# Linuxカーネル開発動向
# Trends in Linux Kernel Development



２００２年５月２９日
アレキザンダー・リーダー
Alexander Reeder
技術部、開発
alexr@valinux.co.jp

VA LINUX SYSTEMS JAPAN

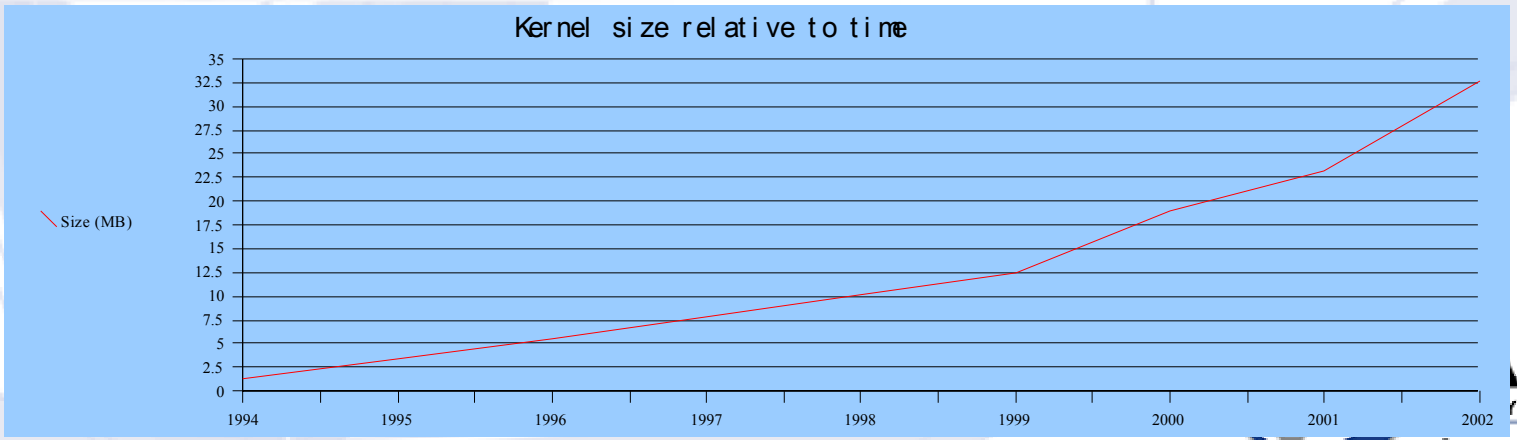
# Agenda

- General overview of development from v1.0 to v2.5
- New developments in v2.5
- Sub-system changes in v2.5
- Overall Trends
- Conclusion

VA LINUX SYSTEMS JAPAN

# Evolution of the Linux Kernel

| | |
|---|---|
| v1.0 | Released 1994.03.13<br>Size 1.2 MB (gzip compression) |
| v2.0 | Released 1996.07.09<br>Size 5.6 MB |
| v2.2 | Released 1999.01.26<br>Size 12.5 MB |
| v2.4 | Released 2001.01.04<br>Size 23.2 MB |
| v2.5.18 | Released 2002.05.24<br>Size 32.7 MB |



# Supported architectures

| | v1.0 | v2.0 | v2.2 | v2.4 | v2.5.18 |
|---|---|---|---|---|---|
| alpha | | o | o | o | o |
| arm | | | | o | o |
| cris | | | | | o |
| i386 | o | o | o | o | o |
| ia64 | | | | o | o |
| m68k | | o | o | o | o |
| mips | | o | o | o | o |
| mips64 | | | | o | o |
| parisc | | | | o | o |
| ppc | | o | o | o | o |
| ppc64 | | | | | o |
| s390 | | | | o | o |
| s390x | | | | | o |
| sh | | | | o | o |
| sparc | | o | o | o | o |
| sparc64 | | | o | o | o |
| x86_64 | | | | | o |
| Totals | 1 | 6 | 7 | 13 | 17 |



VA LINUX SYSTEMS JAPAN

# Supported file systems

| | v1.0 | v2.0 | v2.2 | v2.4 | v2.5.18 |
|---|---|---|---|---|---|
| adfs | | | o | o | o |
| affs | | o | o | o | o |
| autofs | | | o | o | o |
| autofs4 | | | | o | o |
| bfs | | | | o | o |
| coda | | | o | o | o |
| cramfs | | | | o | o |
| efs | | | | o | o |
| exportfs | | | | | o |
| ext | o | o | o | o | o |
| ext2 | o | o | o | o | o |
| ext3 | | | | | o |
| fat | | o | o | o | o |
| freevxfs | | | | | o |
| hfs | | | o | o | o |
| hpfs | o | o | o | o | o |
| intermezzo | | | | | o |
| isofs | o | o | o | o | o |
| jffs | | | | o | o |

| | v1.0 | v2.0 | v2.2 | v2.4 | v2.5.18 |
|---|---|---|---|---|---|
| jffs2 | | | | | o |
| jfs | | | | | o |
| minix | o | o | o | o | o |
| msdos | o | o | o | o | o |
| ncpfs | | o | o | o | o |
| nfs | o | o | o | o | o |
| ntfs | | | o | o | o |
| openpromfs | | | | o | o |
| qnx4 | | | o | o | o |
| ramfs | | | | o | o |
| reiserfs | | | | | o |
| romfs | | | o | o | o |
| smbfs | | | o | o | o |
| sysv | o | o | o | o | o |
| udf | | | | | o |
| ufs | | o | o | o | o |
| umsdos | | o | o | o | o |
| vfat | | o | o | o | o |
| xiafs | o | o | o | o | o |

| Totals | 9 | 16 | 25 | 32 | 40 |
|---|---|---|---|---|---|

# New Developments in v2.5

## Video for Linux redesign

The API is being totally rewritten to be more flexible and extensible, and to support a larger variety of devices. The old API will be left in the kernel for backwards compatibility, but V4L2 (the new API) is not compatible with V4L.

# Initial Support for USB 2.0

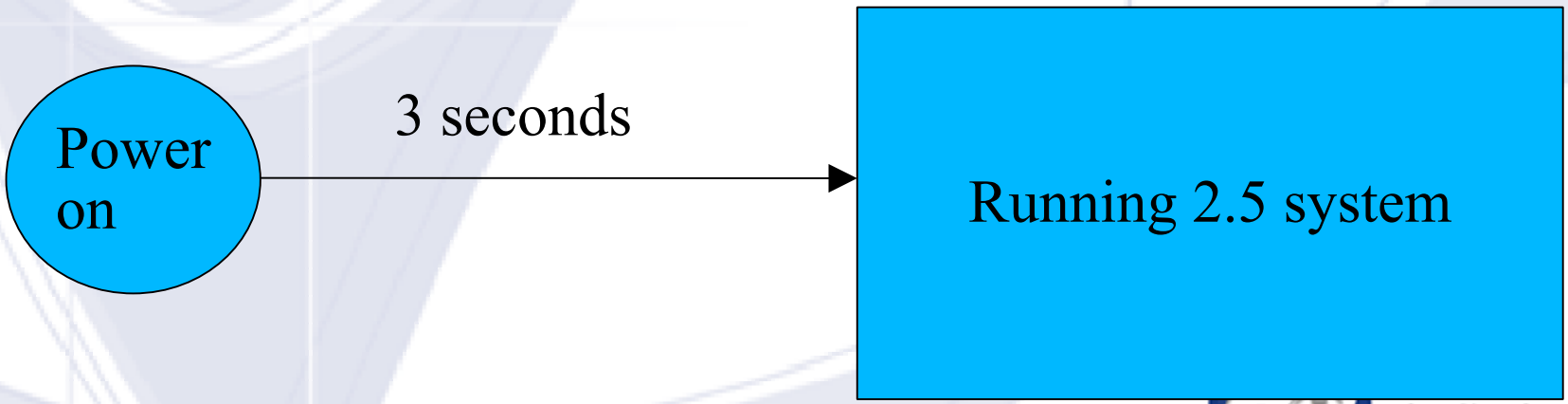
USB 2.0 will reach speeds of up to 480 mb/s as compared to USB 1.1's max of 12 mb/s. There should be no compatibility issues in using USB 1.1 device code with USB 2.0.

# Addition of ALSA
## (Advanced Linux Sound Architecture)

For the first time the ALSA code base was incorporated into the kernel (as of 2.5.5). This brings advanced support of a large number of sound cards to the kernel.

# LinuxBIOS support (initial support)

The LinuxBIOS project is aimed at replacing the normal BIOS with a little bit of hardware initialization and a compressed Linux kernel that can be booted from a cold start. While this project has been around for awhile, it has begun to work its way into the kernel.



# Bluetooth support

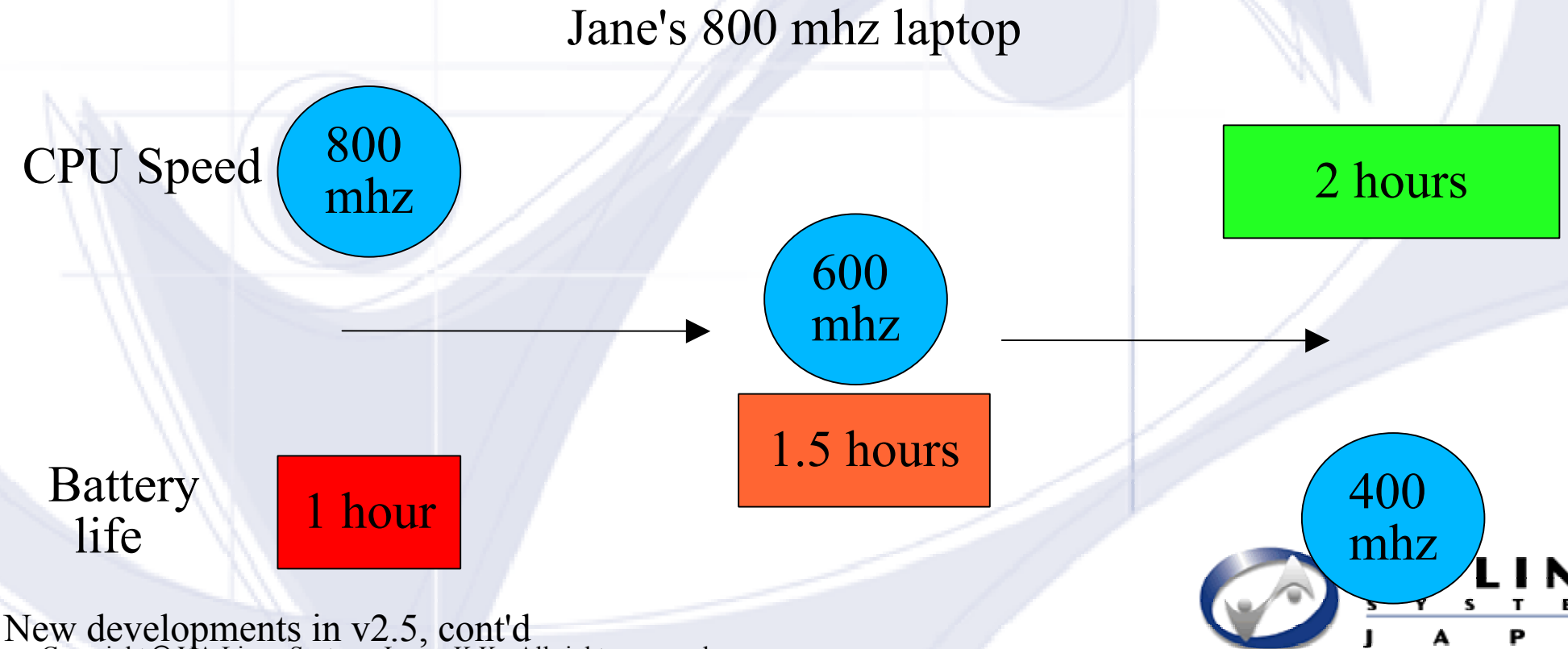
BlueZ, the official linux Bluetooth protocol stack, was merged into the kernel tree. (Bluetooth allows short-range wireless connections between computers)

# Suspend to RAM (or disk)

While a new feature of 2.5, new improvements have been made to 2.5.18 making suspending a machine to RAM or disk much more robust.

# Support for CPU clock/voltage scaling

Present in Intel CPU's since the Pentium and Transmeta's Crusoe, clock and voltage scaling is used mainly in laptops for optimal power consumption (and thus longer battery life). Support for manipulating both is coming to the kernel.



# Linux Security Module

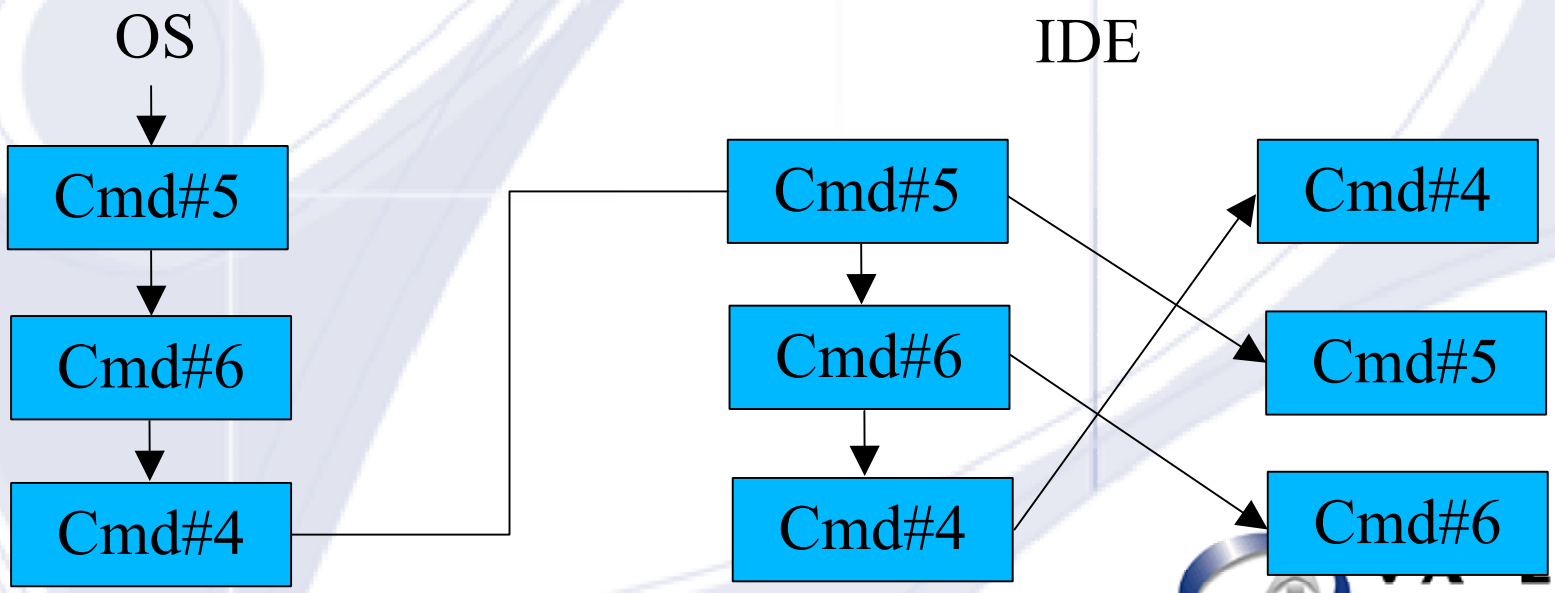
The security module provides the kernel with a general purpose framework for access control. This framework enables the development of enhanced security policies as loadable modules. (control of processes, resources mandatory access control, etc.)

# EVMS
## (enterprise volume management system)

EVMS provides flexibility and extensibility in managing storage, and is a new approach to logical volume management. The EVMS architecture introduces a plug-in module to the kernel for easy customization and expansion of various levels volume management.

# Support for IDE TCQ
## (tagged command queuing)

Tagged command queuing has been supported in SCSI drives for many years, but has just recently made its way into IDE. It allows commands to be 'tagged' so the OS does not necessarily have to send every command in order; the controller reassembles them, decreasing CPU load.



# New Mount API

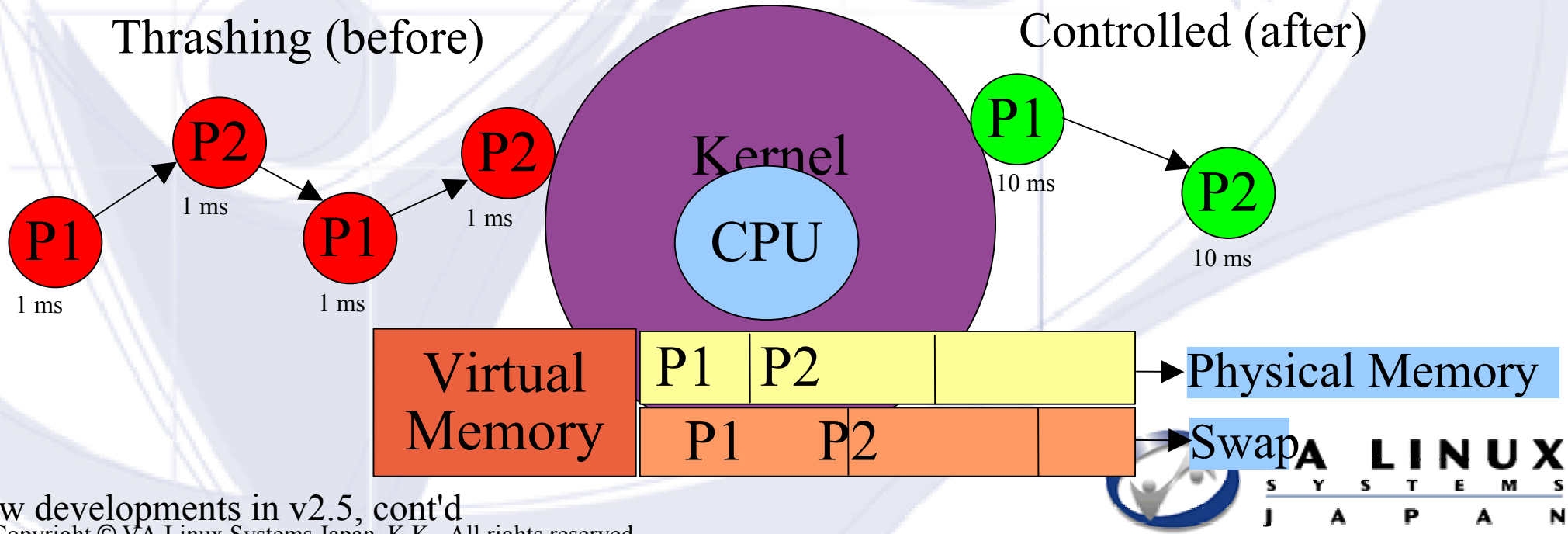
Al Viro is currently working on planning and implementing a new mount API. Currently flags are passed as a 32-bit value, with only 31 non-fs dependent flags (such as read-only or no-suid). Quota support is also limited due to poor support in mount and the kernel (via the API). The new API should resolve these limitations, by fixing problems with export and simplifying the handling of quotas.

# Preemptible Kernel

The preemptible patch serves to reduce latency in the kernel. This is particularly useful to normal users (faster response time) and embedded system developers. It allows processes to be preempted even in kernel mode. Average latencies have been measured at under 1 ms. (Nonpreemptible regions are marked by spinlocks).

# Thrashing Control

When many processes that use a vast amount of memory are paged in and out (swapped in and out rapidly) a condition occurs called thrashing. Detection of thrashing can be implemented, and thus controlled (such as IBM's AIX does). Rik van Riel is working on a solution for v2.5 which, when implemented, will improve Linux's place in the high-end server market.



# Sub-system changes in v2.5

## VM (Virtual Memory) Manager

Most of the major changes to the VM occurred in 2.4.10 when Andrea Archcangeli's VM was chosen over Rik van Riel's. Bug fixes and optimizations have been implemented since, up until the current 2.4.18. These VM changes were also introduced to 2.5. Actually, the 2.4 and 2.5 VM algorithms are basically the same.

Let's move on to look at some of these changes in more detail...

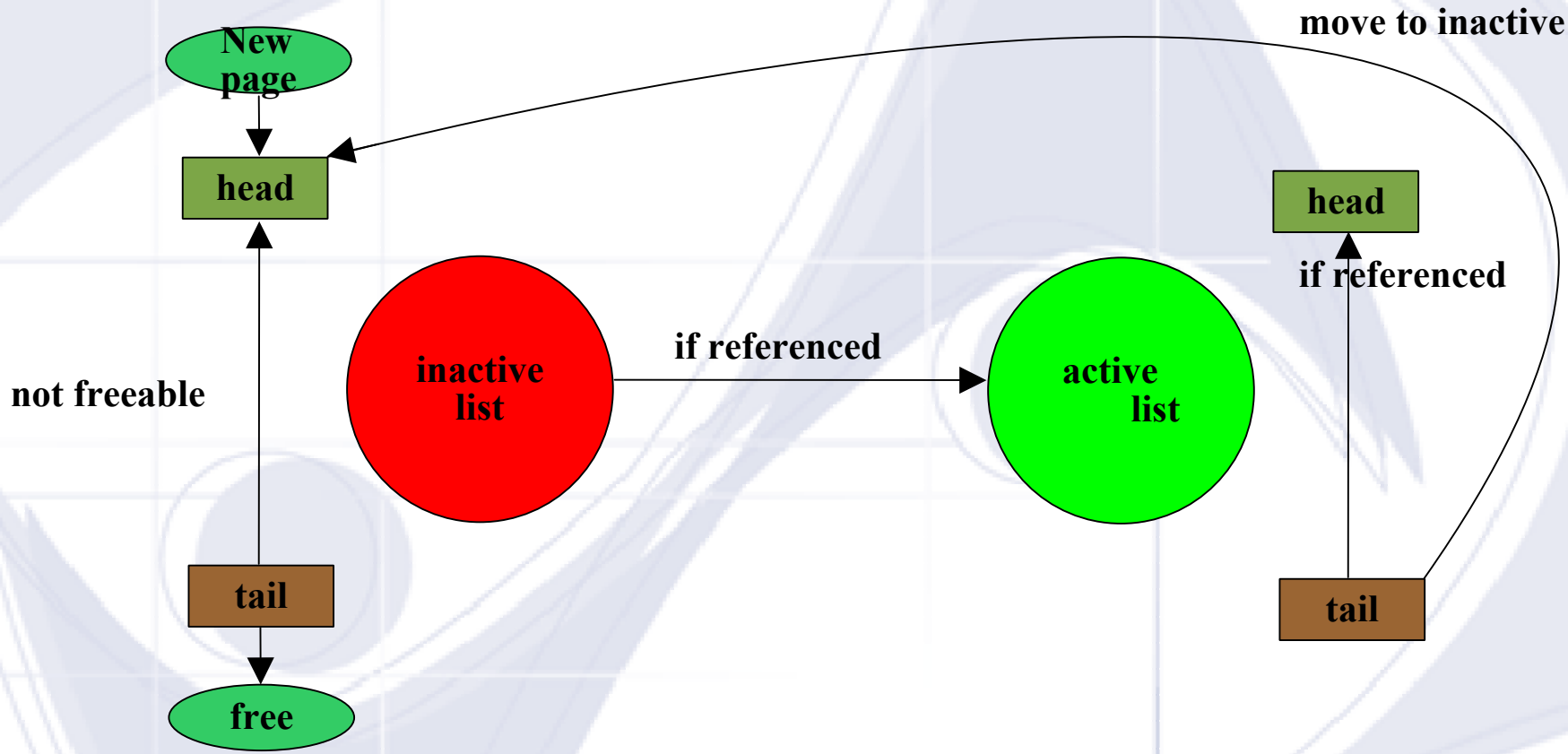
VA LINUX SYSTEMS JAPAN

## Two LRU (Least Recently Used) Lists

"active list": contains pages which have been marked as being frequently used and defines the working set of memory.

"inactive list": contains pages which are rarely referenced. Pages which are marked as dirty (have waiting I/O) are allowed to stay in the inactive_list for a few iterations while being out written to disk.

When visualized it looks like...




Sub-system changes in v2.5, cont'd

## Properties of the active and inactive lists

- New pages are added to the head of the inactive list
- When the system is low on memory, pages from the active list (tail) are moved to the inactive list (head)
- If an inactive page is referenced it is moved to the active list
- If a page in the active list is referenced it is moved to the head

## Properties of rotation

- The larger active is relative to inactive the faster active pages are moved to inactive
- Rotation will occur at the same speed if the size of active is the same as the VM balance ratio multiplied by the size of inactive
- Typically inactive rotates fastest

What kind of page conditions might we encounter and how would we handle them?

A busy page...

In particular we *need to be conscious of the status of the page as we try to free it from the inactive list. A page will be considered busy if it is mapped (referenced), locked (in I/O), or dirty (has I/O to be flushed).*

If the page is mapped...

...we need to unmap it from the address space before freeing it. This requires us to traverse the pagetable in a round robin fashion (swap_out()). While walking the pagetable we check to see whether the CPU has accessed a given page and subsequently activate them.

If the page is locked...
...we roll the page over and search for other pages to free. On the next pass we'll try to free this page again. The page could be locked due to unfinished I/O, being freed by another CPU, or part of a critical VM section.

If the page is dirty...
...write it out to disk using the callback of the underlying address space. Next time we come across the page it will be locked and we'll wait for I/O completion. If memory is not in demand then we don't need to worry about blocking for I/O completion.

# Out of Memory (OOM) Killer

When the system runs out of memory we need to have some way of freeing pages to make room for system critical processes. Rik van Riel designed the current OOM killer that decides which processes to reap for memory.

As Riel states his goals to be:

- Don't kill important system services
- Minimize the amount of work lost
- Free up as much memory as possible
- Be predictable
- Be simple and small

Using these goals as an outline, judgment is implemented as follows:


The OOM is based on a scoring system, so the process with the most points is killed.

# Rewrite of the IO block layer

Well deserved attention was given to the IO block layer which was in need of rewriting. Major changes that were made included:

- io_block_lock was separated from a single lock into many
- highmem IO no longer uses bounce buffers, straight to the device
- blocks are no longer broken down as they pass through layers

The layer still needs to be improved to handle a larger number of devices. Differentiation should also be made between read and write errors.

# O(1) Scheduler

The new scheduler by Ingo Molnar makes processes play even nicer than they used to. A new weighting system has been implemented. Weights included are:

- CHILD_PENALTY (whether child or not)
- PARENT_PENALTY (whether parent or not)
- EXIT_WEIGHT (if the process exits...)
- INTERACTIVE_DELTA (how interactive?)

The new system provides a much more fine-grained manner of differentiating between interactive processes and giving them proper CPU time.

VA LINUX SYSTEMS JAPAN

# Big Kernel lock removal

Removal of the “big kernel lock” actually begun quite some time ago, around v2.1. The lock is generated when a processor enters the kernel, thus locking resources, making it so only one processor can execute in the kernel at a time. As this severely effects SMP (the greater the # of CPUs the larger the bottleneck) more fine grained locking has since been implemented. However, the poll() (until 2.4) and select() system calls still depend on the BKL due to driver (amongst other) dependencies. Removal of the BKL continues as SMP support improves.

# Support for Next Generation POSIX

Current the (GNU) pthread implementation it not fully POSIX compliant and has issues with scaling. IBM is working on implementing a new thread system which has (near) POSIX compliance and offers N:M threading capability. Any sub-system or application utilizing the pthread library will see performance improvements. These improvements will also bring the Linux pthread implementation in lines with IBM AIX or SGI IRIX.

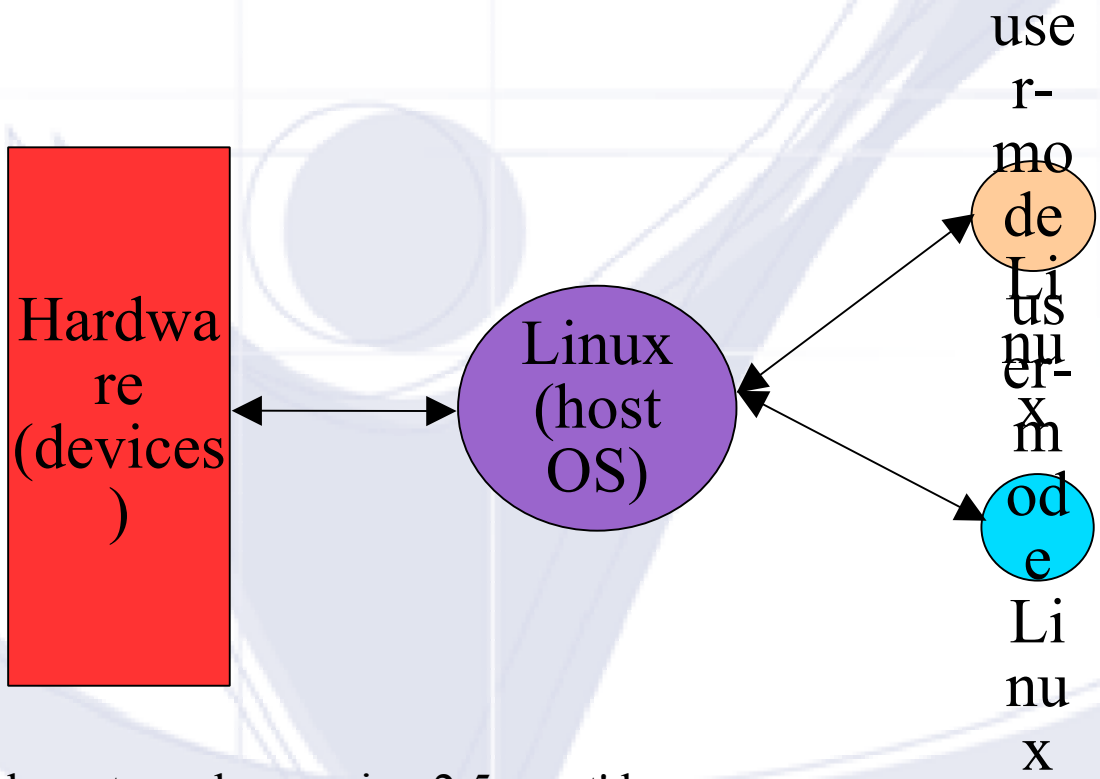
# Full compliance with IPv6

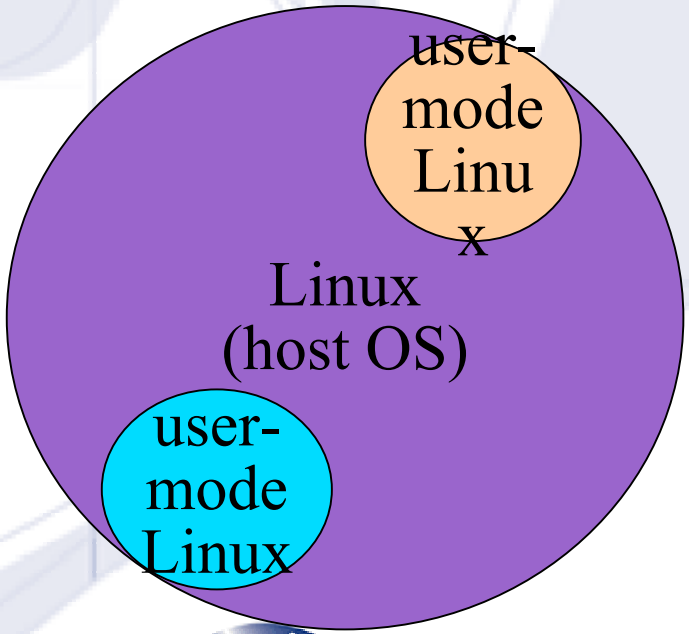
USAGI and Alexey Kuznetsov bring production quality IPv6 to 2.5!

VA LINUX SYSTEMS JAPAN

# User-mode Linux

User-mode linux has been around for awhile, but is aiming to be included in v2.5 (we'll see). Its goal is to provide a virtual machine runnable inside of Linux. The user can then choose which devices to allow the virtual machine access to, and use it for safe experimentation (kernel development, process debugging, etc).



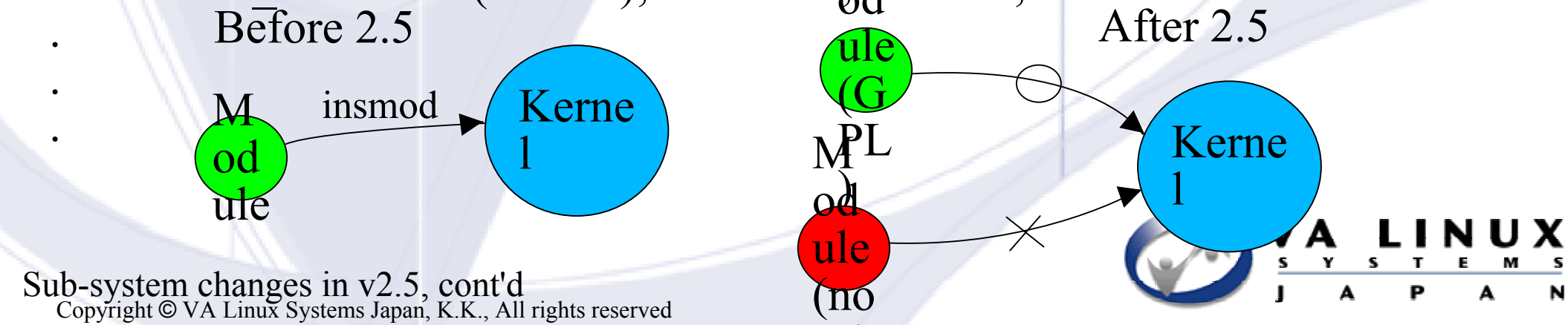



# Clean-up: modules

Before a module is loadable into the kernel its license must be set. Fields include: AUTHOR, LICENSE, DESCRIPTION, SUPPORTED_DEVICE, PARM and PARM_DESC. There is a push to remove all static device drivers, and simultaneously enforcing license declaration is a clear assertion of Linux's status as a GPL'ed system.

3c59x.c (net driver)

```
MODULE_AUTHOR("Donald Becker <becker@scyld.com>");
MODULE_DESCRIPTION("3c59x ethernet" DRV_VER
DRV_RELDATE);
MODULE_LICENSE("GPL"); # or BSD, or other, etc.
```



# kbuild 2.5

The rewritten kbuild is faster, better documented, easier to write build rules in, has better install facilities, allows separate source and object trees, supports CML1 and CML2, and is significantly more accurate than the current kernel building system. While it has not been included in the kernel yet (Linus has been ignoring Keith) it should be at some point soon.

# Bitkeeper

Management of the kernel has been moved from CVS to bitkeeper due to its extra functionality. Two major differences are a parent-child relationship instead of client-server, and the ability to 'push' and 'pull' changes for multiple cloned trees.

# CML2

Note that kbuild 2.5 and CML2 are different, and not interdependent. CML2 (written by Eric Raymond) aims to replace CML1 for configuring the kernel. If implemented in the manner of which Raymond speaks many config rule interdependences should be fixed, auto-configurations possible, help/config/make sets separated and localized. There are currently strong disagreements in the kernel circle about whether or not to use CML2.

# Zerocopy NFS <shameless plug>

Hirokazu Takahashi of VA Linux Systems Japan, K.K., developed a way of avoiding copying data into sk_buff (in csum_partial_copy_generic) meaning greatly improved I/O (in particular read). On an SMP machine the patch allows NFS to use up to 100% of the CPU. NFS is improving on Linux, although this patch only applies to NFS over TCP (work on NFS over UDP is underway).

# Linux kernel crash dumps

LKCD attempts to 1) save the kernel memory image when the system dies due to software failure, 2) recover that memory image on reboot, 3) analyze the memory image to determine what caused the failure. While LKCD has patches for 2.2, actual inclusion to the kernel is targeted for 2.5.

# Linux Trace Toolkit

LTT provides for dynamic tracing of the Linux kernel. For example, it is nearly impossible to debug inter-process communication using a conventional debugger, however it is possible with LTT. Synchronization problem solving and performance management are the two broad categories where LTT is unique in its capabilities.

# SGI's kernel debugger (kdb)

Available for both 2.4 and 2.5, when kdb is enabled in the kernel and a crash occurs you are dropped out to a kdb debugging prompt (similar to gdb). From kdb you can trace where the problem occurred, check memory and data structures, stop execution on a specific instruction or access to memory; all while the system is operational.

# 64-bit support

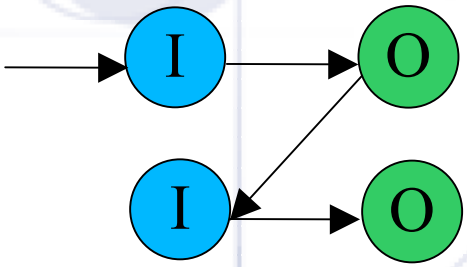
64-bit support of CPU's (as the CPU's themselves are being released) is included in 2.5. Currently linux supports AMD, Intel, IA, PPC, and SPARC's 64-bit CPU's (often before other OS's).

# Asynchronous I/O

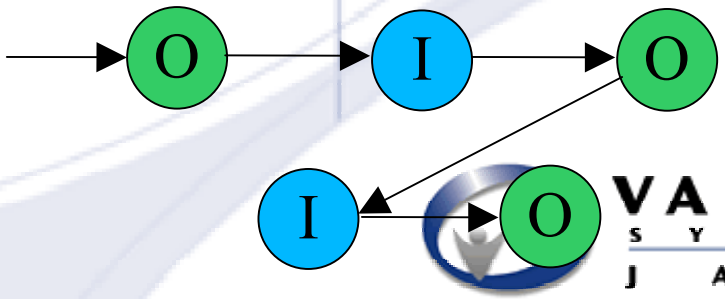
The lack of asynchronous I/O has always capped device driver and raw I/O performance. Finally in 2.5 Ben LaHaise has begun implementing async I/O which will make efficient use of raw I/O (to any device) and further expand Linux's diversity. For the technical, asynchronous I/O will be implemented using the following system calls

- __submit_ios()
- __io_cancel()
- __get_events()
- __io_wait()

Synchronous I/O




Asynchronous I/O




VA LINUX SYSTEMS JAPAN

# NUMA

NUMA is an architecture where memory access for different regions of memory from a given processor varies according to the “distance” of the memory region from the processor. The NUMA architecture is heavily used in large systems with many nodes (CPU's), from 2 up to 32. For NUMA it is important to keep inter-process communication at a low, and page allocation data structures must be distributed on a per-node basis. Corporations are leading NUMA Linux development cooperatively (as price is a deterrence for individuals), specifically, HP, IBM, SGI, and Sun. This is a good example of Linux's expansion into hardware (and markets) that were traditionally dominated by proprietary UNIX(s).

# “Big iron”

Linux on mainframes? Yup. Linux currently runs on IBM's zSeries, and S/390 mainframes (from $250,000). IBM's desire to get Linux running on big iron has fueled development to improve Linux's utilization of large amounts of memory, multiple CPU's, and large file-systems.

# Overall trends

## Legalese

The landscape of software development has changed drastically since 1994. The DMCA (Digital Millennium Copyright Act) has been passed in the US, companies are patenting software left and right, and organizations are being taken to court at the drop of a hat. Utilizing media, Microsoft has begun to attack Linux and the GPL in earnest. Despite all of this kernel development remains steadfast. Perhaps as a defensive measure, mandatory license declaration of module code has been implemented. Compared to 8 years ago developers must be more careful in making sure code is written in a legal manner, and their code isn't stolen.

VA LINUX SYSTEMS JAPAN

# Corporate support

IBM alone spent 1 billion USD on linux during 2001 (media, development, etc). After the dot-com bubble burst in the US stock market more companies began considering Linux as a viable solution due to its low-cost. Many corporations are pushing new development to meet this demand and expand their markets. Such companies include: IBM, Intel, HP & Compaq, SGI, and Sun. The new influx of money and developers has effected development in several fundamental ways:

- Explosion of development
- Intermingling of developers working for fun as opposed to money
- Smaller teams working on specific projects producing tighter code (XFS, IBM's new pthreads, kdb, etc.)
- Much of the code is maintained separately as patches rather than being submitted for inclusion (Linus does not always approve).
- Conflicts in sub-system implementation; does the change really work best for all systems? (or embedded, or mainframe, etc.)

# In conclusion...

Where is the kernel heading?

- v2.6 or 3.0 in the next year or two

What can be expected?

- Greater stability
- More device driver support
- Robustness across a variety of hardware
- Better support of large amounts of memory
- More reliable under extremely heavy loads

World Domination?

Not quite yet...

VA LINUX SYSTEMS JAPAN

御静聴有難う御座いました。
Thank you for attending!

VA LINUX SYSTEMS JAPAN