Программа

1. Введение

1. Тенденции развития вычислительных систем, обуславливающие необходимость применения распределённых (параллельных) методов вычислений. Примеры вычислительно ёмких задач из разных областей науки.
2. Классификация параллельных систем (SIMD, MISD…, SMP, MPP)
3. Современные высокопроизводительные системы: начиная от расширений SSE, через многоядерность к узлам кластеров
4. Понятия ускорения, эффективности (закон Амдала)
5. Многопоточность или IPC
6. План курса

2. Создание/завершение потоков

1. Механизм запуска потока
2. Корректное завершение потоков:
    - cancellation points
    - interrupted exception
    - примеры кода в glibc
3. Сравнение различных потоков (POSIX, boost, java)
4. Проброс исключений между потоками

3. Примитивы синхронизации

1. Необходимость синхронизации: простые гонки данных
2. Реализация примитивов синхронизации: алгоритм булочника
3. Виды мьютексов:
    - рекурсивные/нерекурсивные
    - read/write
    - spin
    - futex
4. Корректные захват/освобождение примитивов
5. CAS-операции и атомики
6. Условные переменные:
    - использование wait/notify
    - Spurious wakeups
7. Thread Local Storage (TLS)

4. Алгоритмы синхронизации

1. Грубая
2. Тонкая

3. Оптимистичная
4. Ленивая
5. Неблокирующая *(параллель с ORM)*

5. Атомарные снимки регистров

1. Классификация алгоритмов:
    - lock-free
    - wait-free
2. Lock-free snapshot
3. Wait-free snapshot

6. Ошибки || программирования

1. Основные ошибки многопоточного программирования
    - Гонки данных (Data Race)
    - Взаимная блокировка (Deadlock)
2. Специфические ошибки
    - Реакция потока на сигнал
    - Блокировки при fork многопоточных программ
    - Проблема *ABA*
    - Инверсия приоритетов

7. Модель памяти

1. Устройство кэшей процессора
2. Пример на протоколе MESI
3. Барьеры памяти (store/load)
4. Модели памяти: Sequential consistency…
5. Acquire/release семантика

8. Профилирование многопоточных приложений

1. Средства анализа производительности
    - Утилита time
    - Intel Parallel Studio
    - Valgrind (модули callgrind, cachegrind)
2. Пример поиска узких мест
3. Профилирование промашек по кэшу и метрика CPI

9. Flat-Combining

1. Схема Flat-Combining
2. Возможные оптимизации за счёт интерференции операций
3. Сравнение производительности с lock-free очередью Michael & Scott

10. RCU

1. Суть RCU и синхронизация на эпохах
2. Kernel-space RCU
3. User-space RCU

11. Транзакционная память

1. Идея transactional memory
    - Software transactional memory
    - Hardware transactional memory
2. Преимущества и круг задач
3. Реализация HTM на линейках кэша
4. Lock teleportation

12. Сети Петри

1. Суть модели сетей Петри
2. Пример с обедающими философами
3. Верификация || программ

13. Консенсус

1. Консенсус:
    - Консенсусное число RMW-регистров
    - Универсальность CAS-операций

14. Асинхронный ввод/вывод

1. Блокирующий/неблокирующий
2. Синхронный (реактор)/асинхронный (проактор)
3. Преимущества асинхронной работы и реализация со стороны операционной системы
4. Библиотеки асинхронного ввода/вывода

15. Линеаризуемость

1. Понятие линеаризуемости
2. Lock-free стек Trieber
3. Пример на очередях
4. Lock-free очередь Michael & Scott
5. Точки линеаризации

16. Оптимизации в компиляторах

1. Статические оптимизации
2. Оптимизации циклов:
    - Развёртывание
    - Повторение
    - Вынесение инварианта
3. JIT-оптимизации
    - Объединение захвата примитивов
    - Оптимистичный захват
    - Адаптивные блокировки
    - Замена виртуального вызова

17. Шаблоны || программирования

1. Структурные шаблоны:
    - Декомпозиция по задачам
    - Геометрическая декомпозиция
    - Recursive Data
    - Pipeline
2. Некоторые программные структуры:
    - Parallel loops
    - Boss/Worker
3. Разное:
    - Double check
    - Local Serializer

18. OpenMP

1. Архитектура работы через директивы препроцессора
2. Параллельные секции
3. Области видимости переменных
4. Ограничения
5. Миграция вычислений

19. Intel TBB

1. Алгоритмы
2. Аллокаторы
3. Деревья задач
4. Особенности планирования (work stealing…)
5. flow graphs *(параллель с BPEL)*

20. Кластерные вычисления (MPI)

1. Виды кластерных систем:
    - Балансировки нагрузки
    - Высокой надёжности
    - Вычислительные
2. История и назначение стандарта MPI
3. Обмен сообщениями:
    - С блокировкой
    - Без блокировки
    - Отложенные запросы на взаимодействие
4. Взаимодействие процессов:
    - Группы и коммуникаторы
    - Операции коллективного взаимодействия процессов
    - Редукция
    - Виртуальные топологии
5. Средства анализа производительности:
    - Jumpshot
    - Intel® Trace Analyzer и Intel® Trace Collector

21. Сопрограммы / Coroutines

1. Преимущества по отношению к callback-программированию
2. Примеры co_await и сравнение с синхронным кодом
3. Проблемы реализации примитивов и TLS
4. Архитектурная аналогия с асинхронными framework

22. Акторная модель

1. Суть модели:
    - Передача сообщений
    - Легковесные процессы
    - BEAM
2. Применение в современных языках:
    - Erlang
    - Elixir

23. Java.util.concurrent и Fork-Join Framework

1. Пулы потоков, корректное завершение пула
2. Контроль задач через Future
3. CompletionStage и CompletableFuture
4. Потокобезопасные контейнеры

24. Wait-free MRMW снимок регистров

1. Напоминание о MRSW алгоритме
2. Переход к *bounded* версии на битовых *handchake*
3. Расширание до MRMW

25. Средства поиска ошибок

1. Google Thread Sanitizer
2. Intel Parallel Studio
3. Valgrind (модуль helgrind)
4. Пример использования

26. Lock-free изнутри

1. Feldman Multi Array
2. Схемы управления памятью:
    - Tagged pointers
    - Hazard pointer

27. Оптимизации в реализации контейнеров

1. Relaxed SkipList

From:
http://wiki.osll.ru/ - **Open Source & Linux Lab**

Permanent link:
http://wiki.osll.ru/doku.php/courses:high_performance_computing:lectures?rev=1648130403

Last update: **2022/03/24 17:00**