Surviving the Out of Memory Killer

Dave Hansen & Balbir Singh

OOF Condition

- **Airlines discovered that it was cheaper to fly planes with less fuel on board since it is heavy. Sometimes, they calculated wrong and and the plane would crash. The "fix" was a special OOF (out-of-fuel) mechanism. In emergencies, passengers could be ejected to save weight.**
- **How do we choose the right passenger?**
  - Randomly? Heaviest? Oldest? Cheapest seats? Should we let passengers buy ejection-exempt fares so the poor or cheap ones go?
  - What if the *pilot* is the heaviest or oldest?

thanks to Andries Brouwer

Out of Memory

- **From the kernel's perspective:**
  - "Someone asked for memory and I'm not making any progress helping"
  - We fell under `min_free_kbytes`, scanned memory 6 times, and have not been able to get back above the limit
- **... so we are now going to start killing things**
- **The YKWTLOMFTLAYPHTD Killer lacks the ring of "OOM Killer"**
  - (The Kernel Was Too Low On Memory For Too Long And Your Process Had To Die Killer)

Keeping Score

- **Good News**
  - You have been running for a long time
  - You are root (really CAP_SYS_ADMIN|RAWIO)
- **Bad News**
  - You are a niced process
  - You use a lot of memory (RSS)
  - Your children use a lot of memory

Common Concerns

- There was collateral damage – it killed the “wrong” thing
- It should have never triggered
- It should have triggered faster
- It should have triggered slower

Out of Memory Killer

- **How do you know when it strikes?**
- **Normal causes:**
  - All the memory/swap really is gone
    - Leaks in kernel or userspace?
  - I/O is too slow to swap or write out*
  - The kernel let too much get dirty*
  - Too little memory is reclaimable*
  - The kernel is being stupid
- **Not necessarily indicative of a bug... anywhere**

User Perspectives

- **High Performance Computing**
  - I will take as much memory can be given
  - P.S. Please tell me how much memory that is
  - P.S.S. Swapping is the devil
- **Enterprise (App/DB/Web servers)**
  - Applications do their own memory management
  - If the system gets low on memory, I want the kernel to tell me, and I'll give some of mine back
- **Desktop**
  - When OpenOffice/Firefox blows up, please just kill it quickly, I'll reopen it in a minute
  - P.S. Please don't kill sshd

Memory Reclaim

- **The Linux Philosophy:**
  - ➢ A free page of RAM is a wasted page of RAM
  - ➢ Implication: you will always eventually fill up memory with disk caches
- **Being out of memory is normal!**
- **No free memory? Scan the least-recently-used list (LRU):**
  1) Scan each page in memory (oldest first)
  2) Find users... make them unuse
  3) GOTO 1

Reclaim Speedbumps

- **Pages that can not be reclaimed**
  - Dirty pages, or `malloc()` with no swap
  - `mlock()`, `shm`, slab, `task_struct`
- **Best page to reclaim is a needle in a haystack**
  - 1991 – i386, 16 MHz, 4MB RAM, 4k pages
    - 1,024 pages to scan
  - 2009 – x86_64, 2 GHz, 4GB RAM, 4k pages
    - 1,048,576 pages to scan
- **The reclaim job continues to get harder**
- **If too many speedbumps stop progress -- OOM**

Beat the LRU into shape

- **Never run out of memory, never reclaim, never look at the LRU**
- **Keep troublesome pages off the LRU lists**
  - Right decisions get made faster
  - hugetlbfs, split LRU (~2.6.28)
- **Mitigate other LRU speed bumps**
  - Tune dirty_bytes sysctl
- **Split up the LRU lists**
  - Each NUMA node has its own LRU list(s)
  - Use NUMA machines and kernels or fakenuma=

If you can't beat 'em...

join 'em and make your own LRU

cgroups

- **Kernel-enforced task grouping**
  - “cpusets on steroids”
  - Task grouping specified from userspace
- **Easy-to-develop “controllers”**
  - Care only about cgroups – not individual tasks

cgroups

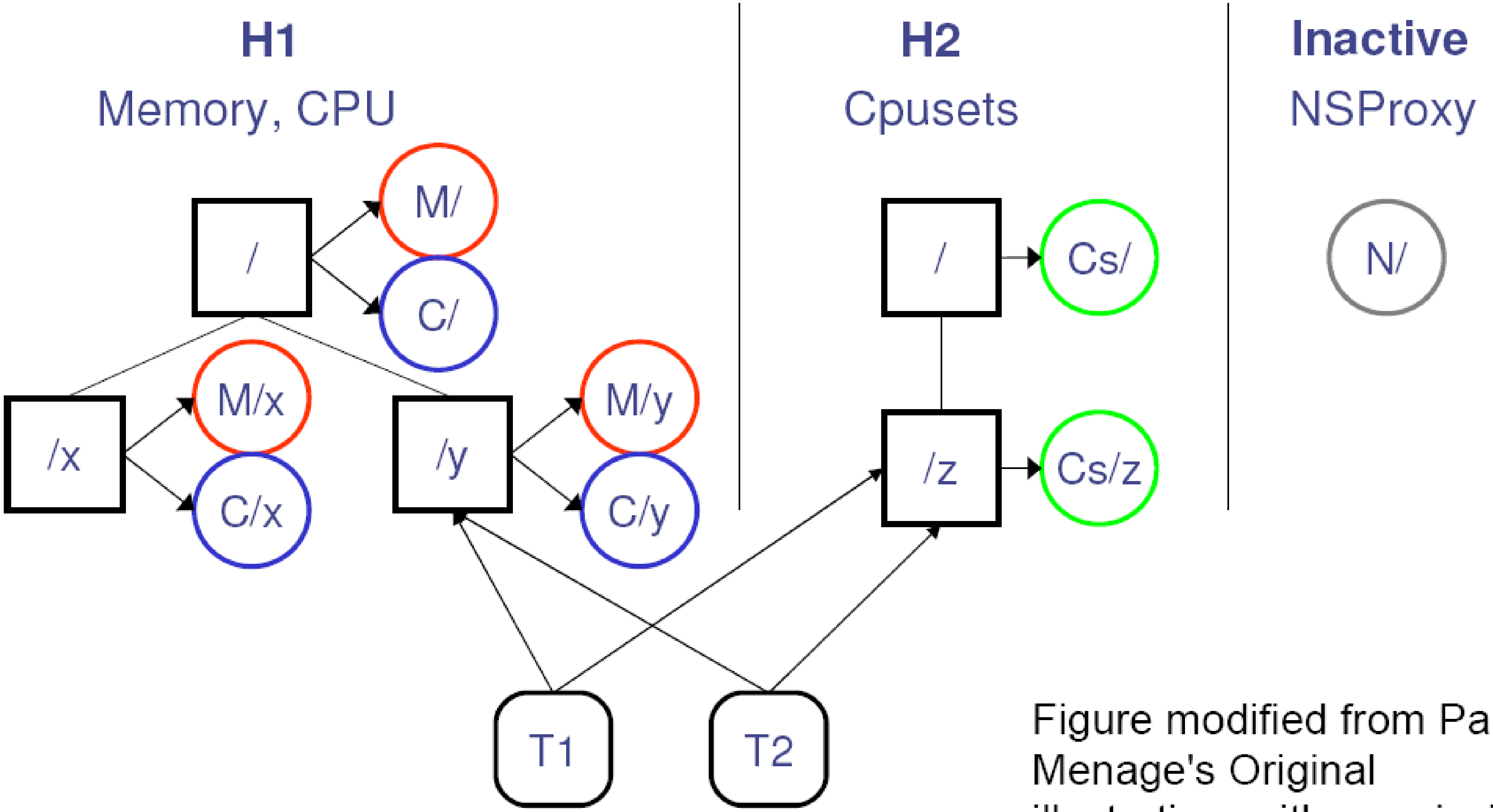


Figure modified from Paul Menage's Original illustration, with permission

Memory Controller

- **Built on top of cgroups**
- **Private LRU per cgroup**
- **Uses**
  - Enforce fairness, but allow workload flexibility
  - Contain memory hogs
  - Segregate sensitive processes
  - Containers
- **Tracks RSS, page cache, swap cache**
- **Enforces limits on memory and swap usage**
- **Individual groups can OOM**

Memory Controller

- **Conventional wisdom**
  - When the system is OOM, it is in real trouble
  - Last thing we want to do is ask userspace either what to kill or to get its help
- **Per-cgroup OOMs change all that**
  - OOM is no longer global – healthy apps can help
- **Kernel can take action against cgroups rather than individual tasks**
  - Kill whole cgroup
  - Reduce cgroup resources

Memory Controller

- **Requires extra accounting**
  - Effectively bloats **`struct page`**, or
  - Accounting costs extra CPU overhead
- **Requires unusual setup above and beyond a normal system**
- **Does not limit kernel memory use**
  - dcache, inode cache, task struct, etc...

Userspace OOM Control

- **Requirement comes from “The Enterprise”**
- **JVM, App/DB/Web Server, workload managers**
  - All do their own memory management
  - Not reflected in kernel's LRU
  - `madvise()` not finely grained-enough
- **Kernels are dumb, applications are smart**
  - Apps are a better position to enforce policies
  - Kernel has no idea about SLAs, etc...

Other Helpful Features

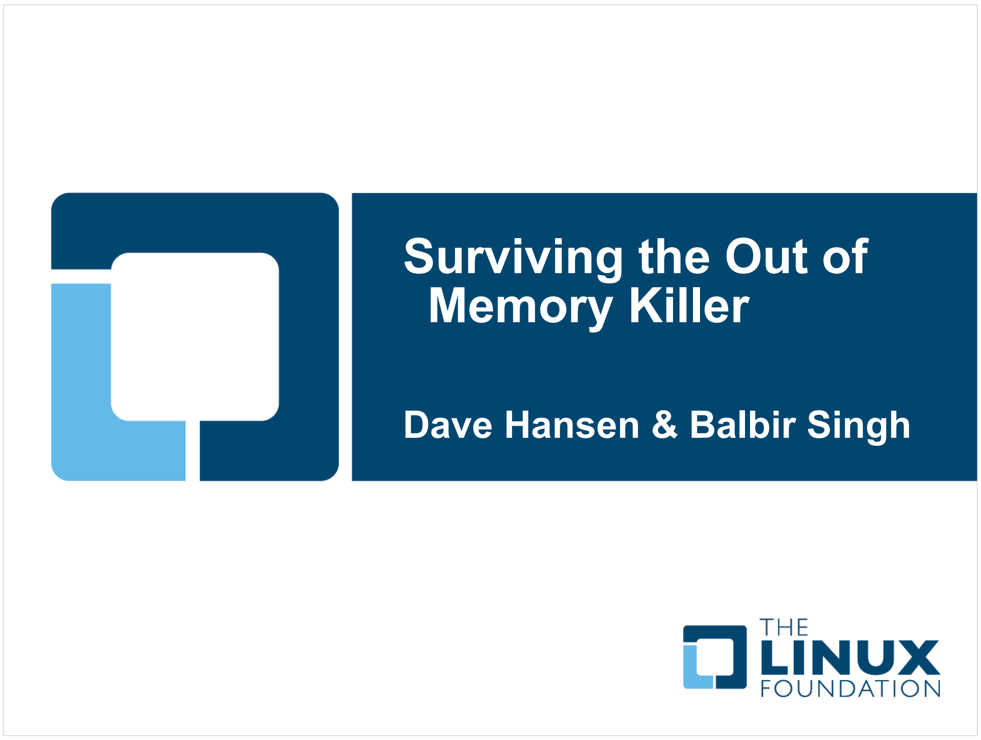
- **kernelcore= (2.6.23)**
  - Specifies ceiling on kernel memory for “non-movable allocations”
  - Inherently controls what the memory controller can not
- **oom_adj / oom_score**
  - Documented ~2.6.18, around longer than that
  - -17 adjustment “disables” OOM for a task
  - Can reduce collateral damage
  - Does not currently exist at cgroup level

Help Needed

- Who has their own OOM code?
- Does using cgroups help having OOMs?
- Does `oom_adj` reduce collateral damage?
- Is swap control effective in preserving consistent application performance?
- Can applications help the kernel during OOM?
- Are any new statistics needed to help applications make OOM decisions?
- What kinds of notifications are preferred?

Further reading

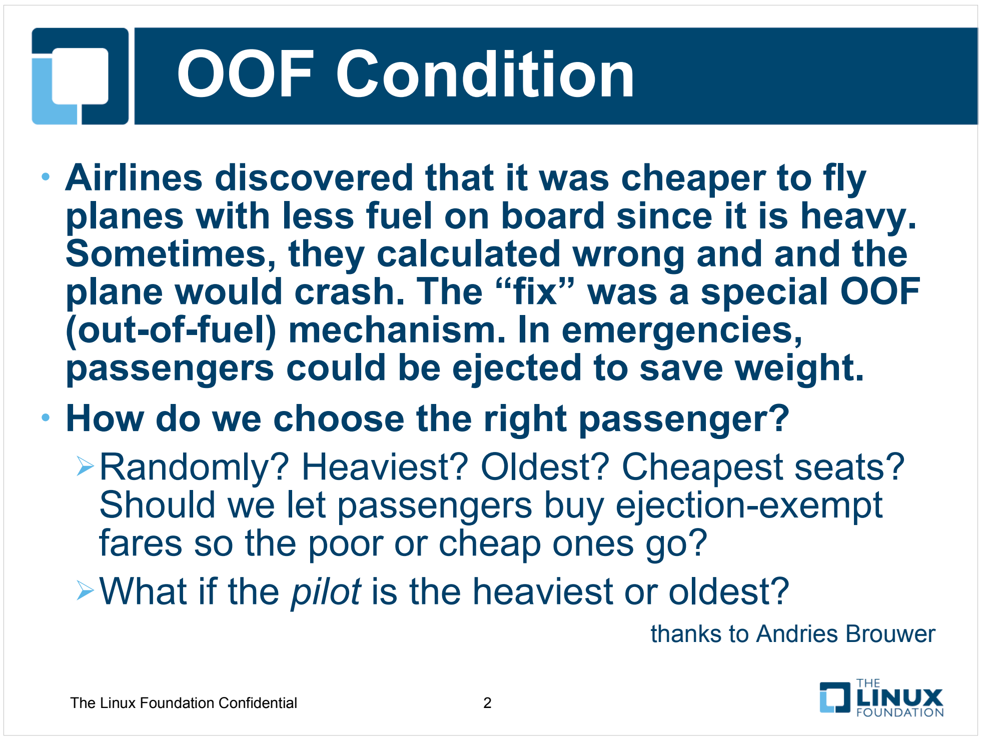
- http://linux-mm.org/OOM
- Documentation/cgroups.txt



Surviving the Out of Memory Killer

Dave Hansen & Balbir Singh

THE LINUX FOUNDATION

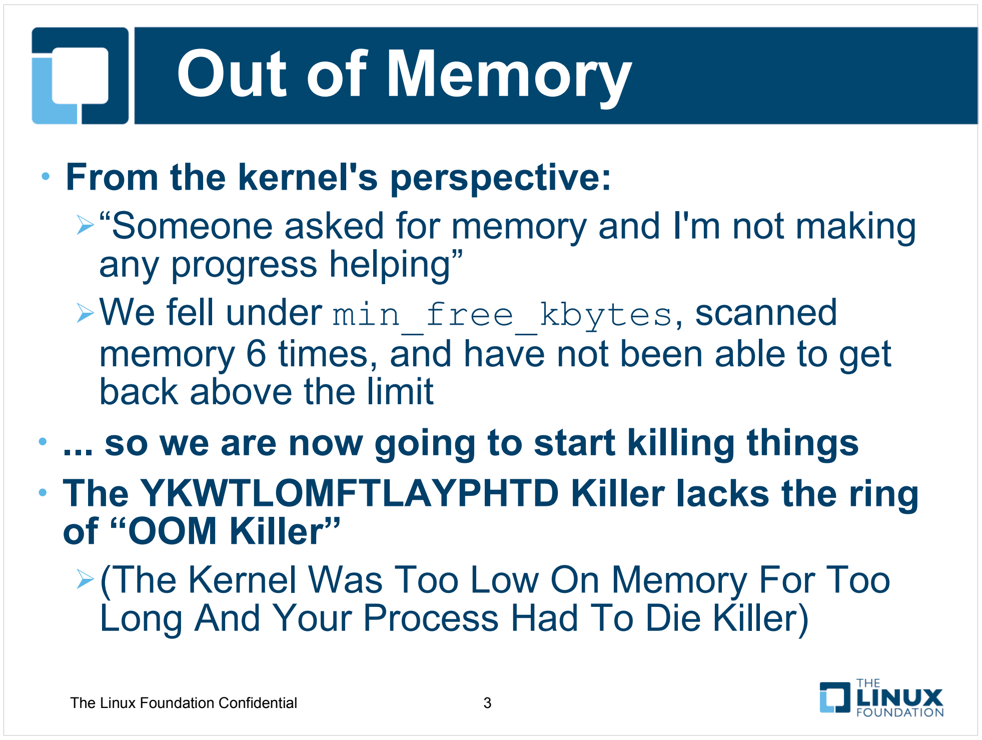


OOF Condition

- **Airlines discovered that it was cheaper to fly planes with less fuel on board since it is heavy. Sometimes, they calculated wrong and and the plane would crash. The “fix” was a special OOF (out-of-fuel) mechanism. In emergencies, passengers could be ejected to save weight.**
- **How do we choose the right passenger?**
  - Randomly? Heaviest? Oldest? Cheapest seats? Should we let passengers buy ejection-exempt fares so the poor or cheap ones go?
  - What if the *pilot* is the heaviest or oldest?

thanks to Andries Brouwer

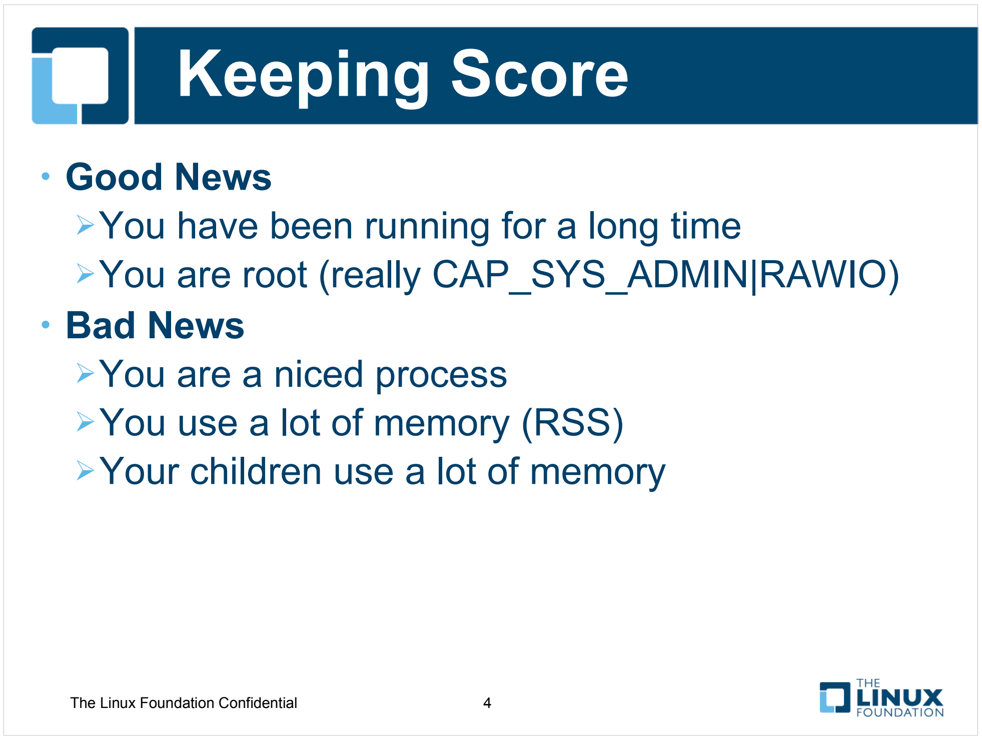
struct page: 32-byte object



Out of Memory

- **From the kernel's perspective:**
  - “Someone asked for memory and I'm not making any progress helping”
  - We fell under `min_free_kbytes`, scanned memory 6 times, and have not been able to get back above the limit
- **... so we are now going to start killing things**
- **The YKWTLOMFTLAYPHTD Killer lacks the ring of “OOM Killer”**
  - (The Kernel Was Too Low On Memory For Too Long And Your Process Had To Die Killer)

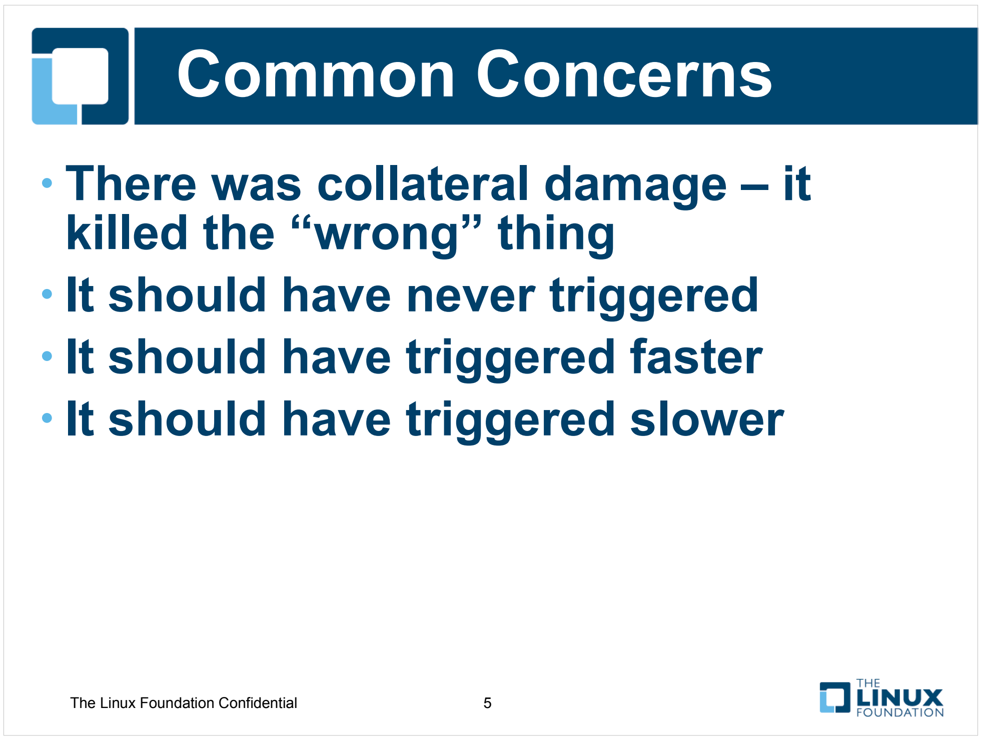
struct page: 32-byte object



Keeping Score

- **Good News**
  - You have been running for a long time
  - You are root (really CAP_SYS_ADMIN|RAWIO)
- **Bad News**
  - You are a niced process
  - You use a lot of memory (RSS)
  - Your children use a lot of memory

struct page: 32-byte object

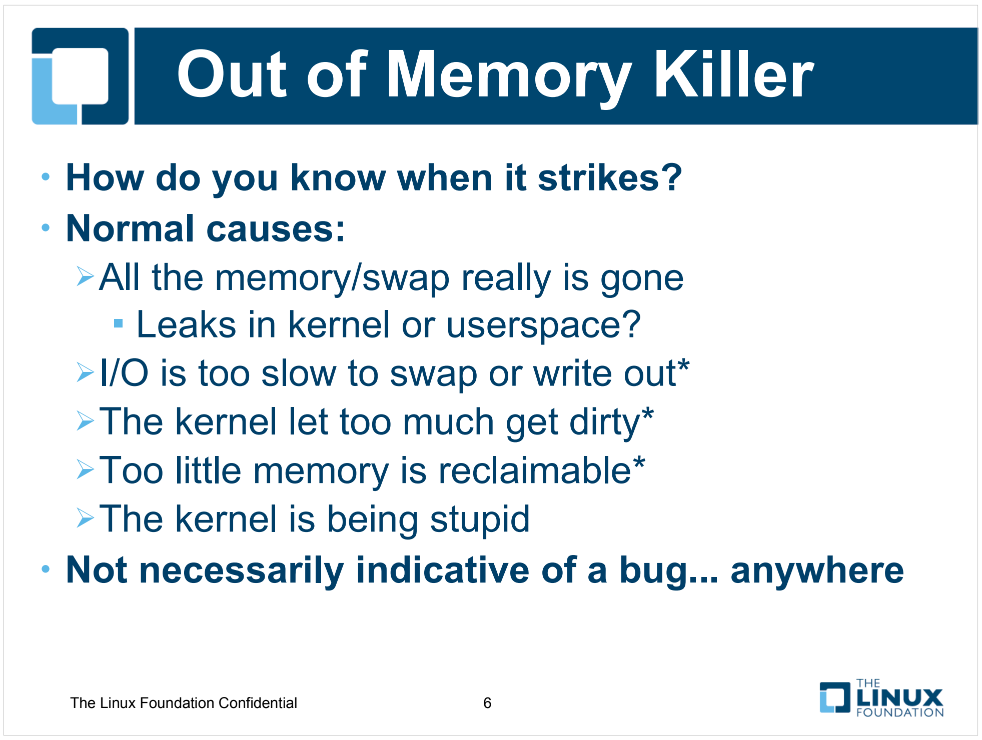


Common Concerns

- There was collateral damage – it killed the “wrong” thing
- It should have never triggered
- It should have triggered faster
- It should have triggered slower

The Linux Foundation Confidential 5

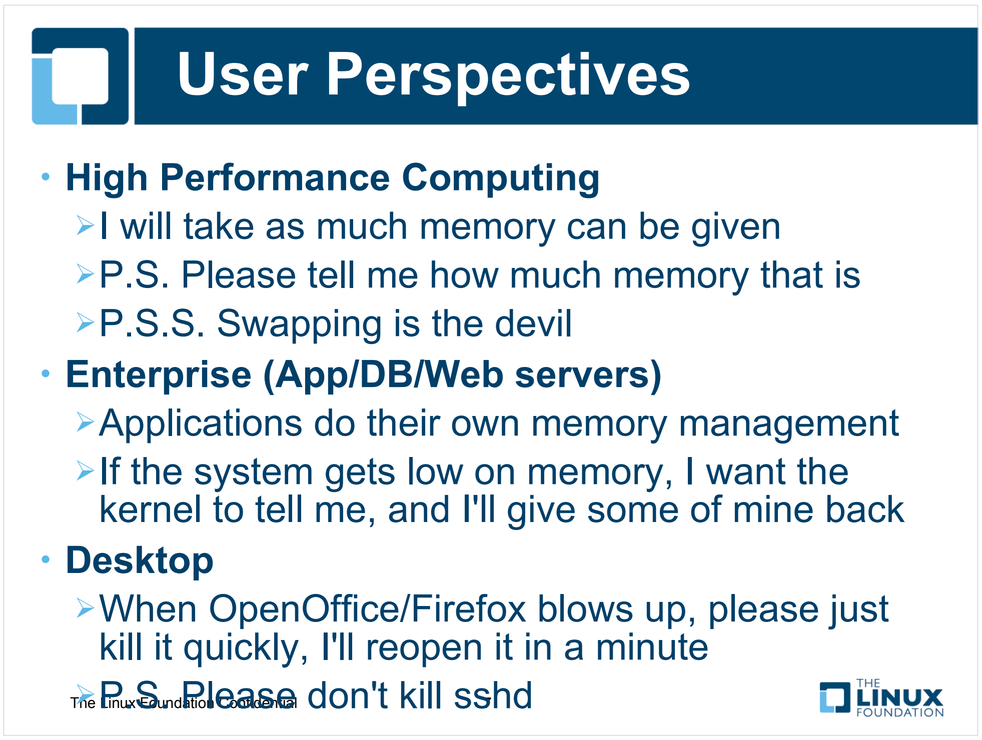
struct page: 32-byte object



Out of Memory Killer

- **How do you know when it strikes?**
- **Normal causes:**
  - All the memory/swap really is gone
    - Leaks in kernel or userspace?
  - I/O is too slow to swap or write out*
  - The kernel let too much get dirty*
  - Too little memory is reclaimable*
  - The kernel is being stupid
- **Not necessarily indicative of a bug... anywhere**

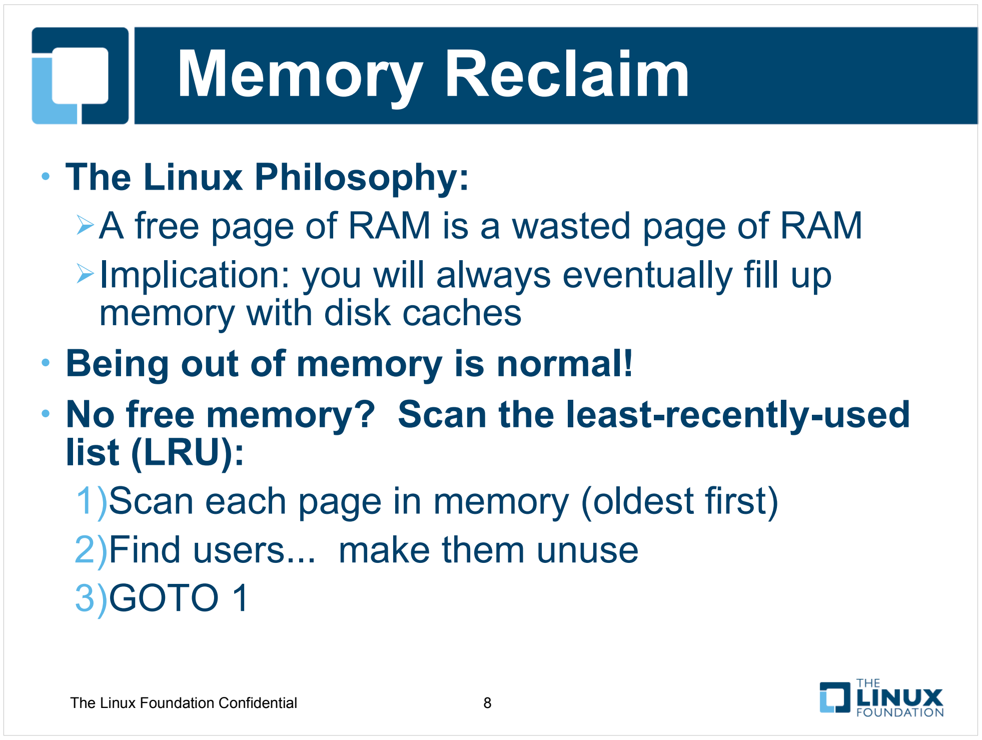
struct page: 32-byte object



User Perspectives

- **High Performance Computing**
  - I will take as much memory can be given
  - P.S. Please tell me how much memory that is
  - P.S.S. Swapping is the devil
- **Enterprise (App/DB/Web servers)**
  - Applications do their own memory management
  - If the system gets low on memory, I want the kernel to tell me, and I'll give some of mine back
- **Desktop**
  - When OpenOffice/Firefox blows up, please just kill it quickly, I'll reopen it in a minute
  - P.S. Please don't kill sshd

struct page: 32-byte object

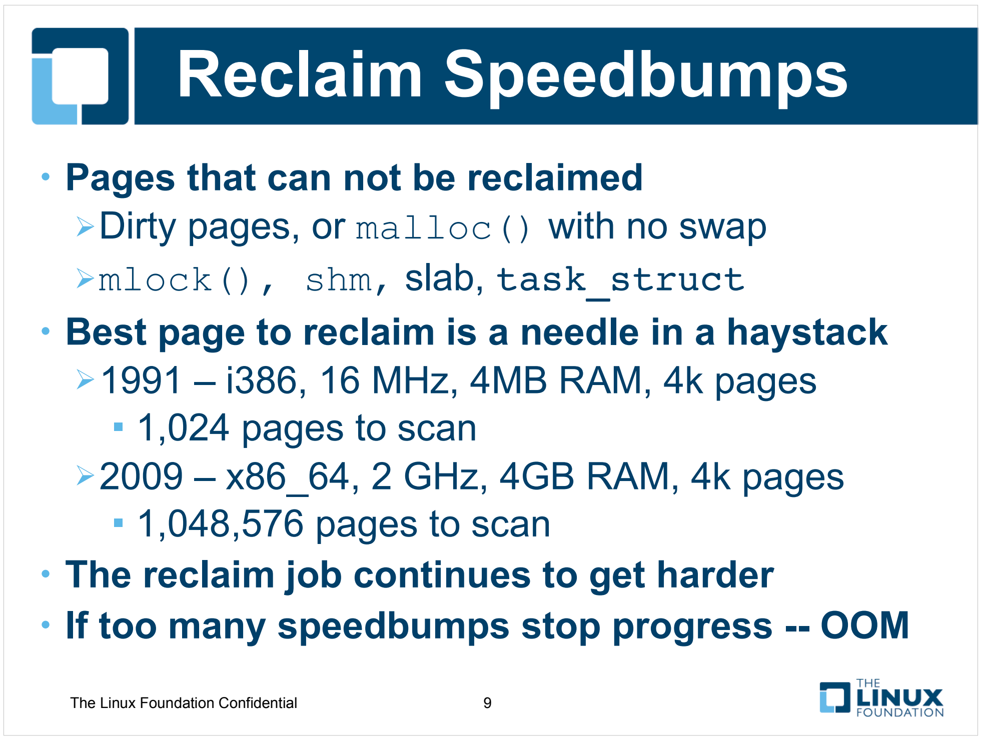


Memory Reclaim

- **The Linux Philosophy:**
  - A free page of RAM is a wasted page of RAM
  - Implication: you will always eventually fill up memory with disk caches
- **Being out of memory is normal!**
- **No free memory? Scan the least-recently-used list (LRU):**
  1) Scan each page in memory (oldest first)
  2) Find users... make them unuse
  3) GOTO 1

The Linux Foundation Confidential 8

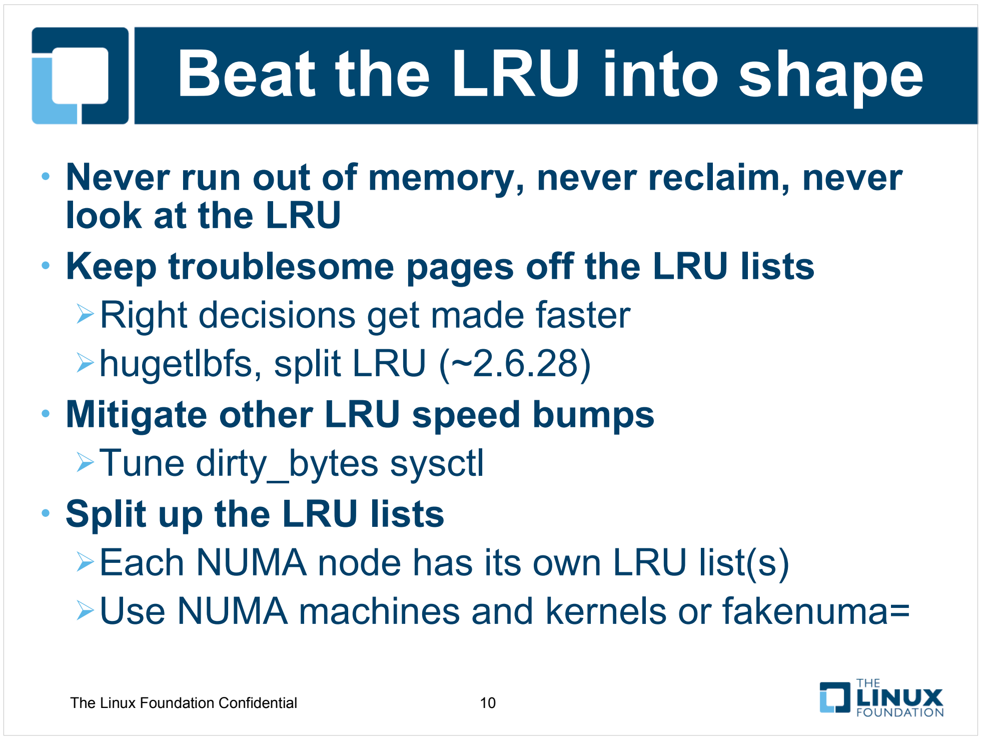
struct page: 32-byte object



Reclaim Speedbumps

- **Pages that can not be reclaimed**
  - Dirty pages, or `malloc()` with no swap
  - `mlock()`, `shm`, slab, `task_struct`
- **Best page to reclaim is a needle in a haystack**
  - 1991 – i386, 16 MHz, 4MB RAM, 4k pages
    - 1,024 pages to scan
  - 2009 – x86_64, 2 GHz, 4GB RAM, 4k pages
    - 1,048,576 pages to scan
- **The reclaim job continues to get harder**
- **If too many speedbumps stop progress -- OOM**

struct page: 32-byte object


Beat the LRU into shape

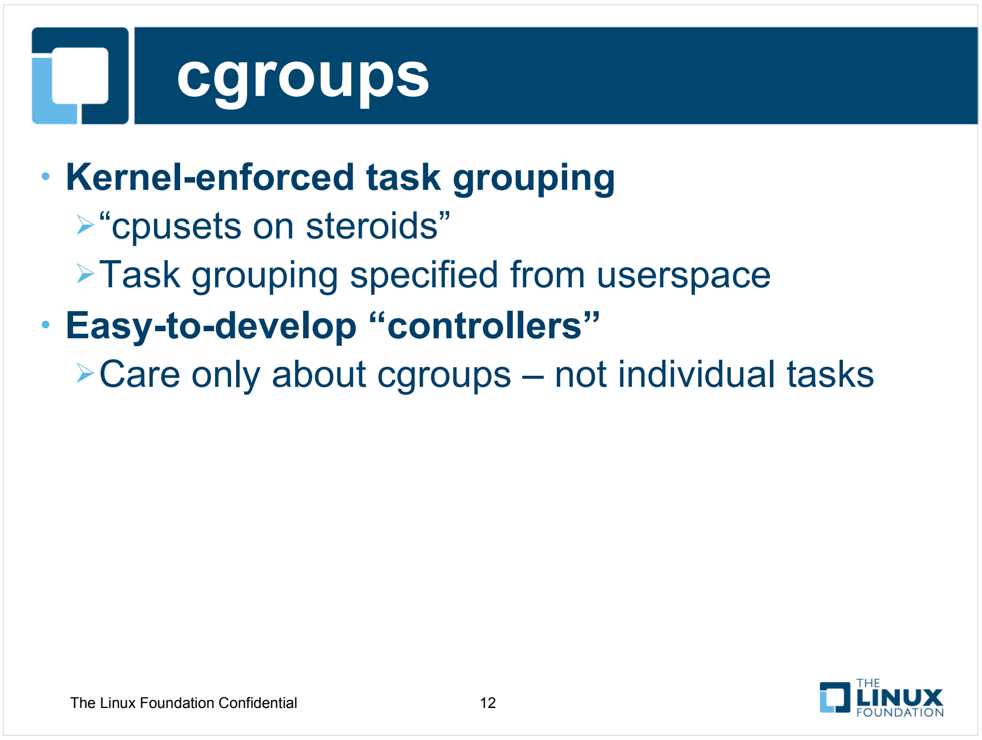
- **Never run out of memory, never reclaim, never look at the LRU**
- **Keep troublesome pages off the LRU lists**
  - Right decisions get made faster
  - hugetlbfs, split LRU (~2.6.28)
- **Mitigate other LRU speed bumps**
  - Tune dirty_bytes sysctl
- **Split up the LRU lists**
  - Each NUMA node has its own LRU list(s)
  - Use NUMA machines and kernels or fakenuma=

struct page: 32-byte object

If you can't beat 'em...

join 'em and make your own LRU

struct page: 32-byte object



cgroups

- **Kernel-enforced task grouping**
  - “cpusets on steroids”
  - Task grouping specified from userspace
- **Easy-to-develop “controllers”**
  - Care only about cgroups – not individual tasks

struct page: 32-byte object

cgroups

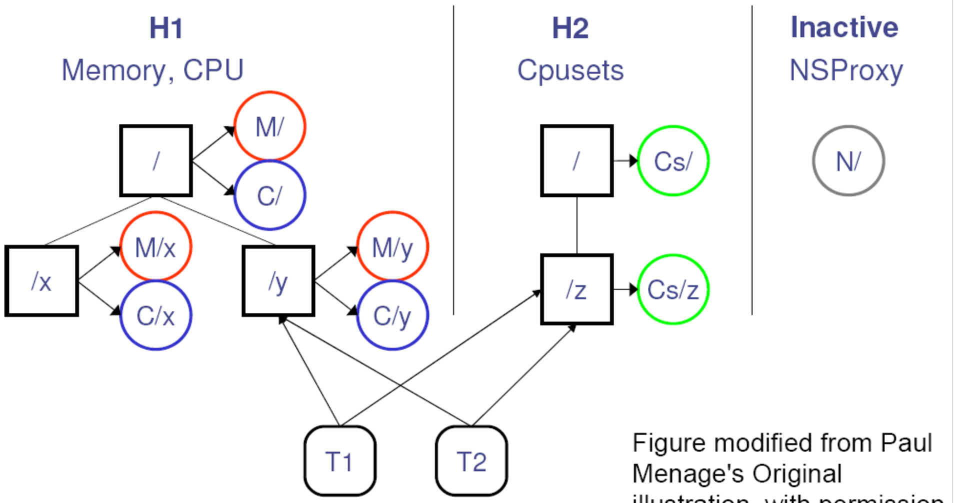
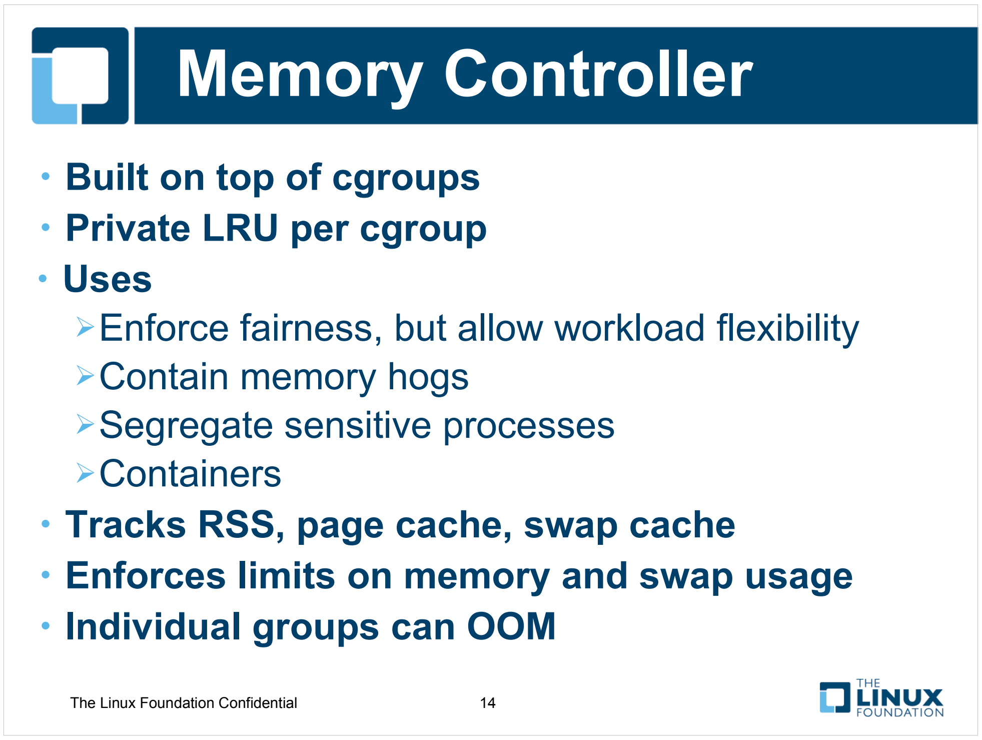


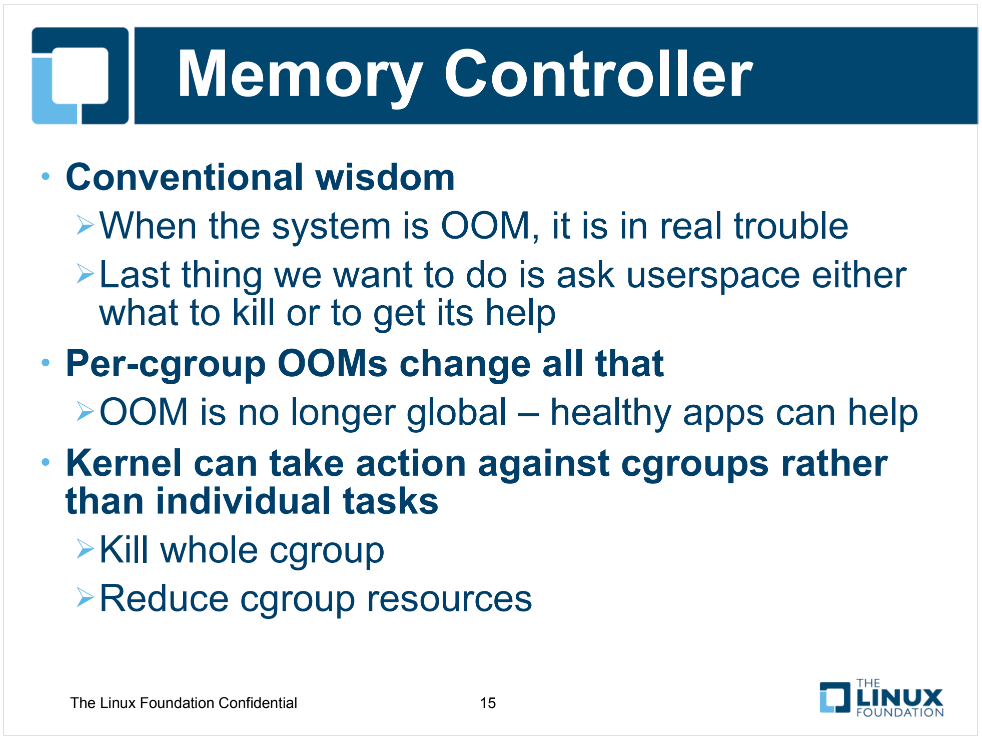
Figure modified from Paul Menage's Original illustration, with permission



Memory Controller

- **Built on top of cgroups**
- **Private LRU per cgroup**
- **Uses**
  - Enforce fairness, but allow workload flexibility
  - Contain memory hogs
  - Segregate sensitive processes
  - Containers
- **Tracks RSS, page cache, swap cache**
- **Enforces limits on memory and swap usage**
- **Individual groups can OOM**

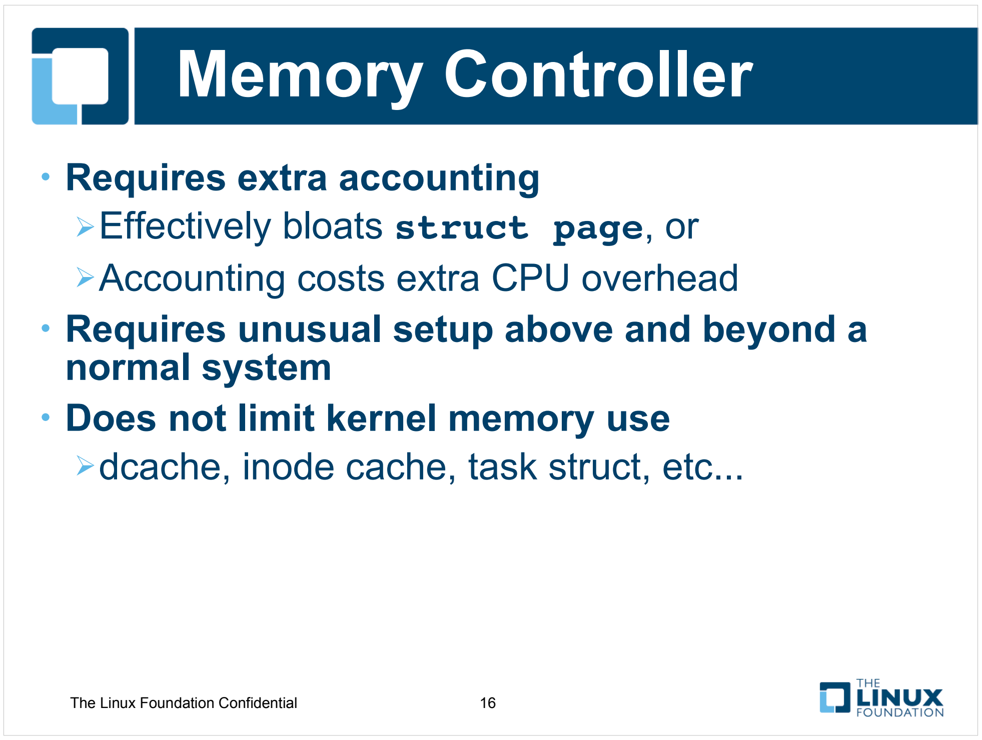
struct page: 32-byte object



Memory Controller

- **Conventional wisdom**
  - When the system is OOM, it is in real trouble
  - Last thing we want to do is ask userspace either what to kill or to get its help
- **Per-cgroup OOMs change all that**
  - OOM is no longer global – healthy apps can help
- **Kernel can take action against cgroups rather than individual tasks**
  - Kill whole cgroup
  - Reduce cgroup resources

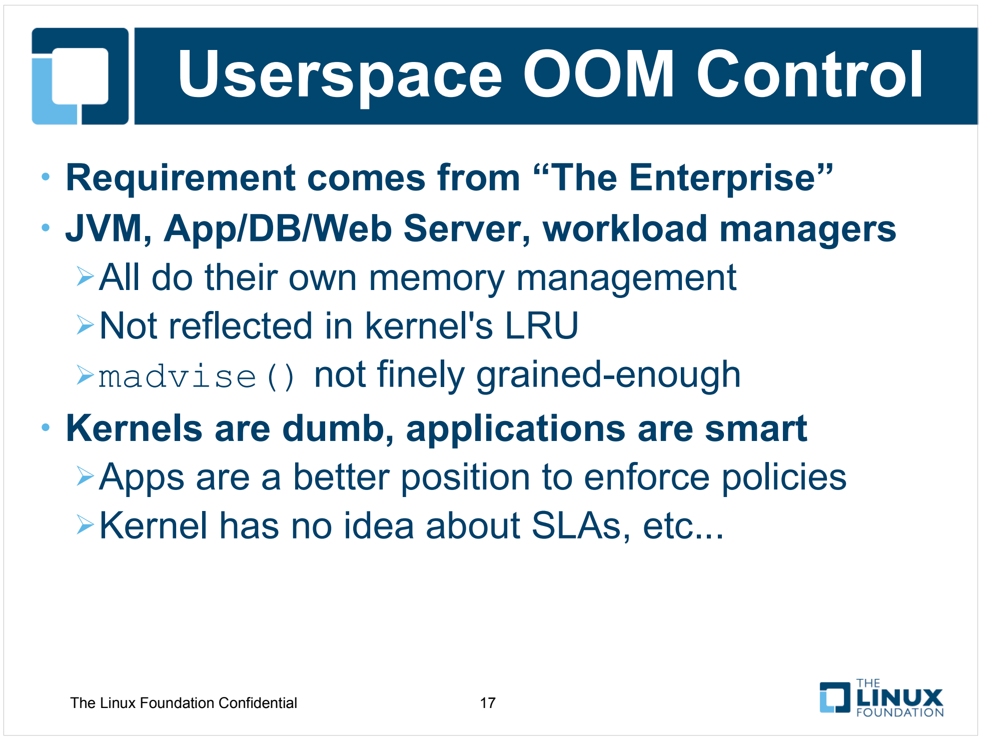
struct page: 32-byte object



Memory Controller

- **Requires extra accounting**
  - Effectively bloats **`struct page`**, or
  - Accounting costs extra CPU overhead
- **Requires unusual setup above and beyond a normal system**
- **Does not limit kernel memory use**
  - dcache, inode cache, task struct, etc...

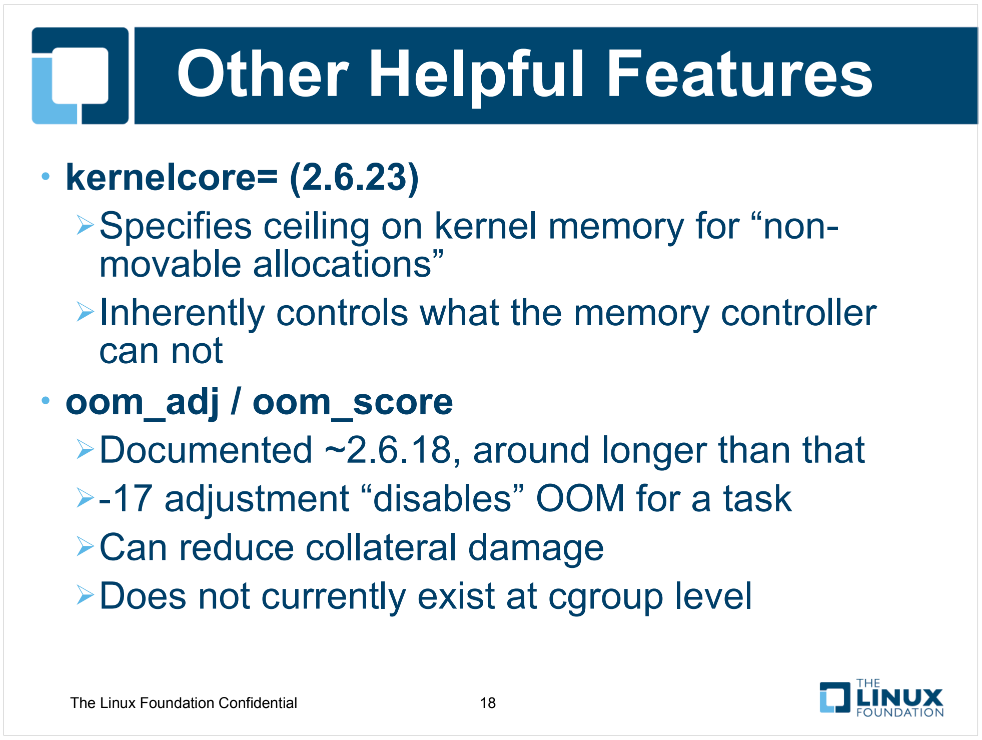
struct page: 32-byte object



Userspace OOM Control

- **Requirement comes from "The Enterprise"**
- **JVM, App/DB/Web Server, workload managers**
  - All do their own memory management
  - Not reflected in kernel's LRU
  - `madvise()` not finely grained-enough
- **Kernels are dumb, applications are smart**
  - Apps are a better position to enforce policies
  - Kernel has no idea about SLAs, etc...

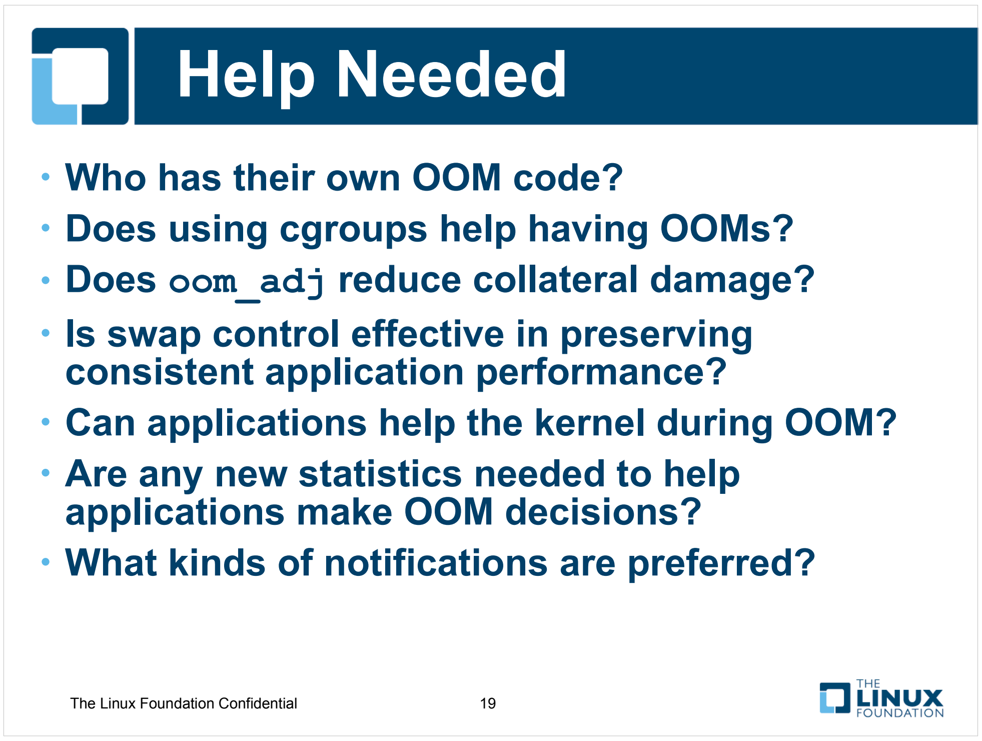
struct page: 32-byte object



Other Helpful Features

- **kernelcore= (2.6.23)**
  - Specifies ceiling on kernel memory for "non-movable allocations"
  - Inherently controls what the memory controller can not
- **oom_adj / oom_score**
  - Documented ~2.6.18, around longer than that
  - -17 adjustment "disables" OOM for a task
  - Can reduce collateral damage
  - Does not currently exist at cgroup level

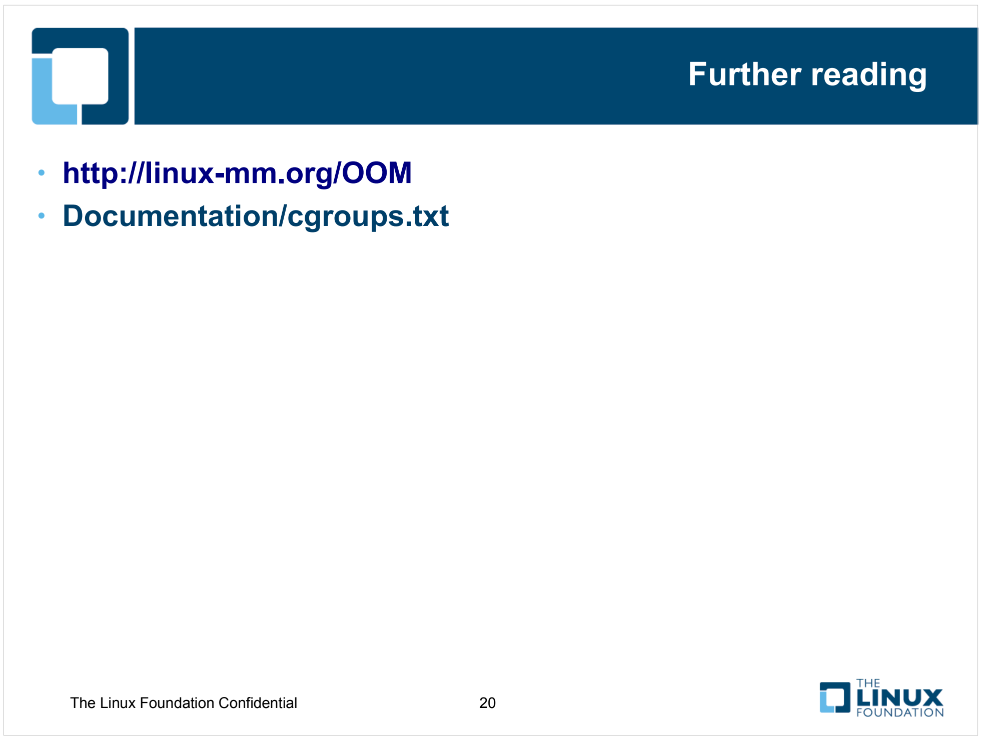
struct page: 32-byte object



Help Needed

- Who has their own OOM code?
- Does using cgroups help having OOMs?
- Does `oom_adj` reduce collateral damage?
- Is swap control effective in preserving consistent application performance?
- Can applications help the kernel during OOM?
- Are any new statistics needed to help applications make OOM decisions?
- What kinds of notifications are preferred?

struct page: 32-byte object



Further reading

- http://linux-mm.org/OOM
- Documentation/cgroups.txt

struct page: 32-byte object