Мультипроцессорность

Кэш процессора

- Современные процессоры состоят из несколько ядер
- Каждое ядро (обычно) имеет собственный L1 и L2 кэш
- Единый L3 кэш на все ядра
- Чтение из основной памяти происходит большими выровненными кусочками — кэш линиями (cache line)
- Размер кэш-линии обычно 64 байта

Когерентность кэшей

- Механизм синхронизации данных во всех L1/L2 кэшах
- Протоколы когенернтности кэшей (например, MOESI)

Модель памяти x86

Модель памяти x86

- Модель памяти определяет то, как будут видны инструкции исполненные на одном ядре в остальных ядрах
- => имеет смысл только если есть несколько ядер :)

Модель памяти x86: атомарные операции

Следующие операции всегда атомарны:

1. Чтение/запись байта
2. Чтение/запись выровненного word (16 бит)
3. Чтение/запись выровненного double word (32 бита)
4. Чтение/запись выровненного quad word (64 бита)
5. Чтение невыровненных word/dword/qword помещающихся в одну кэш-линию

Модель памяти x86: lock prefix

- Специальный *префикс* команд
- Работает только на write инструкциях, иначе `#UD`
- «Спинлок» для ядер процессора на регион памяти

Модель памяти x86: orderings

- ISDM, Volume 3A, 8.2

Модель памяти x86: memory barriers

- SFENCE — store barrier
- LFENCE — load barrier
- MFENCE — full memory barrier
- SFENCE + LFENCE != MFENCE

Модель памяти x86: serializing instructions

- `lgdt`, `lidt`, `mov crX`, `wrmsr`, etc
- Не только являются барьерами памяти, но и ждут прекращения out-of-order действий

MP boot protocol

MP boot protocol

- ISDM, Volume 3A, 8.4
- BSP = bootstrap processor, хардварно выбираемый процессор, который осуществляет предварительную загрузку и инициализацию, ОС начинает работать в нём
- AP = application processor, все остальные процессоры, на момент старта ОС находятся в специальном спящем состоянии
- Получить информацию о ядрах в процессоре можно с помощью ACPI таблицы MADT

MP boot protocol

- Чтобы AP начал выполнять код, ему нужно послать специальные прерывания через APIC: initialization IPI (INIT) и startup IPI (SIPI)
- SIPI содержит в себе адрес, по которому AP начнёт исполняться
- Проблема: AP начинает исполняться аж в *real mode* (16-битный режим)
- AP trampoline – кусочек кода, который переводит AP в 64-битный режим
- Два варианта: последовательная инициализация всех AP или бродкаст SIPI

Intel Hyper-threading

- Simultaneous multithreading (SMT)
- По два логических ядра на физическое
- Процессор может *простаивать* (например, при чтениях памяти), в этот момент можно запустить исполнение инструкций на соседнем логическом ядре

RCU

- Read-copy-update
- Механизм синхронизации
- Может быть использован в любой среде, однако из-за специфики ОС может работать более эффективно

RCU

- Создать новую структуру, скопировав данные из старой, изменить новую
- Изменить глобальный указатель на новую структуру
- Дождаться, когда все читатели старой стуктуры закончат работу с ней
- Удалить старую структуру

```
void reader() {
    rcu_read_lock();
    struct_t* s = load_ptr();
    // ...
    rcu_read_unlock();
}
```

```
void writer() {
    struct_t* old = load_ptr();
    struct_t new = struct_from_old(old);
    store_ptr(new);
    synchronize_rcu();
}
```

RCU

Как реализовать `rcu_read_lock` / `rcu_read_unlock` / `synchronize_rcu`?

RCU: ✨✨✨✨

```
void rcu_read_lock(void) { }

void rcu_read_unlock(void) { }

void synchronize_rcu(void) {
    for_each_possible_cpu(cpu) {
        run_on(cpu);
    }
}
```

RCU: 🪄🪄🪄🪄

- Ядро запрещает любые операции переключения контекста внутри rcu_read_lock/rcu_read_unlock
- => если переключение контекста случилось, то ядро вышло из read-секции
- => если запустить себя последовательно на всех процессорах, мы выйдем из всех read-секций
- Подробнее про RCU

Per-cpu переменные

- Аналог thread-local storage, но для каждого ядра процессора
- Помещаются в отдельную секцию (`.percpu`)
- `#define PER_CPU(name) (GS + (&name - &_percpu_start))`
- При загрузке очередного CPU, под секцию выделяется память и её содержимое туда копируется
- GS/FS в 64-bit mode не указывают на дескриптор в GDT, а просто содержат базовый адрес
- `swapgs` меняет местами `GS` и `IA32_KERNEL_GS_BASE`