2.6
Многоядерные/многопроцессорные системы. Одновременная многопоточность. SMT/HT.

Даниэль Ползик

7 января 2020 г.

1 Ядро vs Процессор. Многоядрерность vs Много(мульти)процессорность. В чем разница?

1.1 Ядро - часть процессора

Процессор - это всё, что находится на кристалле. Включая всякие кэши, системы предсказаний и т.п.

Ядро - это вычислитель. Простым языком на ядрах выполняются все команды. Обычно у каждого ядра есть 2 уровня кэшей: L1 (L1i и L1i), L2. И еще у всех ядер процессора есть общий уровень кэша - L3 (подробнее в билете про кэши)

1.2 Немного историчеких справок

1.2.1 Про многоядерные

Уже с 2001 года разные компании такие, как **IBM**, **Intel**, **AMD**, очень хотели создать многоядерные процессоры и даже создавали что-то наподобии.

Это были действительно процессоры с 2-мя ядрами на кристалле, но вот в чем штука. По факту, это были двухпроцессорные системы запиханные в один кристалл и они не сильно взаимодействовали и параллельность была слабовата. По скорости эти "двухядерные процессоры"не были сильно быстрее.

Но к 2005 году начали появляться уже **реальные** без кавычек двухядерные процессоры с измененной под параллельность архетиктурой. А в 2006 Intel выпустил для персональных компьютеров: *Core 2 Duo*

Дальнейшая история не сильно поможет в сдаче экзамена, но если очень хочется - википедия вам в помощь. Если захотелось, то советую ознакомиться с разборками Intel и AMD. Легендарная сага!

1.2.2 Про многопроцессорные

История многопроцессорных систем еще менее полезная, но просто чтобы вы понимали, насколько это было раньше многоядерных процессоров:

Первая система была создана в 1970-х,а в 1980-х появились уже вполне себе рабочие системы.

1.2.3 Пояснение простым языком к историческим справкам

Как вы знаете из прошлых билетов мы создали кучу всяких фич для увелечения скорости: кэши, предсказатели и даже superscalar/VLIW. Но все хотят играть в Доту (которой еще нет), а для этого нужно ускорять работу.

Для этого мы понимаем, что можно добавить на комп еще один процессор, который будет работать с той же оперативной памятью и сможет делать что-то параллельное. **Важно!** Это не тоже самое, что поставить два компа рядом. Оперативка-то общая.

Как вы наверно догадываетесь, два процессора занимают место и один из них находится сильно дальше, чем другие от опертивки и нужных вещей, потому либо один процессор работает быстро, а остальные сильно медленее (тогда зачем они нужны), либо все процессоры равноудалены и тогда их всех близко не расположить и значит все они работают медленне. К тому же еще фиг знает, как распределять работу между процессорами. В те времена это скинули на программистов. Но программ, которые бы хорошо параллелились сильно больше не стало и в итоге почти всегда работал только один процессор из всех.

Короче говоря, фича с несколькими процессора конечно ускоряла, но не так, как хочется. А хочется, чтобы ты удвоил кол-во процессоров и скорость увеличелась в 2 раза. Но это было далеко не так. Еще и не всегда ускоряет. Короче, сплощное фиаско, еще и стоит дороже.

P.S. В те далекие времена процессор состоял только из вычислителя и поэтому тогда считалось: *процессор = ядро* Дальше мы уже говорим о процессорах помоднее, на которые напиханы предсказатели, кэш, ви-

деокарточка, вычислители и уже оставшееся место занимает исполнитель(ядро). То есть теперь: *процессор* $\neq$ *ядро*

Наука развивается и компании хотят это использовать. И что же они делают? Идейно: они хотят запихнуть на один кристалл несколько ядер (для начала хотя бы два), так как наука стала позволять это делать. Но для этого нужно переделать всю архитектуру процессора, а компании про которые мы говорим уже имеют надежные архитектуры, но для одного ядра. И что же вы думаете они делают? Да, они берут пихают на один кристалл грубо говоря два процессора, которые друг с другом почти не взаимодействуют.

Вроде бы хорошо, но скорости для доты всё еще не хватает, так как остаются те же проблемы, что и у многопроцессорных систем. Тогда разработчикам архитектур приходится оторвать свои пятые точки и начать создавать нормальные архитектуры для двух ядер. Что Intel успешно и сделал в 2006 году. Но по факту, проблемы с тем, как распараллелить остались, но благодаря новой архитектуре стало возможным добавление фич, которые действительно параллелят. И к тому же все стало ближе друг к другу, а это не хухры мухры.

2 Но как то же параллелят. Как?

2.1 Иллюзия параллельности. Прерывания

Прерывание (*interrupt*) - сигнал процессору о событии, которое требует его внимания. Процессор отвечает на сигнал прерывания, останавливая выполнение текущей задачи, сохраняя её состояние и вызывает **обработчик прерываний** (*interrupt handler).*

Бывает два вида прерываний:

Аппаратное прерывание используется устройствами чтобы сообщить о том, что им требуется внимание со стороны операционной системы. Например, нажатие клавиши на клавиатуре вызывает прерывание, после чего процессор считывает, какая клавиша была нажата, а затем продолжает выполнение.

Программное прерывание вызывается программным исключением (например, делением на ноль) или специальной инструкцией, которая вызывает прерывание при исполнении.

ОС с каким-то промежтком переключается между задачами, прерывая одну и запуская другую. Так как это происходит быстро, то создается

иллюзия многозадачности(параллельности).

2.2 Иллюзия параллельности. SMT/HT

2.2.1 Процессы. Треды

Процесс - экземпляр компьютерной программы, запущенный на выполнение. У каждого процесса своя память и свои права.

Тред - наименьшая последовательность инструкций, которая может независимо управляться планировщиком. У каждого треда свои регистры (IP и общего назначения), свой выделенный стек в общей для процесса оперативной памяти (треды не пересекаются).

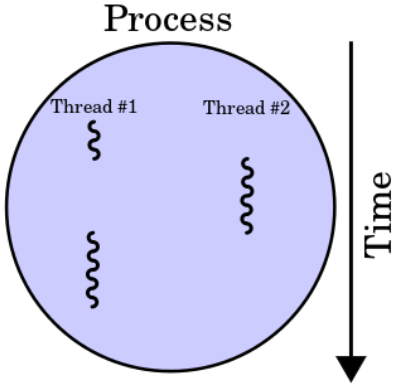

Рис. 1: Процесс с двумя тредами, запущенными на одном процессоре

Теперь распределение памяти происходит на уровне процессов, а распределение вычислительной мощности - на уровне тредов. Но чтобы вся эта прелесть работала и давала ускорение, нужно, во-первых, чтобы задачу вообще можно было распараллелить, а во-вторых, программисту нужно правильно написать.

2.2.2 SMT/HT

SMT - *Simultaneous multithreading* - техника, улучшающая эффективность работы суперскалярной архитектуры. Один физический CPU притворяется двумя ненастоящими(логическими). Треды независимы, поэтому планировщику проще найти независимые команды, следовательно конвееры эффективнее используются.

HT - *Hyperthreading* - SMT у Intel. Ядро с SMT должно иметь больше регистров (отдельные для каждого треда).

В лучшем случае мы можем получить выигрыш в два раза. В худшем тредам может не хватить кэша и тогда бо-бо. И мы очень сильно проигрываем.

Выигрыш от SMT получается, если:

- Тредов больше, чем физических ядер
- Каждый тред "написан не очень хорошо т.е. не по максимуму использует конвееры.

А проигрываем так же, как и в HT при нехватке кэша на треды.

Рис. 2: Удачи!