

# Graceful Failure During OOM Conditions

# Embedded Systems

- Combo of hardware and software with specific purpose
- May be very simple or relatively complex
- Some run variants of linux
- Examples: remote sensors, medical devices, self driving car radar or gps, elevator control units, etc..



# Memory Allocation And OOM Killer



- Linux intends for “good” malloc() calls never to fail
- If a mmap/malloc call fails due to lack of memory, OOM killer is activated
- The OOM killer tries to kill processes to free memory
- Complicated score to decide which process is killed, but primarily based on size of memory usage
- Adjusted OOM Score in proc fs allows user to make a process less likely (or impossible) to be killed by OOM killer
- But what if all killable processes have been killed?
- Deadlock, unresponsiveness, and system crash as OS processes fail due to bad memory allocation calls.

# Memory, OOM, and Embedded Systems

- OOM conditions are less likely on embedded systems as workload is relatively well known and consistent. However...
- Overcommit may cause OOM
- Coding errors may cause memory leaks
- Fundamental mismatch between hardware and task

# Handling OOM Conditions -- Graceful failure

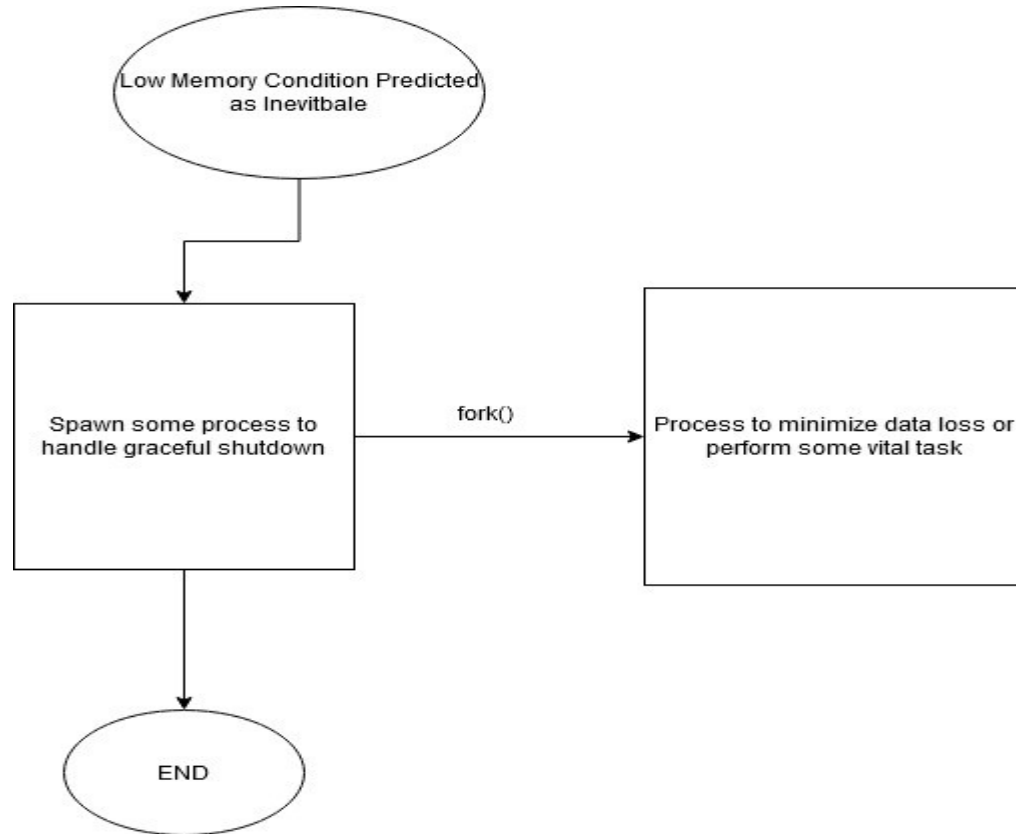
- Currently two options -- let OOM killer kill process or mark process as unkillable and hope system can kill other processes
- In some situations , it may be better for a process to fail and execute a “graceful shutdown” procedure rather than get killed by OOM killer, or cause an unresponsive state.
- Real-time Embedded Systems
  - Medical Devices -- shut down and notify user of failure
  - Data Loggers -- transmit data and then shut down
  - Self Driving Car Radar -- gently apply brakes and notify user to take over

# Design Fulfillment

Three main components of pipeline:

1. Detection/Prediction of Out of Memory Condition
2. Minimising Data Loss / Barebone Functionality
3. Graceful Shutdown

# Spawning Processes



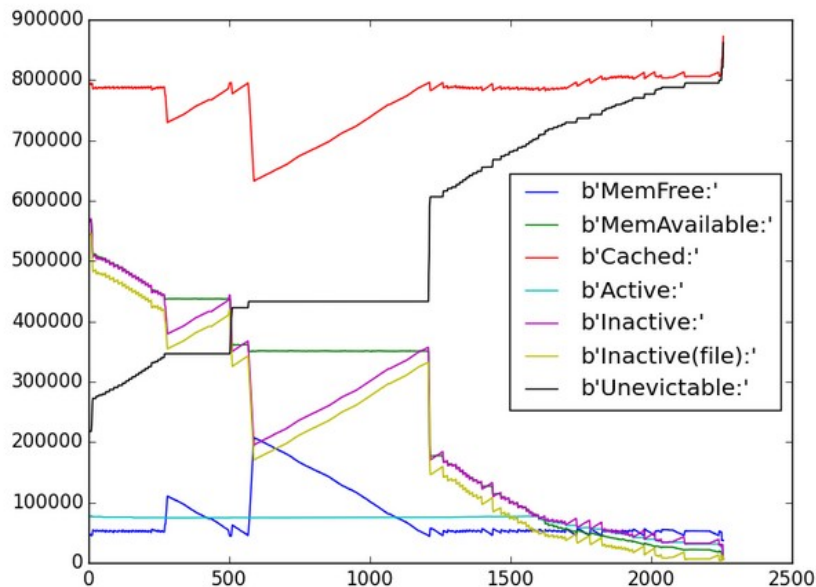
# Spawning Processes cont'd

System Function	Process to Spawn
Data Logging	Write data stream to file
Safety Critical	Barebone Functionality
Monitoring Equipment	Raise Alarm



# Predicting Out of Memory Condition

- Gather Statistics
  - OOM Scores
  - CPU Load
  - /proc/meminfo
- Combination of Factors
- Kickstart Pipeline
  - Minimise Data Loss
    - Or Barebone Functionality
  - Graceful Shutdown



# Graceful Shutdown

Three main components of pipeline:

1. Detection/Prediction of Out of Memory Condition
2. Minimising Data Loss / Barebone Functionality
3. Graceful Shutdown
  - a. Complete any other critical safety-measures
  - b. Call custom shutdown procedure, SIGTERM, SIGKILL?

# Graceful Shutdown

Three main components of pipeline:

1. Detection/Prediction of Out of Memory Condition
2. Minimising Data Loss / Barebone Functionality
3. Graceful Shutdown
  - a. Complete any other critical safety-measures
  - b. Call custom shutdown procedure, SIGTERM, SIGKILL?

# Graceful Shutdown

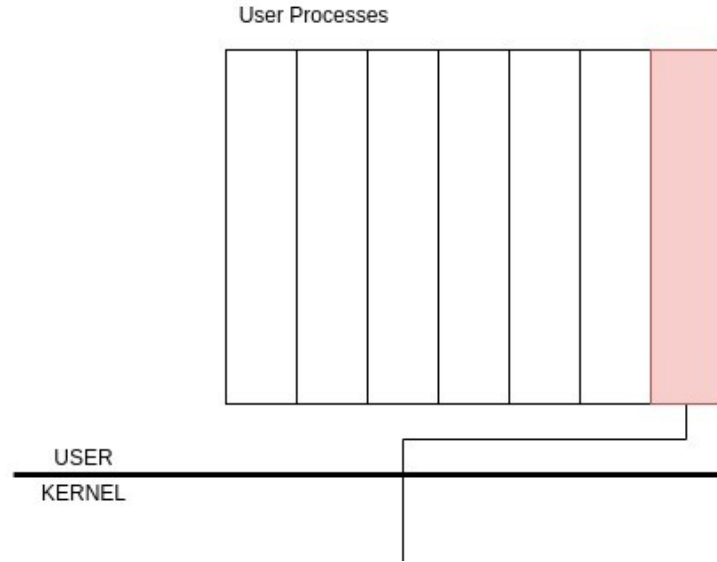
## Current Proposed Solutions

- Kernel Process related to SIGTERM or SIGUSR1
- LKM / System Call Intercept

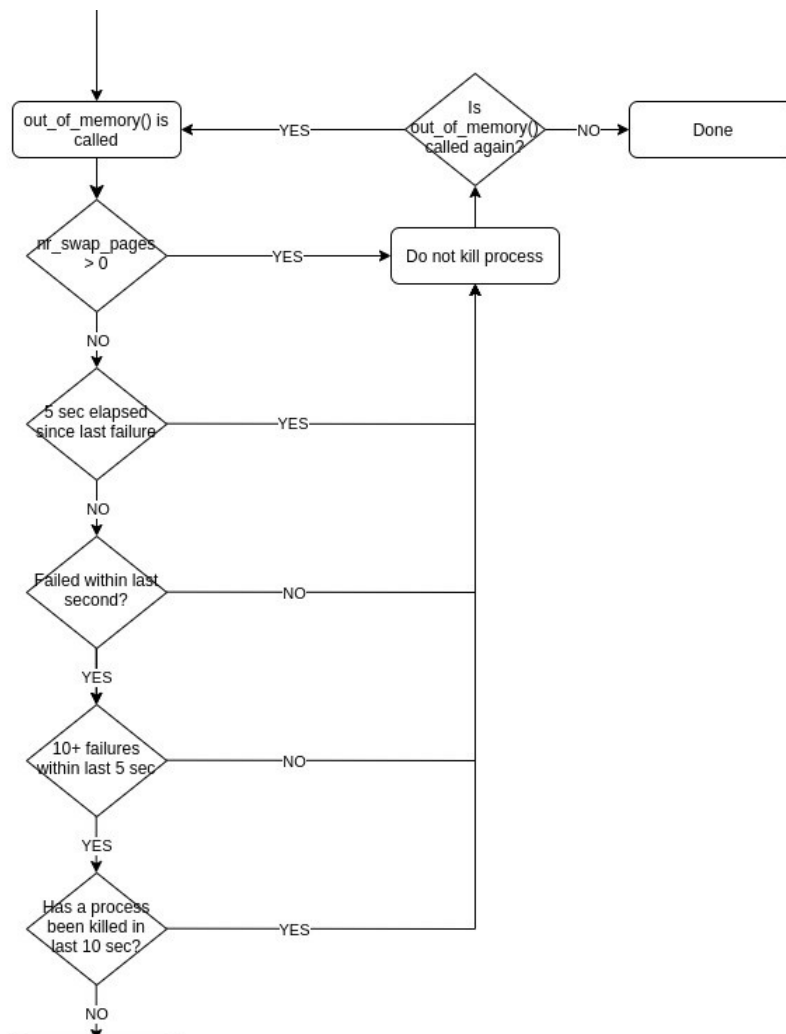
## Solution Requirements

- Customizable across platforms / OS
- Has access to user-level process info and kernel-level

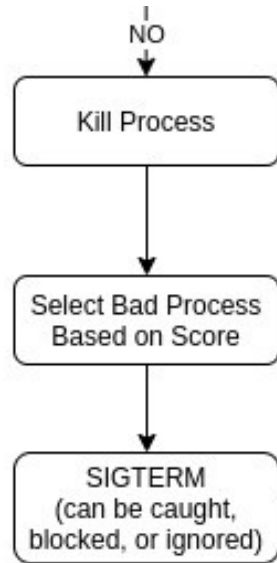
# Graceful Shutdown



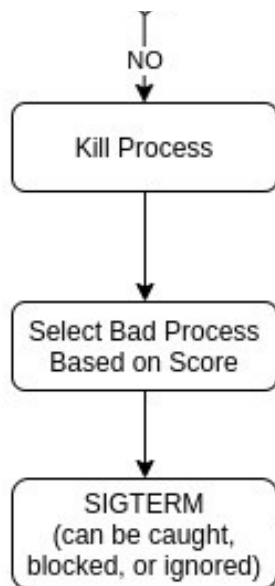
# Graceful Shutdown



# Graceful Shutdown



# Graceful Shutdown



LKM or Kernel process related to SIGTERM, SIGUSR1, to handle graceful shutdown

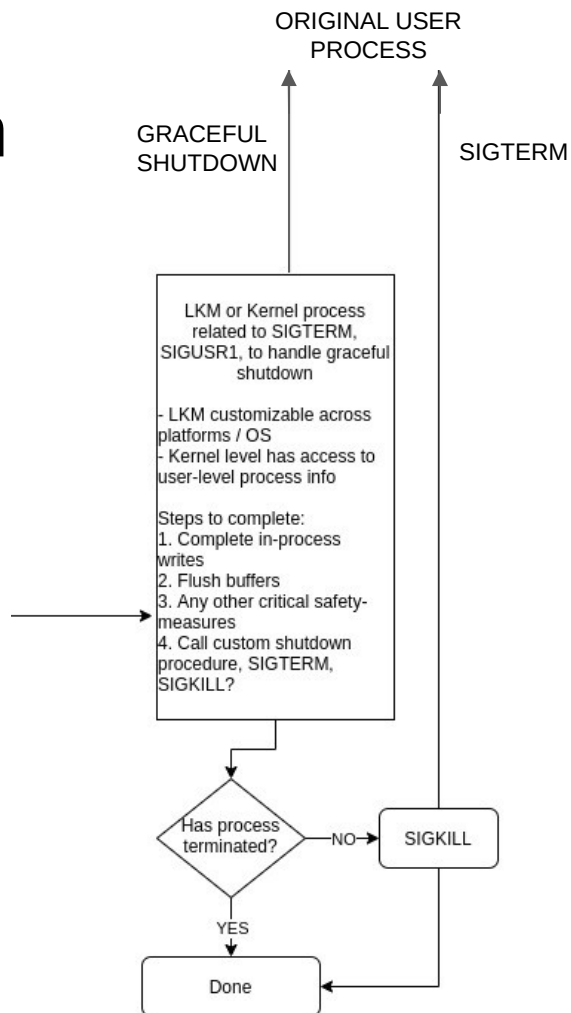
- LKM customizable across platforms / OS
- Kernel level has access to user-level process info

Steps to complete:

1. Complete in-process writes
2. Flush buffers
3. Any other critical safety-measures
4. Call custom shutdown procedure, SIGTERM, SIGKILL?



# Graceful Shutdown



# Graceful Shutdown - Method Pro / Con

## LKM / System Call Intercept

### PRO

- Wraps existing functionality
- Speed (as compared to process)
- Can still invoke kernel process if needed
- Users do not need to rebuild kernel

### CON

- Complexity of newer kernel security
- Memory fragmentation
- Loaded late in boot cycle

## Kernel Mode Process

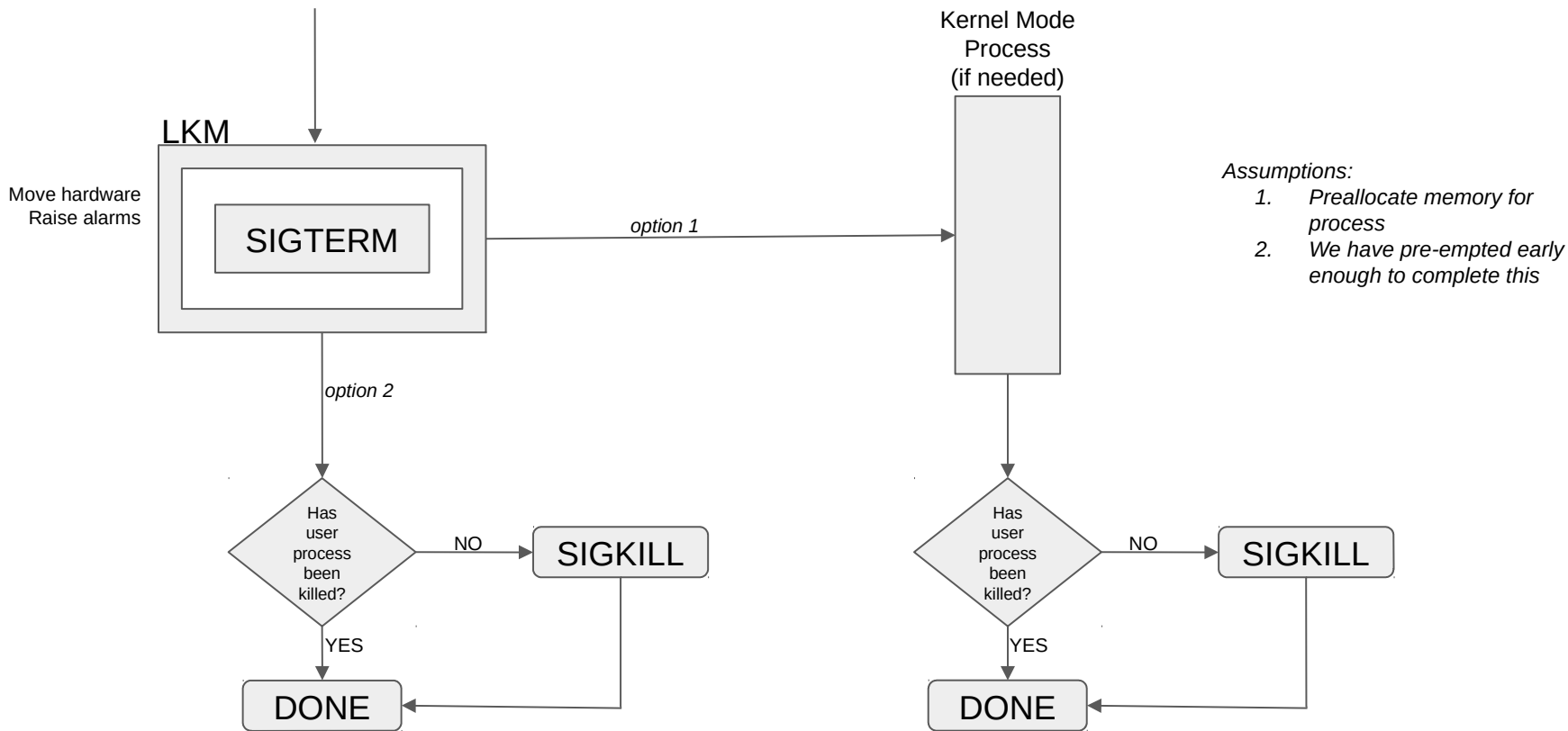
### PRO

- Premarked as “unkillable”
- Access to kernel level and user space info

### CON

- Slower to create / manage
- Likely slower than kernel execution

# Graceful Shutdown - Potential Implementations



Questions?