## Планировщики ввода/вывода (I/O Schedulers)

**I/O Scheduling** — общее название метода управления очередью операций ввода-вывода к жесткому диску и планировании данных операций компьютерных операционных систем.

Зачем знать ? Можно повысить производительность системы.

* Приоритизация некоторых запросов ввода-вывода
* Увеличение производительности дисков
* Гарантия исполнения важных запросов в кратчайшие сроки

Способы

* слияние запросов. Суть его заключается в объединении нескольких запросов к физическим разделам жесткого диска в один и тем самым уменьшении операций ввода-вывода.
* установка приоритет выполнения - очень старые запросы имеют высший приоритет выполнения перед вновь поступившими

Non-multiqueue I/O schedulers

NOTE: Non-multiqueue have been deprecated

* noop
* dealine
* cfq

Смена планировщика

/sys/block/<device>/queue/iosched

echo SCHEDNAME > /sys/block/DEV/queue/scheduler

# cat /sys/block/sda/queue/scheduler

[mq-deadline] kyber bfq none

# echo none >/sys/block/sda/queue/scheduler

# cat /sys/block/sda/queue/scheduler

[none] mq-deadline kyber bfq

Стандартные планировщики

В большинстве современных дистрибутивов Linux доступны три стандартных планировщика: noop, CFQ и deadline.

Название noop — это сокращение от no operation, которое наводит на мысль о том, что этот планировщик ничего не делает. На самом деле это не так: noop реализует простую FIFO-очередь и объединение запросов к соседним блокам. Он не изменяет порядка следования запросов и в этом полагается на нижележащий уровень — например, на возможности Raid-контроллера по планированию.

CFQ (Completely fair queuing) работает сложнее. Он делит пропускную способность между всеми процессами. Для синхронных запросов создается по очереди на процесс. Асинхронные же запросы объединяются в очереди по приоритетам. В дополнение к этому CFQ сортирует запросы, чтобы минимизировать поиски секторов на диске. Время на выполнение запросов между очередями распределяется согласно приоритету, который можно настроить при помощи утилиты ionice. При настройке можно изменить как класс процесса, так и приоритет внутри класса. Мы не будем подробно останавливаться на настройках, отметим только, что этот планировщик обладает большой гибкостью, а настройка приоритета по процессам — весьма полезная вещь для ограничения выполнения административных задач в угоду пользовательским.

Планировщик deadline использует другую стратегию. За основу берется время нахождения запроса в очереди. Таким образом он гарантирует, что каждый запрос будет обслужен планировщиком. Так как большинство приложений блокируются именно на чтении, то deadline по умолчанию отдает приоритет запросам на чтение.

Механизм blk-mq был разработан после появления NVMe-дисков, когда стало ясно, что стандартные средства ядра не дают необходимой производительности. Он был добавлен в ядро 3.13, однако планировщики были написаны позже. В новейших версиях ядра Linux (≥4.12) имеются следующие планировщики для blk-mq: none, bfq, mq-deadline и kyber. Рассмотрим каждый из этих планировщиков подробнее.

Планировщики blk-mq: краткий обзор

Используя blk-mq, можно действительно отключить планировщик, для этого достаточно установить его в none.

Про BFQ написано достаточно много, скажу только, что он наследует часть настроек и основной алгоритм от CFQ, привнося понятие бюджета и ещё несколько параметров для тюнинга.

Mq-deadline — это, как легко догадаться, реализация deadline с использованием blk-mq.

И последний вариант — kyber. Он был написан для работы с быстрыми устройствами. Используя две очереди — запросы на запись и на чтение, kyber отдает приоритет запросам на чтение, перед запросами на запись. Алгоритм измеряет время завершения каждого запроса и корректирует фактический размер очереди для достижения установленных в настройках задержек.