Asynchronous IO vs /dev/epoll

Benjamin LaHaise
bcrl@redhat.com

Overview

- Why do IO models matter?
- Existing IO models under Linux
- What is /dev/epoll?
- What is AIO?
- Performance testing results
- Conclusions
- Future Plans

Why do IO models matter?

- Linux isn’t used just by kernel hackers anymore
  - Database servers
  - Content servers
  - Multimedia applications
  - LDAP servers
- Current models do not scale well enough
- The Linux kernel is capable

Existing IO models under Linux

- select(n, *readfds, *writefds, *exceptfds, *timeout)
- poll(struct pollfd *ufds, nfds, timeout)
- SIGIO
- processes
- threads

What is /dev/epoll?

- Based on poll()
- mmap()ed event buffer
- Register interest by write()ing a struct pollfd
- One syscall per loop

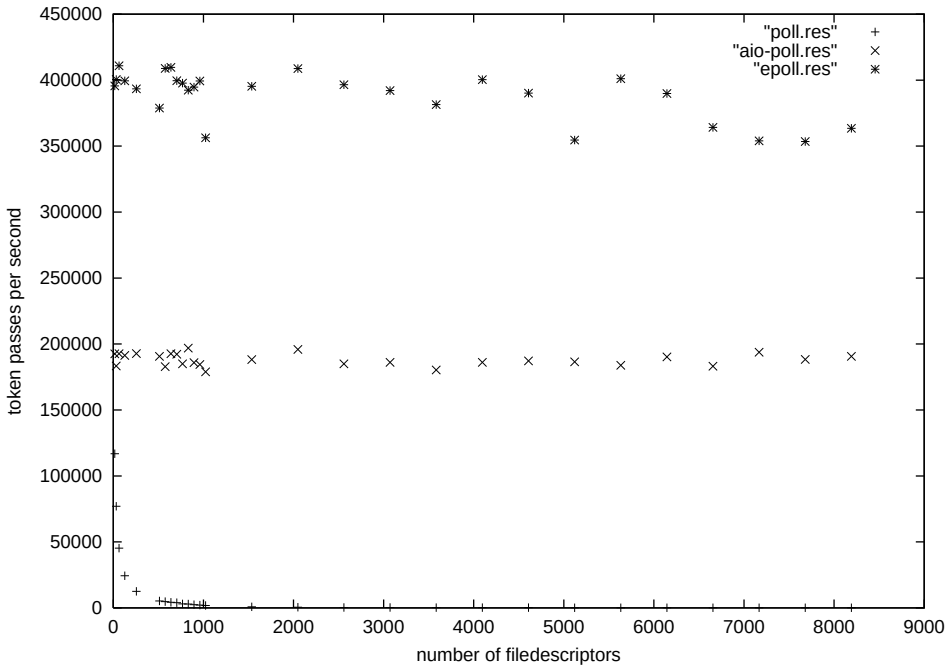
What is AIO?

- Splits IO submission and completion
- Allows overlapping IO without threading
- Enables reduced copy IO
  - O_DIRECT

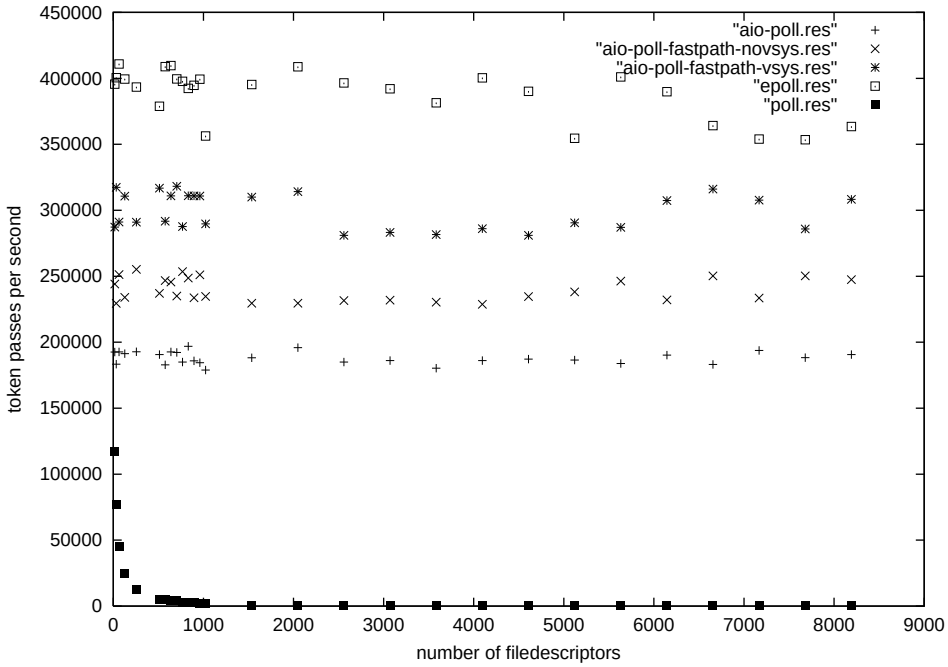
AIO API

- io_setup(nr_reqs, *ctx)
- io_destroy(ctx)
- io_submit(ctx, nr, **iocbs)
- io_cancel2(ctx, *iocb, *event)
- io_getevents(ctx, nr, *events, *timeout)

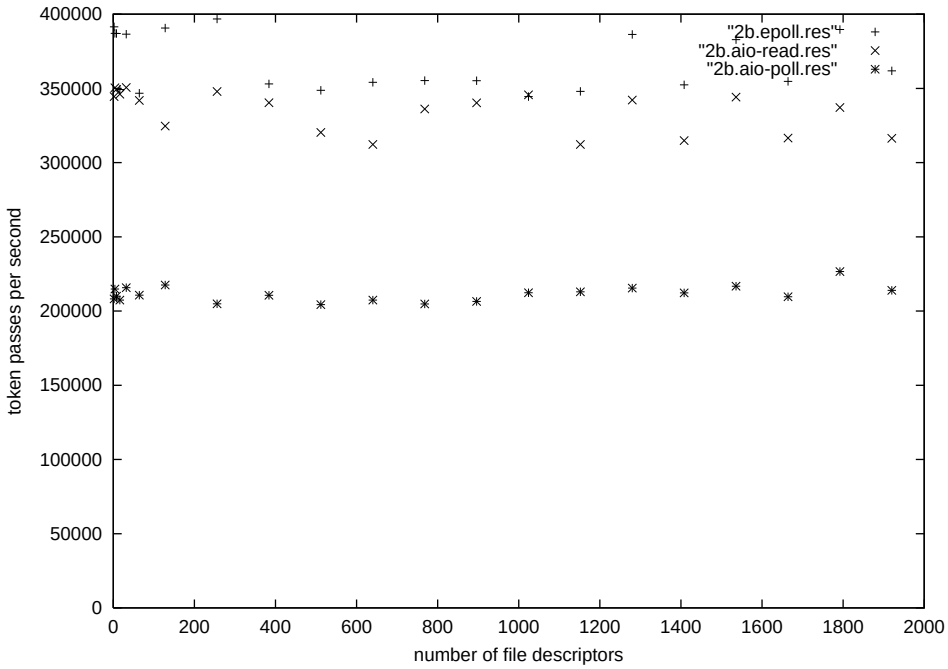
Initial Testing Results



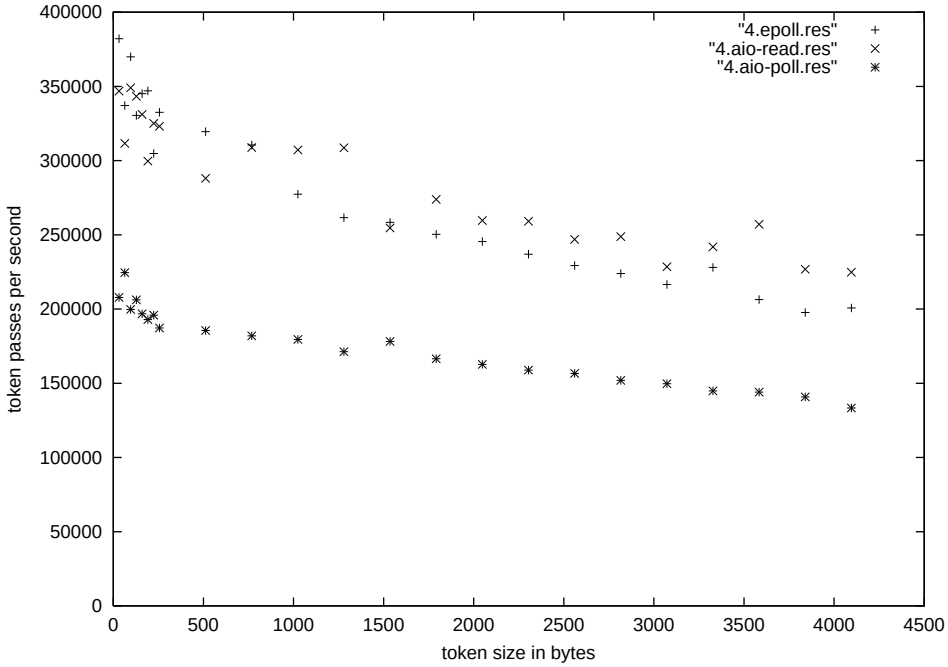
Results with the addition of an async poll fast path and vsyscall mechanism



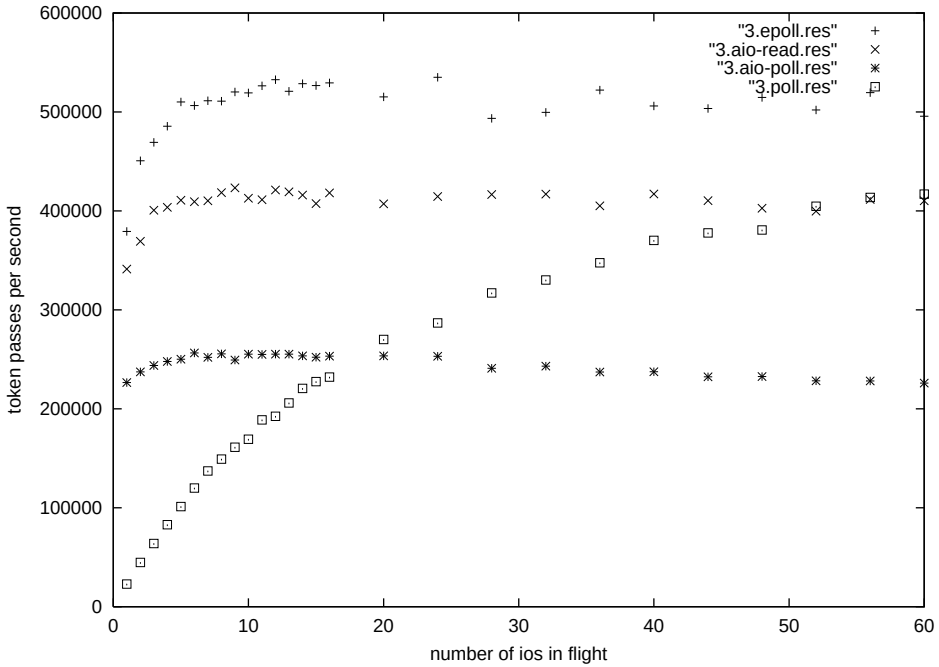
Async read added to the mix



Throughput as message size increases



Throughput while increasing the number of IOs in flight (128 fds)



Conclusions

- /dev/epoll is pretty damned good
- Zero / single copy optimizations really help
- AIO is a scalable interface

Future Plans

- Merge plans in 2.5
- Field testing
- More experimentation
  - sendfile()
  - mread() / mwrite()