Теория и практика многопоточного программирования

ЛЕКЦИЯ 10. ПРО ПОДОХОДЫ К СИНХРОНИЗАЦИИ

В прошлый раз...

Универсальный объект

Невозможность консенсуса в системе со сбоями

Темы лекции

Виды замков

5 подходов к синхронизации

Задачки

Ещё два слова о разновидностях замков

Мониторы – объекты, умеющие в критической секции «приотпускать» замок, чтобы после наступления какого-то события (await for condition) возобновить свою работу (конкурирую за замок на общих началах). *Пример*: dequeue() ждёт, пока в очереди не появится элемент. Проблема мониторов – Lost wake-up – ненаступление условия – лечится установкой timeout'ов.

Reader-Writer locks – замки, устанавливающие уровень доступа к данным. Возможен захват reader-замка несколькими потоками и невозможен при захваченном writer-замке. *Пример*: select/update в базе данных, доступ к файлу.

Reentrant lock – замок, который можно повторно захватить, если он уже принадлежит исполняющемуся потоку.

Semaphore – примитив, пускающий в «критическую секцию» не более N потоков. Пример: распределение ограниченных (счётных) ресурсов – сокетов, ядер, IO-операций, подключений к базе, лицензий...

Синхронизация в структурах данных

Использование Mutex при работе со структурами данных – «*coarse-grained synchronization*»

- часто блокирует больше, чем реально стоило
- Иногда блокировка не нужна, если можно обойтись атомарностью
- Scalable lock не означает, что не будет contention (из-за объекта)

Синхронизация в структурах данных

Fine-grained synchronization

- Вместо использования одного замка для всех операций с объектом, объект разбивается на независимо-синхронизируемые части. Конкуренция только при доступе к одному компоненту

Optimistic synchronization

- Отказ от использования замков и проверка, что объект за время доступа не изменился. Подходит для случаев, когда успех очень вероятен

Lazy synchronization

- Откладывание на «потом» сложных операций. Например, пометить объект как удалённый

Nonblocking synchronization

- Отказ от использования замков в пользу использования атомарных операций вроде CAS

Вопрос?

Что общего у задачек:

- «обедающие философы»
- «дилемма заключённого»
- про людоеда и шапки 2 цветов

Как бы вы решили задачу…

1) У вас есть регулярно появляющиеся БОЛЬШИЕ текстовые файлы (e.g. логи). И многоядерная машина. Нужно файлы БЫСТРО и ЭФФЕКТИВНО сжимать и загружать на удалённые FTP сервер с лимитированным числом МЕДЛЕННЫХ подключений (e.g. Amazon cloud).

2) Можно ли **кратно** увеличить производительность сайта, если при установленном Load Balancer’е умножить в N раз число ядер и виртуальных машин с экземплярами сайта на вашем сервере?

3) Можно ли ускорить или упростить привычные жизненные процедуры, используя подходы параллельного программирования? Как?

- Медкомиссия, посадка-высадка в метро, посадка в забитое кафе, …?
- Почему электронная очередь в Сбербанке работает медленнее обычной? Почему так сделано и что сделано не так?