CS241 #34 – Disks & Signals

> **Solid State Drives/Solid State Disks (SSD)s**

NAND Flash. Use quantum tunneling through an insulator to write/erase! Limited number of erasures. No moving parts.

Requires separate controller:

Error-correcting code (ECC)

Wear leveling

Bad block mapping

Read scrubbing and read disturb management

Read and write caching

Garbage collection

Encryption

Very fast random access & throughput. Can be limited by bus speeds.

Benchmarks measure 'IOPs'

**> Spinning disks**

Cylinders. Platters. Heads.

Two common rotational speeds

5400 & 7200rpm. 7200rpm = \_\_\_\_\_\_ revolutions per second

How many milliseconds for one revolution ? \_\_\_\_\_\_

Access time?

Average seek time ~ 10ms (though seeking to neighboring track ~ 1ms).

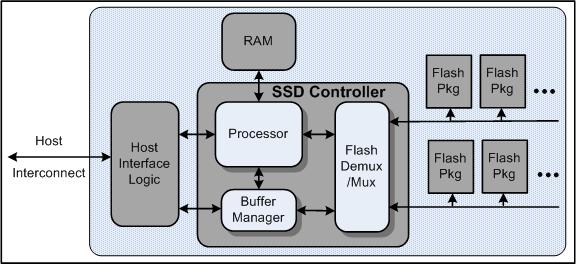
Average rotational latency.

Tiny contributions:

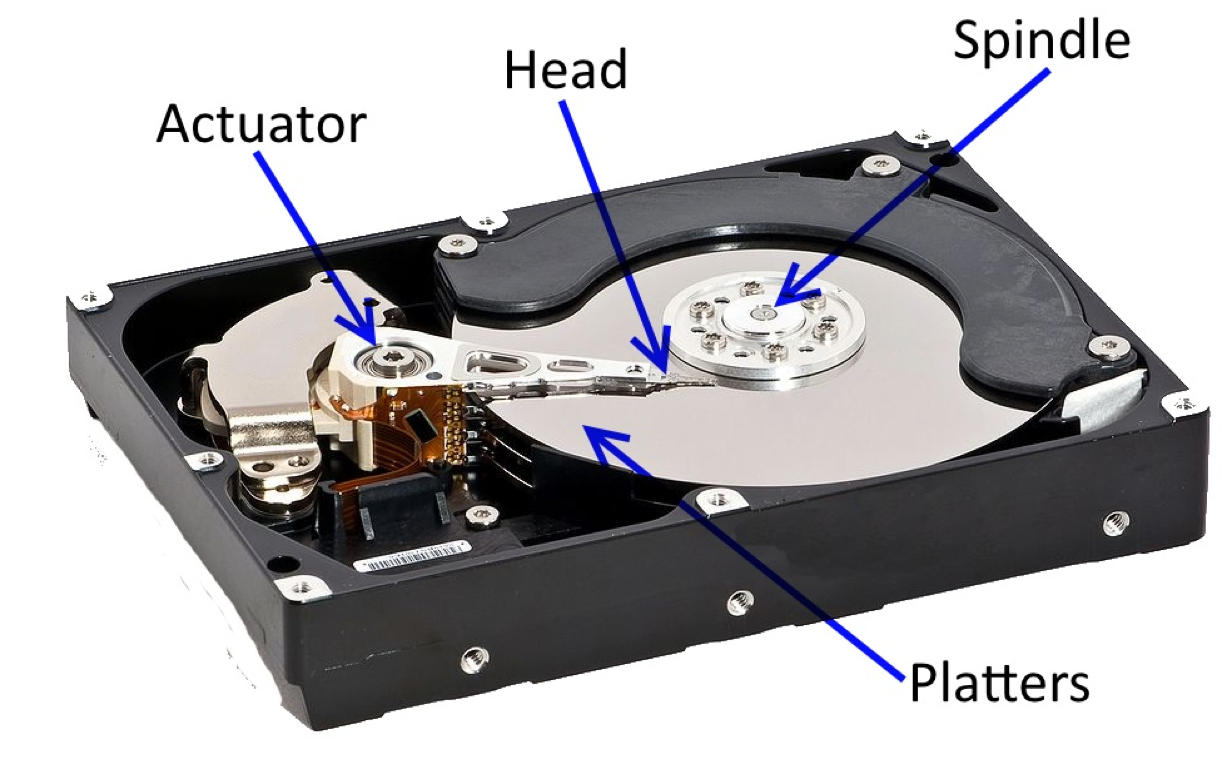
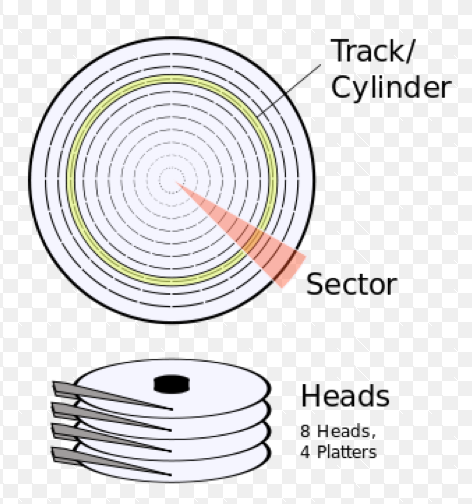
Command processing time (0.003ms)

Head settling time (0.1ms)

IOPs?



Source: https://www.usenix.org/legacy/event/usenix08/tech/full\_papers/agrawal/agrawal\_html/index.html

**> Signals**

For more information man -s7 signal

**Can signals be queued?**

Signal terminology.

Generated

Pending?

Blocked

Delivered

Caught

Disposition

Signal disposition per thread or per process?

Signal disposition after fork?

after exec?

**What is signal masking ?**

When would I use sigprocmask ?

When would I use pthread\_sigmask ?

So which thread will get the signal?

**What are Pending signals ?**

From man -s7 signal

"A child created via fork(2) initially has an empty pending signal set; the pending signal set is preserved across an execve(2)."

**Sending signals?**

pthread\_kill ?

raise

kill?

**Catching signals ?**

sigwait ? signalfd ? signal ? sigaction?

Challenge:

Write a program that demonstrates sigprocmask to block and then unblocks a signal.

Write a program that demonstrates that pthread\_sigmask is working?

How would you show that a particular thread handles a signal?

**What's wrong with signal ?**

**How do I use sigaction ?**

int sigaction(int signum, const struct sigaction \*act, struct sigaction \*oldact);

struct sigaction {

void (\*sa\_handler)(int);

void (\*sa\_sigaction)(int, siginfo\_t \*, void \*);

sigset\_t sa\_mask;

int sa\_flags;

};

struct sigaction sa;   
sa.sa\_handler = handler;

sigemptyset(&sa.sa\_mask); //Also *sigfillset*

sa.sa\_flags = SA\_RESTART; /\* Restart functions if interrupted by handler \*/

sigaction(SIGINT, &sa, NULL)

How do we complete and fix this code to catch SIGCHILD ? (hint WNOHANG and a while loop will be useful here)

int dezombify(int signal) {

int status;

pid\_t child;

child = waitpid(-1, &status, 0);

}

struct sigaction sa;   
sa.sa\_handler = \_\_\_\_\_\_\_\_\_\_\_\_\_;

sig\_\_\_\_\_\_\_\_ (\_\_\_\_\_);

sa.sa\_flags =

sigaction(\_\_\_\_\_\_\_\_, &sa, NULL)

How do I set a threads mask? Why would I want to?

**int pthread\_sigmask(int** *how***, const sigset\_t \****set***, sigset\_t \****oldset***);**

What happens to the new thread during pthread\_create to *pending* signals and the thread's signal mask?

What do I use pthread\_sigmask and sigwait?

(source http://pubs.opengroup.org/onlinepubs/009695399/functions/pthread\_sigmask.html)

static sigset\_t signal\_mask; /\* signals to block \*/

int main (int argc, char \*argv[])

{

pthread\_t sig\_thr\_id; /\* signal handler thread ID \*/

sigemptyset (&signal\_mask);

sigaddset (&signal\_mask, SIGINT);

sigaddset (&signal\_mask, SIGTERM);

pthread\_sigmask (SIG\_BLOCK, &signal\_mask, NULL);

pthread\_create (&sig\_thr\_id, NULL, signal\_thread, NULL);

/\* APPLICATION CODE \*/

...

}

void \*signal\_thread (void \*arg)

{

int sig\_caught; /\* signal caught \*/

sigwait (&signal\_mask, &sig\_caught);

switch (sig\_caught)

{

case SIGINT: /\* process SIGINT \*/

...

break;

case SIGTERM: /\* process SIGTERM \*/

...

break;

default: /\* should normally not happen \*/

fprintf (stderr, "\nUnexpected signal %d\n", sig\_caught);

break;

}

}

Lecture demo notes:

man –s7 signal "A child created via fork(2) inherits a copy of its parent’s signal

dispositions. During an execve(2), the dispositions of handled

signals are reset to the default; the dispositions of ignored sig-

nals are left unchanged. "

=> Can create uninterruptable child using fork (until evec)

SIG\_SETMASK SIG\_BLOCK SIG\_UNBLOCK

sigprocmask – not in multithreaded program! Use pthread\_sigmask

int sigprocmask(int how, const sigset\_t \*set, sigset\_t \*oldset)

int pthread\_sigmask(int how, const sigset\_t \*set, sigset\_t \*oldset)

while( (child= waitpid(-1, &status, WNOHANG)) >0