IOExecutor

The IOExecutor is a wrapper over the Linux kernel's async IO feature. It accepts FilerJob requests whose operation type can be read, write, delete.

Initialization

The IOExecutor accepts a configuration structure that specifies which core to run one, what is the size of the kernel io\_context to allocate, etc.

On initialization, it starts 2 threads on the specified core

1) FdQueue processing thread for jobs which cannot be submitted via io\_submit (e.g. delete a file)

2) Completion thread which handles callbacks for jobs submitted to the kernel.

It creates an IO context with the specified depth for use with io\_submit().

It maintains a queue of submitted FilerJobs using boost lockfree queue.

The files are opened with O\_DIRECT to bypass the filesystem buffer cache. Therefore, the FilerJob buffer has to be aligned with 512 bytes.

Call flows

Call-graph for async read/write

User thread

Kernel thread

Completion Thread

IOExecutor->submitTask(FilerJob)

add to RequestQueue

if queue more than minSubmitSize, call io\_submit() for all jobs, else return

block device queue

job sent to SSD

CPU interrupt

epoll()

io\_getevents()

FilerJob->reset

Call graph for Sychronous operations

User thread

FdQueue Thread

IOExecutor->submitTask(FilerJob)

add to FdQueue

signal condition variable

wakeup on condition

synchronously execute operation

FilerJob->reset

Error handling

Job can encounter an error before io\_submit() or inside the kernel. In either case, the FilerJob->errcode is set.

Statistics

The IOExecutor records the standard queue-related parameters

1. wait time is the difference between call to submitTask and the time job was submitted to the kernel.

2. service time is the difference between the time job was submitted to the kernel and the time the job was notified as finished to the user.

It maintains Average, Min, Max, Std deviation as well as a histogram for wait time and service time for read and write requests.

It also records the Max Queue Size reached during execution

Shutdown

Shutdown is initiated by calling IOExecutor->stop(). It is a two-stage process

First new requests are prevented from entering. Then it Waits until all submitted requests have finished.