Resource Management

Dave Hansen
IBM Linux Technology Center

THE LINUX FOUNDATION

Homework

- https://fedoraproject.org/wiki/Features/ControlGroups

Resource Management

- **Long-standing feature request**
  - CKRM, Beancounters, others...
- **Single OS instance, multiple uses**
  - Departments sharing a DB server
  - Containers
  - Linux as the hypervisor
- **Datacenter-level management**
  - Checkpoint/restart

Requirements

- Group arbitrary processes
  - Processes able to move between groups
  - Kernel->Webserver->DB->Disk
- Easy to add new subsystems
- Definable containment
- Low overhead
- Flexible userspace API
- Arbitrary numbers of groups

cgroups

- **Got in through the back door**
  - cooped existing cpusets interfaces
  - cpusets became one *subsystem*
- **“task-oriented”**
  - associates a set of tasks with a set of parameters for one or more subsystems
- **Subsystems contain “controllers”**
- **Linux-y interfaces: mount, echo, chmod**

cgroup terminology

- **cgroups** associate tasks with subsystems
  - example: “power users”
- **subsystems** utilize cgroups to treat grouped tasks in a common way
  - example: “memory subsystem”
- **hierarchies** provide relationships between cgroups (think inheritance)
  - tasks have 1 position in each

cgroups

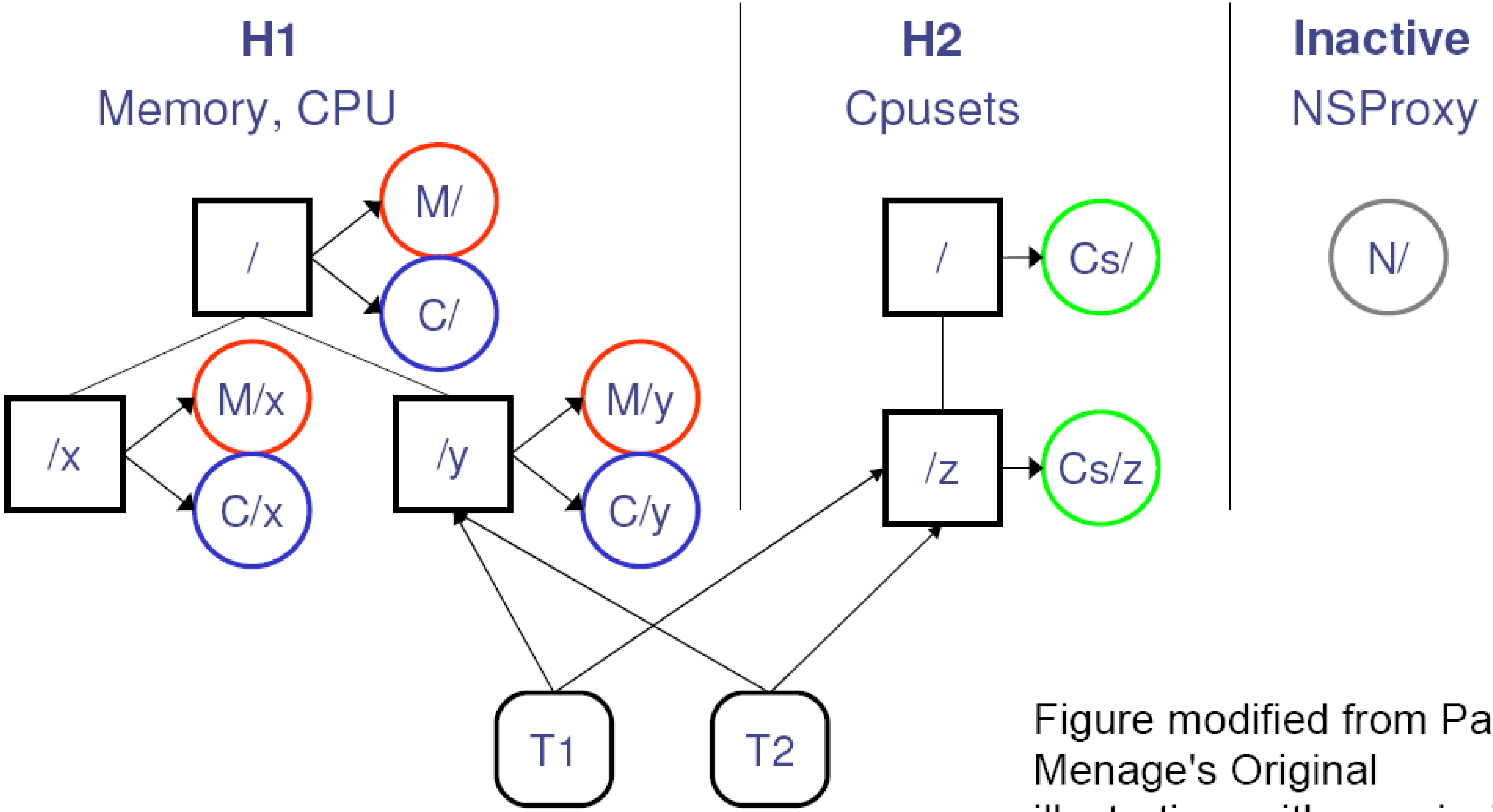


Figure modified from Paul Menage's Original illustration, with permission

CPU Controller

- **Separate from CPUSets**
- **CFS (2.6.23 process scheduler)**
  - People contributed to with cgroups in mind
  - Provides framework for CPU-based control
- **“Share” model**
  - more users mean smaller shares
- **Hierarchies are supported**
  - Users can subdivide their share
- **Also: CPU Accounting subsystem**
  - Accounting-only: no control

Memory Controller

- **Barriers to acceptance:**
  - Performance/space overhead
  - Impact to the core VM
- **Each page has an “owner” cgroup**
  - Assigned at allocation time
- **Limits placed on ownership quantity**
- **Swap controller implemented**
- **Per-group swappiness**
- **RSS control**
- **cgroups can be out-of-memory targeted**

context switch...

Out of Memory

- **“Someone asked for memory and I'm not making any progress helping”**
- **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

Memory Reclaim

- Scan each page on the LRU
- Find users... make them unuse
- Rinse, repeat...
- HPC? All mlocked()
- Progress?

Solutions?

- **Split LRU (2.6.28)**
  - Ignore mlock() during reclaim
- **kernelcore= (2.6.23)**
  - Specifies ceiling on kernel memory for “non-movable allocations”
- **oom_adj / oom_score**
  - Documented ~2.6.18, around for a while
  - -17 adjustment “disables” OOM
- **User jobs in a memory cgroup**
- **Large pages**
  - Great talk in next hour!

</oom>

libcgroup

- **Kernel interface is via ram-based fs**
  - Not user friendly
- **Abstraction**
  - 'mv' is not a user-acceptable interface
- **Persistence**
  - /etc/sysctl.conf vs. /proc/sys
- **Automatic Classification**

Checkpoint/Restart

- **Resource management not limited to a single system**
- **cgroups keeps different users in line**
- **What when users outgrow a cgroup?**
- **Many existing solutions**
  - Zap, OpenVZ, IBM Metacluster, blcr
  - All out of tree – bad for customers
- **Goals: Reliability, Flexibility**

Expected Users

- **OpenVZ-like virtual private servers**
- **Datacenter workload balancing**
- **Live kernel upgrades**
- **Clusters**
  - Job management
  - Debugging

Checkpoint/Restart

- **Step 1: Isolate**
  - cgroups / containers
  - Namespaces: pid, uts, net, fs, ipc...
  - physical resources (MAC, IP, etc...)
- **Step 2: Serialize**
  - pick up those isolated objects
  - write to disk or send across network

Issues

- Filesystem state
  - rsync?
  - btrfs helps
- Infiniband
- New kernel features must be continually supported
- Must not slow down other kernel development

Community

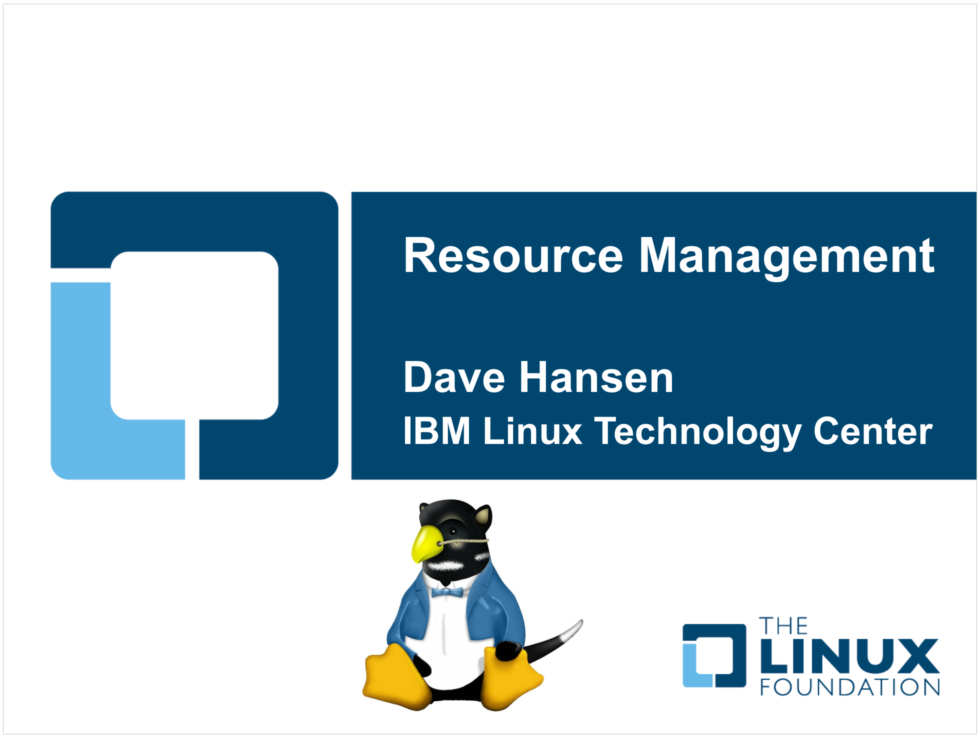
- Participating: OpenVZ, IBM, Zap...
- Goal: same feature set as existing out-of-tree implementations
- Rebuilding from scratch
  - Goals: simple, small, well-factored
- Oren Laadan (of Zap) maintaining
  - Pursuing -mm inclusion
- Alexey Dobriyan has another set

Current Feature Set

- **Architectures: x86, x86_64, ppc, s390**
- **Single and multiple process support**
- **Self and external checkpoint**
- **“Simple” open files, pipes**
- **Shared memory (shmfs)**
- **Efficiently handles shared objects**
  - Like pipe contents or file position

Credits

- Thanks to Balbir Singh and Dhaval Giani for all the input and updates.
- Thanks to Paul Menage for letting me steal his nice pictures



Resource Management

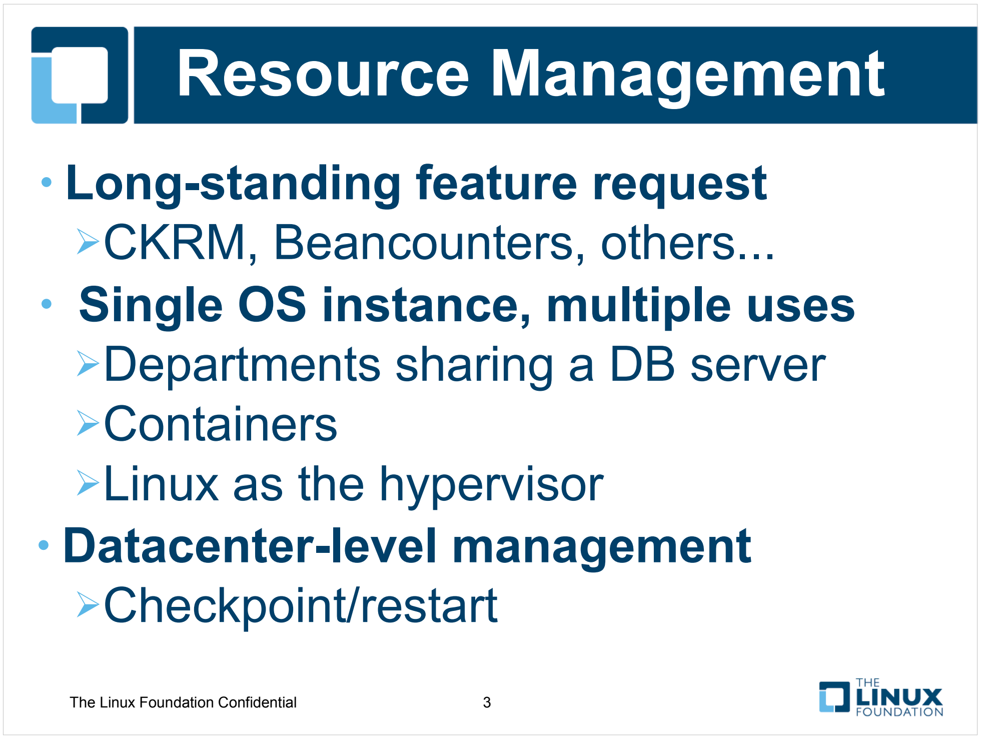
Dave Hansen

IBM Linux Technology Center

THE LINUX FOUNDATION

Homework

- https://fedoraproject.org/wiki/Features/ControlGroups

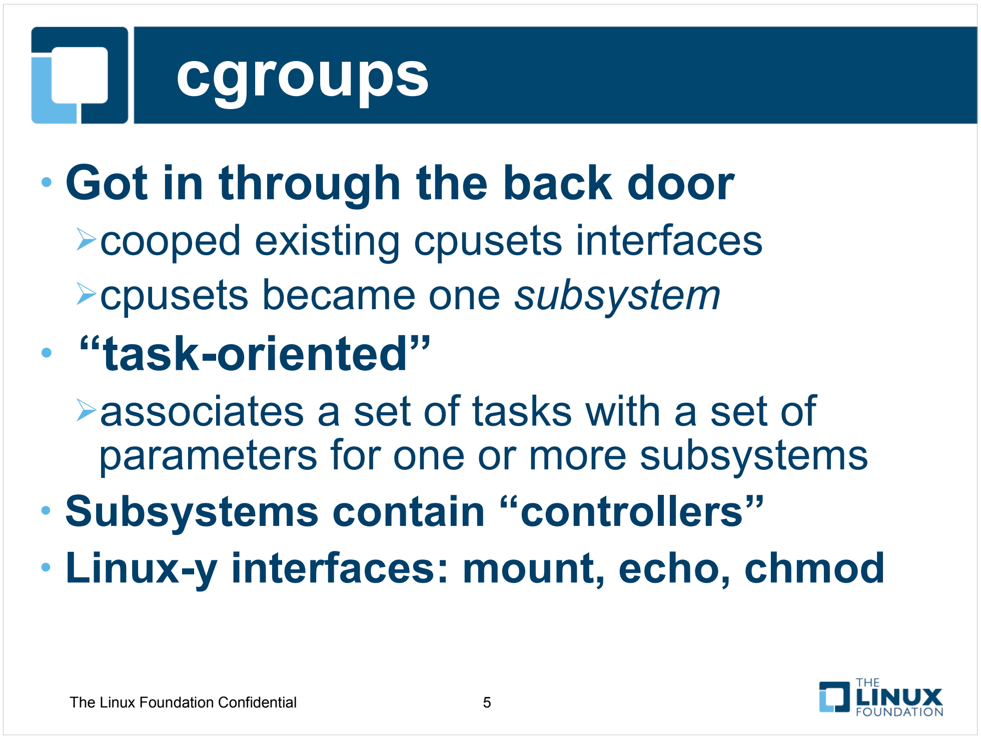


Resource Management

- **Long-standing feature request**
  - CKRM, Beancounters, others...
- **Single OS instance, multiple uses**
  - Departments sharing a DB server
  - Containers
  - Linux as the hypervisor
- **Datacenter-level management**
  - Checkpoint/restart

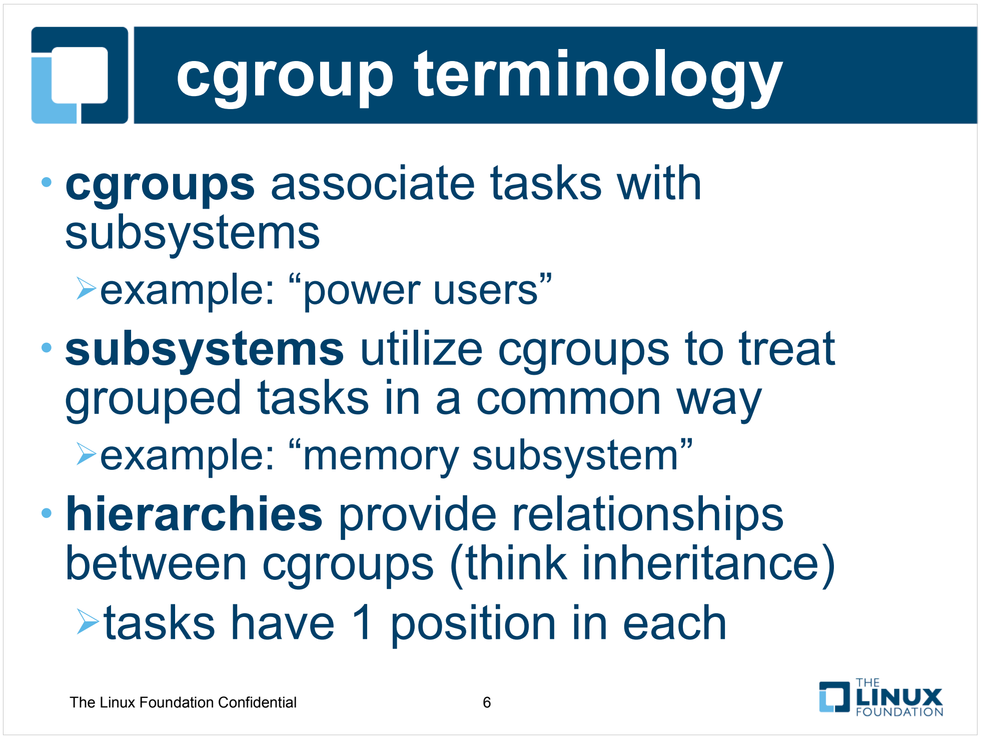
Requirements

- **Group arbitrary processes**
  - Processes able to move between groups
  - Kernel->Webserver->DB->Disk
- **Easy to add new subsystems**
- **Definable containment**
- **Low overhead**
- **Flexible userspace API**
- **Arbitrary numbers of groups**



cgroups

- **Got in through the back door**
  - cooped existing cpusets interfaces
  - cpusets became one *subsystem*
- **“task-oriented”**
  - associates a set of tasks with a set of parameters for one or more subsystems
- **Subsystems contain “controllers”**
- **Linux-y interfaces: mount, echo, chmod**



cgroup terminology

- **cgroups** associate tasks with subsystems
  - ➢example: “power users”
- **subsystems** utilize cgroups to treat grouped tasks in a common way
  - ➢example: “memory subsystem”
- **hierarchies** provide relationships between cgroups (think inheritance)
  - ➢tasks have 1 position in each

cgroups

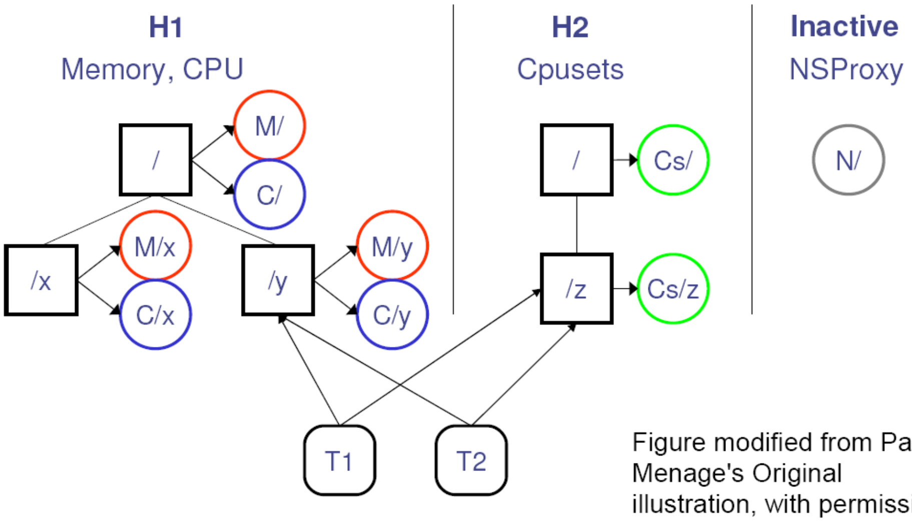
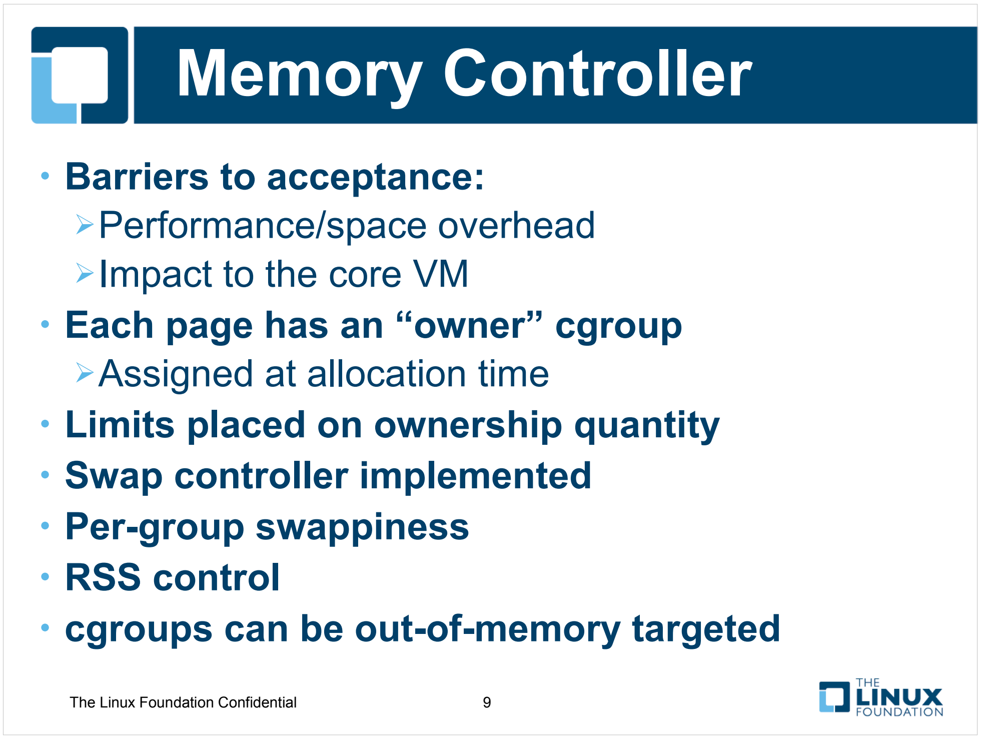


Figure modified from Paul Menage's Original illustration, with permission

CPU Controller

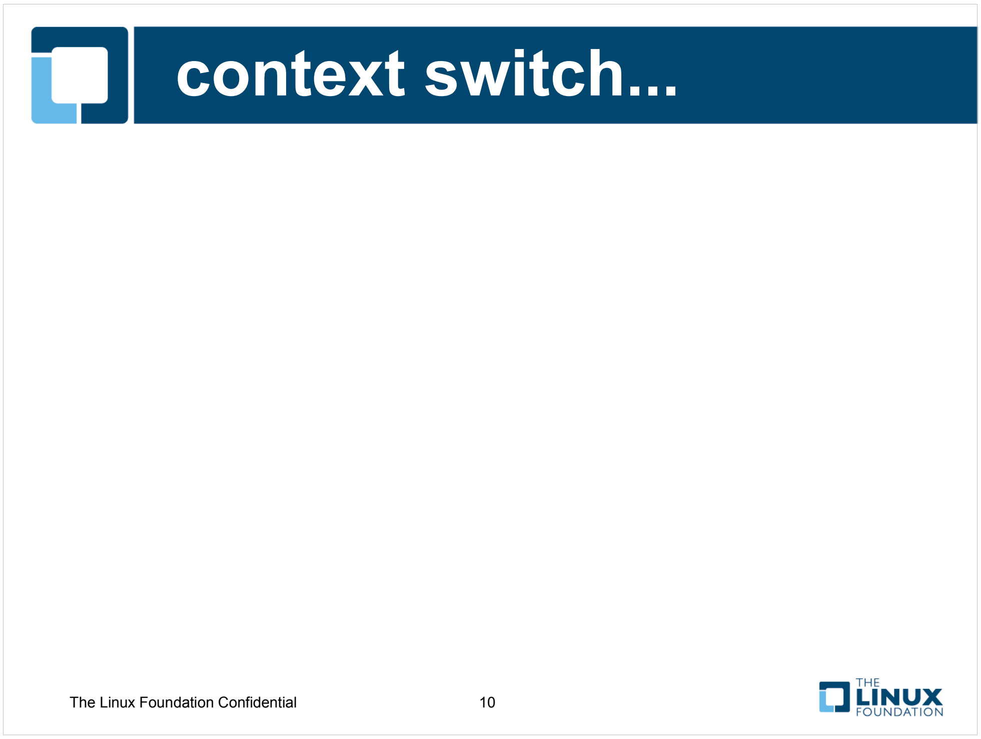
- **Separate from CPUSets**
- **CFS (2.6.23 process scheduler)**
  - People contributed to with cgroups in mind
  - Provides framework for CPU-based control
- **“Share” model**
  - more users mean smaller shares
- **Hierarchies are supported**
  - Users can subdivide their share
- **Also: CPU Accounting subsystem**
  - Accounting-only: no control



Memory Controller

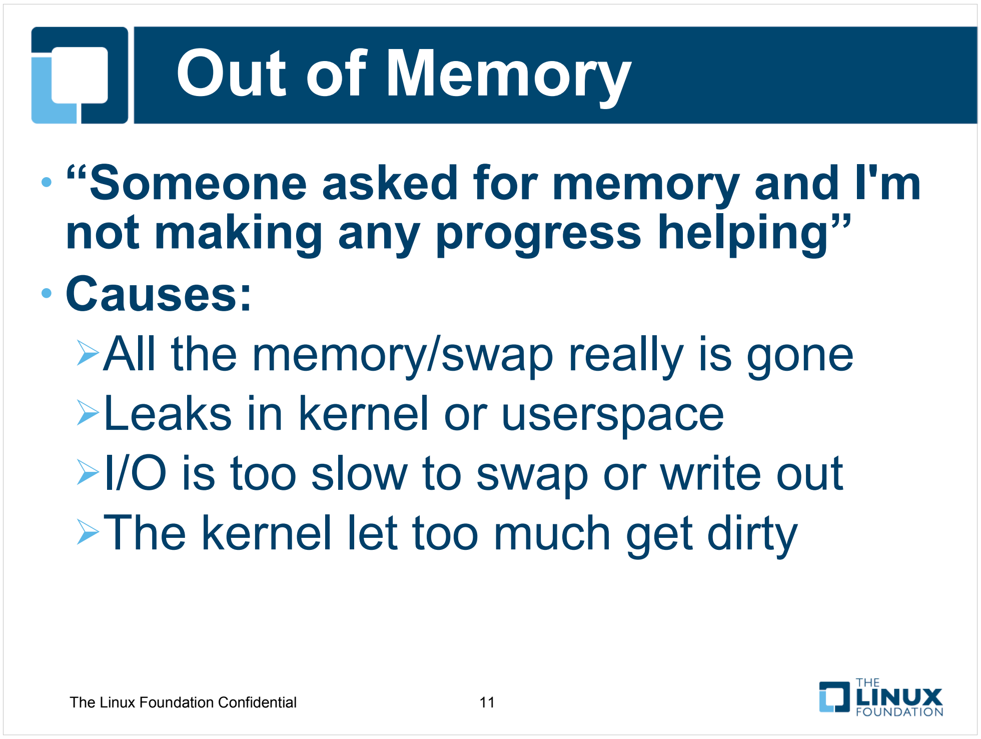
- **Barriers to acceptance:**
  - Performance/space overhead
  - Impact to the core VM
- **Each page has an “owner” cgroup**
  - Assigned at allocation time
- **Limits placed on ownership quantity**
- **Swap controller implemented**
- **Per-group swappiness**
- **RSS control**
- **cgroups can be out-of-memory targeted**

struct page: 32-byte object



context switch...

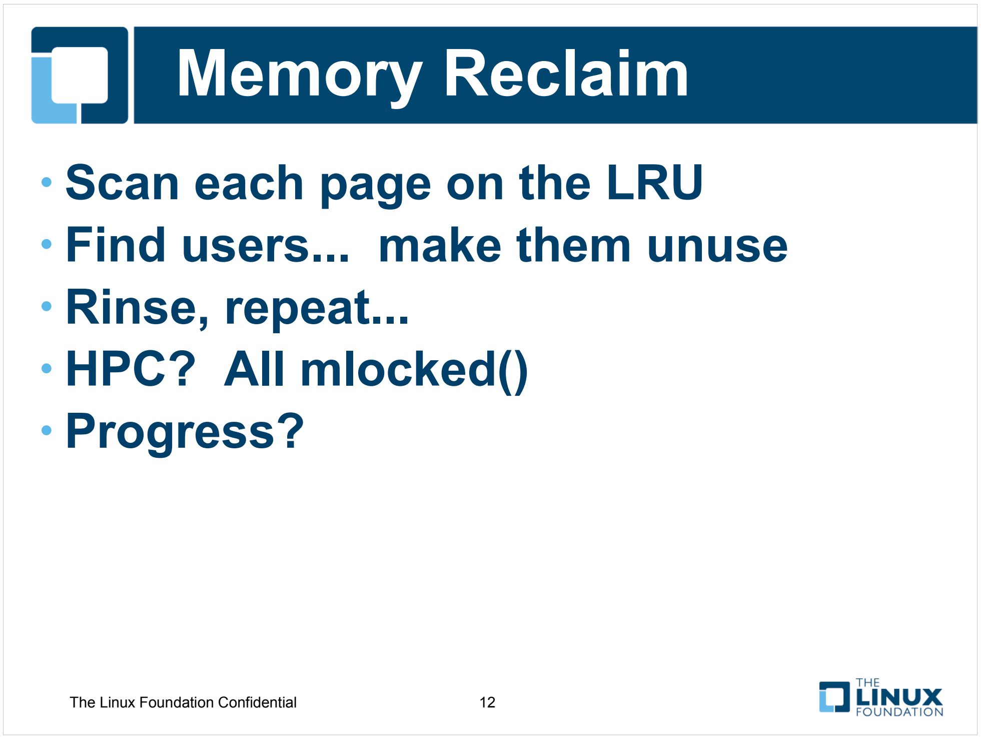
struct page: 32-byte object



Out of Memory

- **“Someone asked for memory and I'm not making any progress helping”**
- **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

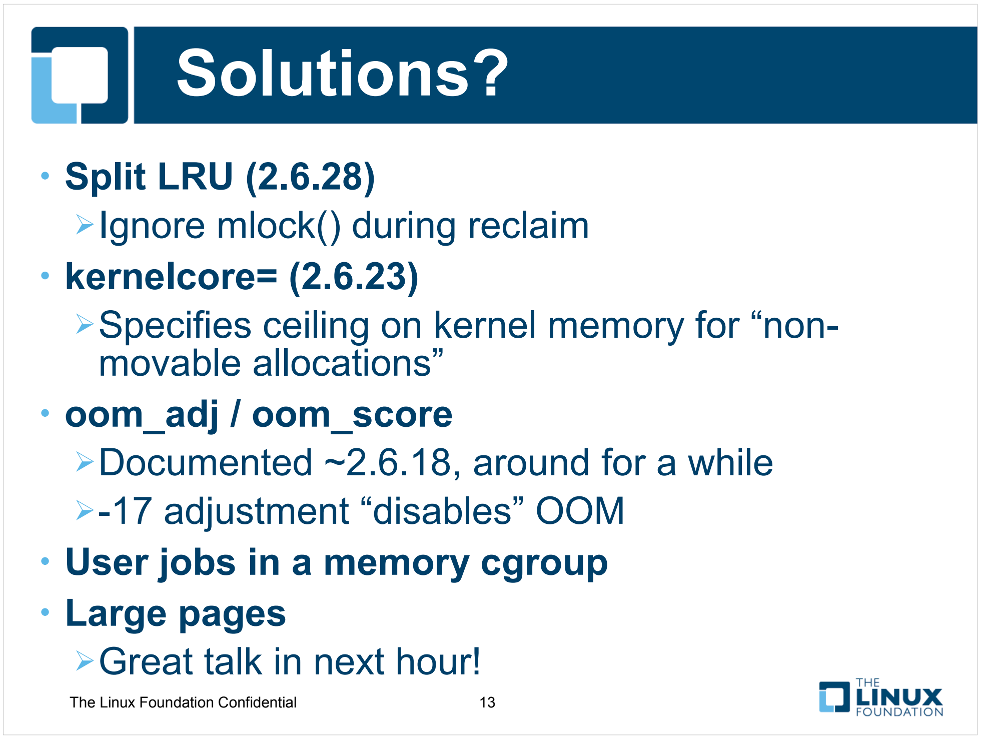
struct page: 32-byte object



Memory Reclaim

- Scan each page on the LRU
- Find users... make them unuse
- Rinse, repeat...
- HPC? All mlocked()
- Progress?

struct page: 32-byte object



Solutions?

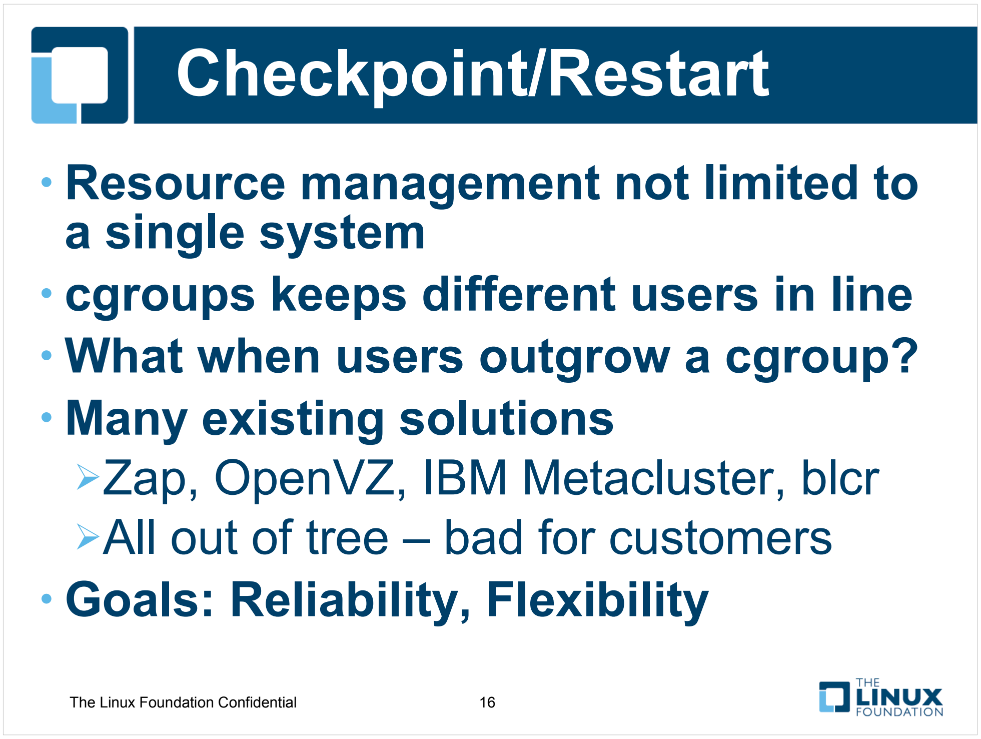
- **Split LRU (2.6.28)**
  - Ignore mlock() during reclaim
- **kernelcore= (2.6.23)**
  - Specifies ceiling on kernel memory for “non-movable allocations”
- **oom_adj / oom_score**
  - Documented ~2.6.18, around for a while
  - -17 adjustment “disables” OOM
- **User jobs in a memory cgroup**
- **Large pages**
  - Great talk in next hour!

struct page: 32-byte object

</oom>

libcgroup

- **Kernel interface is via ram-based fs**
  - Not user friendly
- **Abstraction**
  - 'mv' is not a user-acceptable interface
- **Persistence**
  - /etc/sysctl.conf vs. /proc/sys
- **Automatic Classification**



Checkpoint/Restart

- **Resource management not limited to a single system**
- **cgroups keeps different users in line**
- **What when users outgrow a cgroup?**
- **Many existing solutions**
  - ➢Zap, OpenVZ, IBM Metacluster, blcr
  - ➢All out of tree – bad for customers
- **Goals: Reliability, Flexibility**

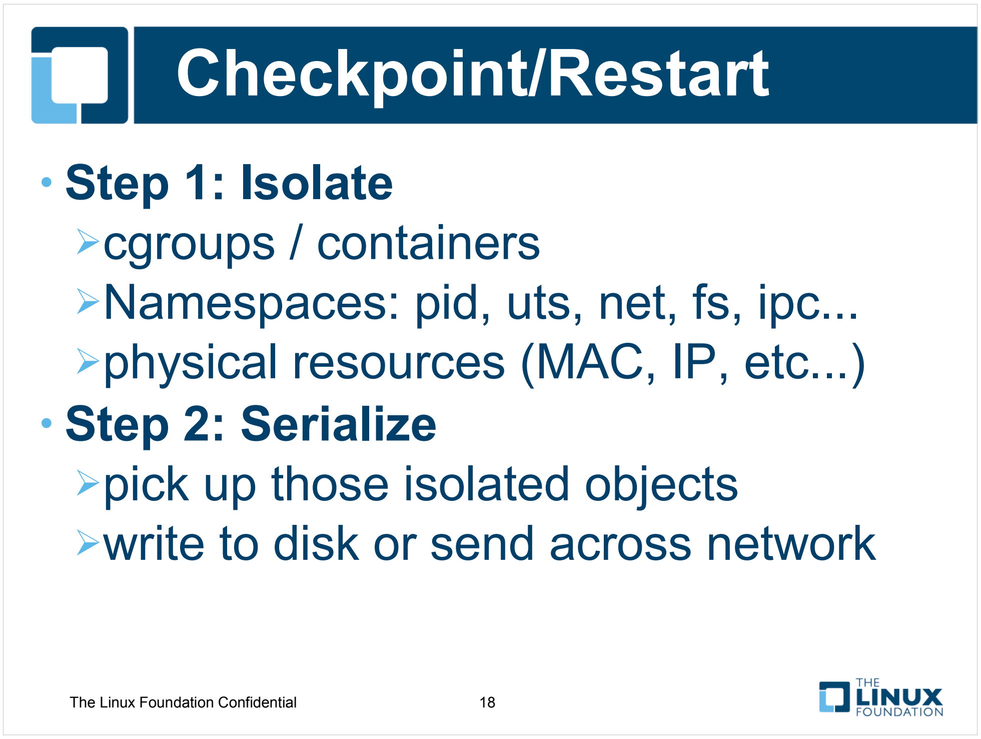
Users:

1. system containers like OpenVZ does
2. workload migration in the datacenter – DB load grew too large to be on the same machine as the web server
3. Live kernel upgrades
4. Clusters: don't want to rewrite that 20-year-old fortran app, but want to be able to save its work

Got a crash? Move it off the cluster for diagnosis

Expected Users

- OpenVZ-like virtual private servers
- Datacenter workload balancing
- Live kernel upgrades
- Clusters
  - Job management
  - Debugging

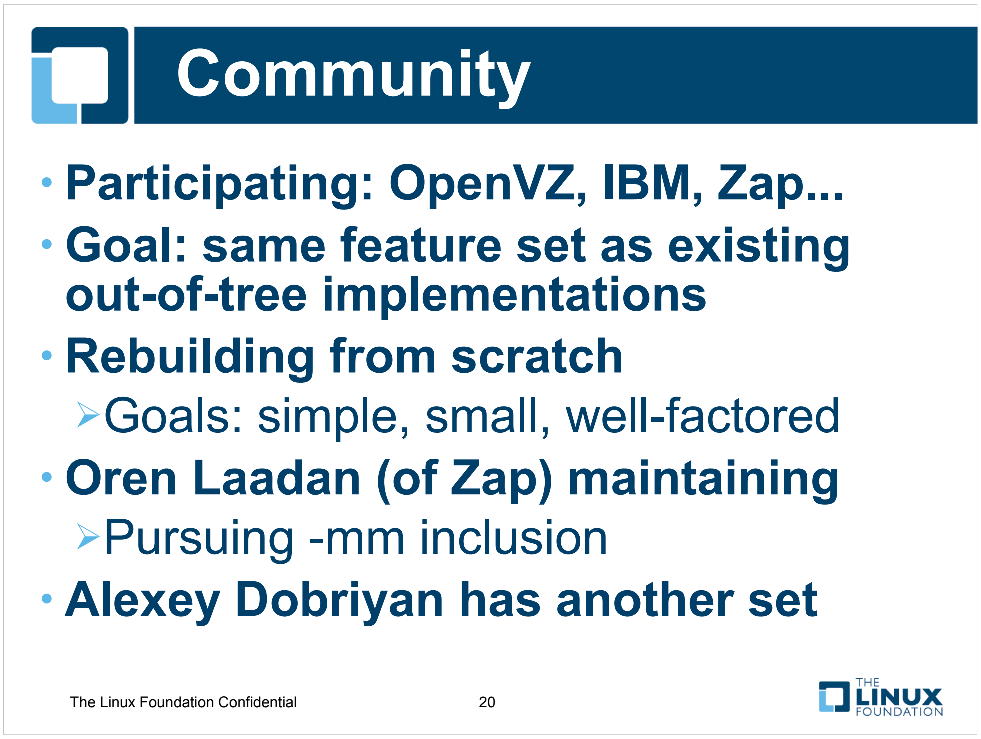


Checkpoint/Restart

- **Step 1: Isolate**
  - cgroups / containers
  - Namespaces: pid, uts, net, fs, ipc...
  - physical resources (MAC, IP, etc...)
- **Step 2: Serialize**
  - pick up those isolated objects
  - write to disk or send across network

Issues

- **Filesystem state**
  - rsync?
  - btrfs helps
- **Infiniband**
- **New kernel features must be continually supported**
- **Must not slow down other kernel development**



Community

- Participating: OpenVZ, IBM, Zap...
- Goal: same feature set as existing out-of-tree implementations
- Rebuilding from scratch
  - ➢Goals: simple, small, well-factored
- Oren Laadan (of Zap) maintaining
  - ➢Pursuing -mm inclusion
- Alexey Dobriyan has another set

mention openvz's demo of a counterstrike server being migrated
or a whole vnc'd x server

Current Feature Set

- Architectures: x86, x86_64, ppc, s390
- Single and multiple process support
- Self and external checkpoint
- “Simple” open files, pipes
- Shared memory (shmfs)
- Efficiently handles shared objects
  - Like pipe contents or file position

Credits

- Thanks to Balbir Singh and Dhaval Giani for all the input and updates.
- Thanks to Paul Menage for letting me steal his nice pictures