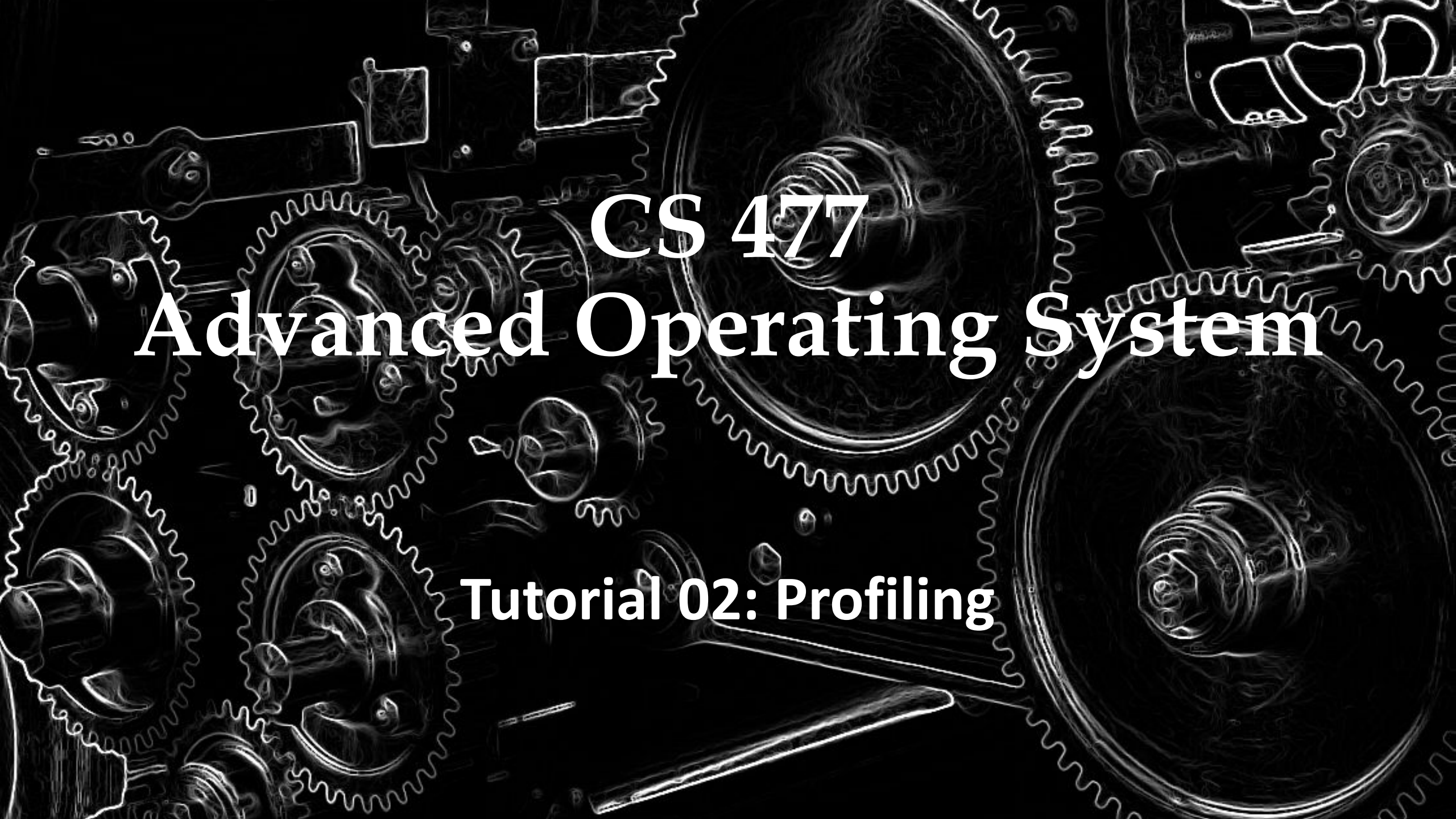
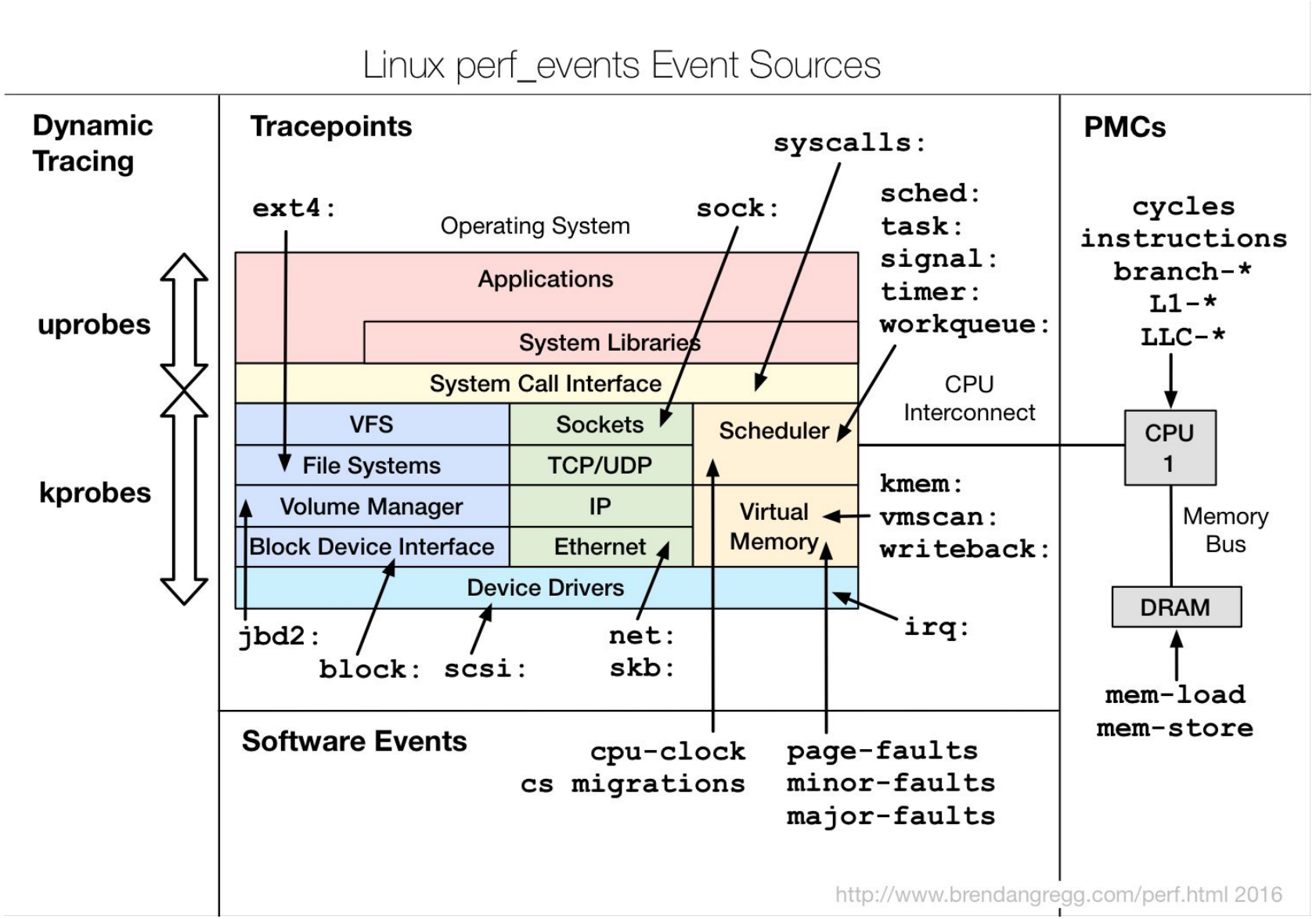


CS 477
Advanced Operating System

Tutorial 02: Profiling

Event Sources



Useful tools

vmstat: for virtual memory stats
- vmstat -s

iostat: monitor system input/output device load
-

pidstat: monitor stats for a particular program.
- pidstat -p <pid>
- pidstat -d ->
- pidstat -r -> memory

Perf cheatsheet

```
# List the available events
perf list

# Overview
perf stat -d <command>

# Counting events
# Count the number of syscalls
perf stat -e cycles,instructions,cache-references,cache-misses

# system calls
sudo perf stat -e 'syscalls:sys_enter_*'

# GUPS demo
perf stat -e dTLB-loads,dTLB-load-misses <command>
```

Perf cheatsheet

```
# Sampling events
perf record / perf report --stdio

# Two main switched
perf record -c -> count
perf record -F -> freq


# Sample page faults with stack traces, until Ctrl-C:
perf record -e page-faults -ag
```

https://www.brendangregg.com/perf.html#OneLiners

bpftrace

Bpftrace is a high level language for profiling. Uses bpf under the hoods.

https://github.com/bpftrace/bpftrace

Language basics:
https://bpftrace.org/docs/release_023/language
https://bpftrace.org/docs/release_023/stdlib

bpftrace

```
sudo bpftrace -l
```

```
sudo bpftrace -e 'BEGIN { printf("Hello world") }'
```

```
sudo bpftrace -e '
t:syscalls:sys_enter_write {
    @start[tid] = nsecs;
}

t:syscalls:sys_exit_write /@start[tid]/ {
    $delta = nsecs - @start[tid];
    @latency = hist($delta/1000);
    delete(@start[tid]);
}'
```