона, доказательства их основных свойств.

Теорема о нижней границе: сколько необходимо независимых ячеек памяти, чтобы добиться взаимного исключения N потоков.

6.2 Формализация многопоточного исполнения: линия времени, атомарное событие (event), трасса исполнения потока (trace model), временной интервал. Частичный порядок, отношение предшествования (precedence), его свойства.

Формальное определение взаимного исключения (mutual exclusion), отсутствия тупика (deadlock freedom), отсутствия голодания (starvation freedom).

Алгоритм **FilterLock**, доказательства его основных свойств.

Теорема о нижней границе: сколько необходимо независимых ячеек памяти, чтобы добиться взаимного исключения N потоков.

6.3 Формализация многопоточного исполнения: линия времени, атомарное событие (event), трасса исполнения потока (trace model), временной интервал. Частичный порядок, отношение предшествования (precedence), его свойства.

Последовательный объект (sequential object), последовательная спецификация (sequential specification). Понятие о многопоточной согласованности (concurrent consistency). Линеаризуемое исполнение (linearizable execution), линеаризуемый объект (linearizable object), точка линеаризации (linearization point). Примеры.

7 Лекция 7: progress guarantees, concurrent operations hierarchy, consensus number

Ключевые понятия: obstruction-free, lock-free, wait-free, safe register, regular register, atomic register, register snapshot, consensus number

7.1 Условия прогресса (progress conditions). Dependent blocking progress conditions. Примеры. Non-blocking progress conditions. Примеры. Dependent non-blocking progress conditions. Пример. Теорема о связи условий прогресса между собой.

Регистр (register): емкость (capacity), поддерживаемая конкурентность (concurrency), гарантии консистентности (consistency). Пространство регистров. Wait-free реализация объекта.

Теоремы о выразительной силе:

- SRSW Safe Boolean -> MRSW Safe Boolean
- MRSW Safe Boolean -> MRSW Regular Boolean
- MRSW Regular Boolean -> MRSW Regular

7.2 Условия прогресса (progress conditions). Dependent blocking progress conditions. Примеры. Non-blocking progress conditions. Примеры. Dependent non-blocking progress conditions. Пример. Теорема о связи условий прогресса между собой.

Регистр (register): емкость (capacity), поддерживаемая конкурентность (concurrency), гарантии консистентности (consistency). Пространство регистров. Wait-free реализация объекта.

Теоремы о выразительной силе:

- MRSW Regular -> SRSW Atomic
- SRSW Atomic -> MRSW Atomic

7.3 Условия прогресса (progress conditions). Dependent blocking progress conditions. Примеры. Non-blocking progress conditions. Примеры. Dependent non-blocking progress conditions. Пример. Теорема о связи условий прогресса между собой.

Регистр (register): емкость (capacity), поддерживаемая конкурентность (concurrency), гарантии консистентности (consistency). Пространство регистров. Wait-free реализация объекта.

Атомарный снимок состояния N регистров (atomic snapshot). Алгоритм **CleanCollect** и его недостатки, пример ABA проблемы. Алгоритм **SimpleSnapshot**, обоснование корректности и его основных свойств.

8 Лекция 8: introduction to atomics

Ключевые понятия: read-modify-write, get-and-add, compare-and-swap, spin lock, lock-free stack, ABA problem

8.1 Условия прогресса (progress conditions). Dependent blocking progress conditions. Примеры. Non-blocking progress conditions. Примеры. Dependent non-blocking progress conditions. Пример. Теорема о связи условий прогресса между собой.

Консенсус. Теорема о невозможности достичь N-потокового консенсуса с использованием только атомарных регистров. Практические следствия. Consensus number.

Read-modify-write objects. Примеры.

Теорема: `compareAndSet` обладает ∞ consensus number. Доказательство и практические следствия.

8.2 Взаимное исключения с помощью циклов и read-modify-write операций. Test-and-set-lock. Test-and-test-and-set-lock. Exponential backoff.

Проблемы масштабирования многопоточных систем, вызванные работой алгоритмов когерентности кешей. Вычислительные системы, использующие шину данных (bus-based architectures). Репликация линий кеша (cache line replication). Когерентность кешей (cache coherence). Проблема массовой инвалидации данных (invalidation storm).

8.3 Условия прогресса (progress conditions). Dependent blocking progress conditions. Примеры. Non-blocking progress conditions. Примеры. Dependent non-blocking progress conditions. Пример. Теорема о связи условий прогресса между собой.

Lock-free реализация потокобезопасного LIFO контейнера. Переиспользование памяти и ABA проблема.

Проблемы масштабирования многопоточных систем, вызванные работой алгоритмов когерентности кешей. Вычислительные системы, использующие шину данных (bus-based architectures). Репликация линий кеша (cache line replication). Когерентность кешей (cache coherence). Проблема массовой инвалидации данных (invalidation storm).

9 Лекция 9: cache coherency

Ключевые понятия: cache memory hierarchy, cache coherency protocol, store-buffer, load-buffer, invalidate-queue, memory barrier, hardware memory model, weak memory model, litmus tests

9.1 Наивные способы синхронизации глобальной памяти. Кеш-линия (cache line). Истинное и ложное разделение данных (true sharing, false sharing). Когерентность (coherence). Ослабленные требования к консистентности независимых ячеек памяти.

Построение эффективной подсистемы памяти: кеширование, шаблон "репликация", протокол обмена сообщениями. MESI протокол и примеры его работы. Преимущества и недостатки.

9.2 Наивные способы синхронизации глобальной памяти. Кеш-линия (cache line). Истинное и ложное разделение данных (true sharing, false sharing). Когерентность (coherence). Ослабленные требования к консистентности независимых ячеек памяти.

MESI протокол: состояния и их семантика, виды сообщений и их классификация (синхронные/асинхронные). Буферизация записи (store buffering), требования корректности (store forwarding).

9.3 Наивные способы синхронизации глобальной памяти. Кеш-линия (cache line). Истинное и ложное разделение данных (true sharing, false sharing). Когерентность (coherence). Ослабленные требования к консистентности независимых ячеек памяти.

MESI протокол: состояния и их семантика, виды сообщений и их классификация (синхронные/асинхронные). Буферизация чтений (load buffering). Топология шины данных (interconnect topology).

9.4 Наивные способы синхронизации глобальной памяти. Кеш-линия (cache line). Истинное и ложное разделение данных (true sharing, false sharing). Когерентность (coherence). Ослабленные требования к консистентности независимых ячеек памяти.

Модель памяти процессора (hardware memory model). Слабая модель памяти (weak memory model). Барьеры памяти (memory barriers), их классификация. Преимущества и недостатки барьерного подхода к написанию корректных конкурентных программ.

10 Лекция 10: language memory model

Ключевые понятия: compiler optimizations, compiler barriers, language memory model, strict consistency, threads cannot be implemented as a library, visibility, volatile

10.1 Оптимизации компилятора: влияние на наблюдаемое поведение однопоточной и многопоточной программы. Видимость операций с памятью (visibility). Барьеры для оптимизаций (compiler barriers), их таксономия. Необходимость использовать специальные языковые конструкции для "борьбы" с оптимизациями.

10.2 Модель памяти языка программирования (language memory model). Мотивация, цели, сложности при написании.

Неизменяемость (immutability). Декларативный подход к описанию параллельных вычислений. Строгая консистентность (strict consistency) с помощью взаимного исключения. Строгая консистентность (strict consistency) с помощью однопоточного исполнения. Реализация потоков на уровне библиотеки языка: преимущества и недостатки. Формальная модель памяти языка программирования. Альтернативы.

10.3 Модель памяти языка программирования (language memory model). Мотивация, цели, сложности при написании.

Видимость операций с памятью (visibility). Load-acquire/store-release подход к описанию консистентности. Ключевое слово `volatile`. Рекомендации к использованию.