Itanium and the Linux Kernel

Andrew Morton
<akpm@linux-foundation.org>
<akpm@google.com>
Gelato ICE
April 2007

Overview

- Assume that the audience have an interest in future kernel developments
  - And a wish to be able to influence them
- Will discuss
  - How the kernel comes together
  - Itanium's position in the overall effort
  - What those who care about Itanium may be able to do to tilt things in their direction
  - How Itanium users can help the effort
  - Will not address Itanium-specific technological issues
    - That's what Tony is for

Development processes: patch flow

- Kernel has 50-100 subsystems, each with a single maintainer
- Individuals feed work into that maintainer
- Maintainers merge into Linus's tree
- I fill in a lot of the remaining gaps
- All subsystem trees are published in the -mm kernels for test and review

Development processes: patch integration

- After a kernel release (eg 2.6.21) there is a 2-week merge window during which all new feature work is merged
- Consolidation and bugfixing happens over the next two months
- During this period, feature work for the next kernel accumulates in the subsystem trees (and hence in -mm tree)
- Once the next 2.6.x tree is released, we open the 2.6.x.y "stable" tree: a bugfixed version of the 2.6.x tree.
- 2.6.x.y contains backports of fixes which went into 2.6.x+1

Kernel development processes

- Very high change rate: around 10,000 LOC added+removed per day
- Bug resolution via emailed reports and bugzilla.kernel.org
- Developers are mainly professionals from distros, IT corps, hardware vendors and device manufacturers. Also private individuals
- Distros will pick a particular 2.6.x release on which to base their major release and will monitor the ongoing 2.6.x+N stream for fixes and features to backport into it
- After testing and stabilization, the vendor will release that kernel as an Enterprise product

Directing kernel development

- How does the kernel team decide which features they will work on?
  - As a result of funding decisions by their employers
  - In reaction to user feedback
  - Personal itches
- How can you use this process to get your features implemented?

Directing kernel development: funding decisions

- Companies who employ kernel developers (or who have commercial relationships with their employers)
- Make (hopefully) commercial(ly sound) decisions about which features
  - their customers need
  - will create revenue
- A lot of kernel development happens for these reasons
- Lessons for Itanium users:
  - Employ kernel developers
  - Shovel $$ at companies who employ kernel developers
  - Lobby companies who employ kernel developers

Directing kernel development: user feedback

- Lots of professional kernel developers take feedback from users quite seriously
- They also have some freedom to make personal decisions about what they will work on
- This is good: a feedback path from end users into kernel developers helps developers optimize the value of their work
- The potential for abuse is obvious: the squeaky hinge gets the oil
- Lessons for Itanium users:
  - Make your presence felt (ie: squeak!)
  - Make a fuss about things you care about on mailing lists, esp. lkml
  - This includes bug reporting
  - A performance problem is a bug too!

Directing kernel development: personal itches

- A significant amount of kernel work comes from people who do it for personal satisfaction
- There is no obvious way of causing such people to become interested in your particular problem
- Except, perhaps, by making Itanium available to mortals

Community perceptions of Itanium

- Itanium has a perception problem in "the community"
  - costly, not a top performer, high power usage, commercially unsuccessful
- But "the community" consists of many subgroups
- Those people who actually *matter* to Itanium are a subgroup of a subgroup
  - Serious, professional, responsible and mature
    - ie: not slashdotters
- Itanium would benefit from being more accessible
  - ie: desktop form factor
  - Mac laptops and G5 probably helped Power
- Tukwila's "common platform architecture": Itanium desktops for mortals?

Itanium's status in the kernel project

- Itanium is a second-tier architecture. Implications:
  - kernel tends to be optimized for the first-tier architectures: x86, x86_64
  - there is much less testing of mainline development kernels for Itanium
  - a change which improves x86 but degrades ia64 would be rejected, but these things can happen by accident
  - powerpc, s390, sparc64, arm and (at present) blackfin are also getting decent amounts of devel-and-test against mainline. Itanium is probably in a similar position.

Itanium's status in the kernel project (cont'd)

- Testing of development kernels for second-tier architectures is mostly by kernel developers
  - x86 gets a lot of end-user testing too.
  - kernel developer testing isn't sufficient
- Net result of all of this: Itanium-affecting kernel problems can sneak through the process more easily
- Core kernel tends to assume x86
  - other architectures sometimes must pretend to be x86
  - and that's OK - we're optimizing for the common case
- Core kernel developers understand x86 best, and sometimes assume x86 semantics
  - sometimes an architecture just cannot live with a core kernel assumption
    - hooks are placed in core kernel to accommodate that architecture
    - That's fine, and we do it regularly

Itanium's status in the kernel project (cont'd)

- Missed opportunities: because core-kernel developers are not familiar with Itanium
  - there might be optimization opportunities which do not occur to them
- Do we actually have a problem?
  - Not as far as I know.
- If we do, how to improve the situation?
  - Developers who are familiar with Itanium requirements, strengths and weaknesses should participate more closely in core-kernel development
  - At least by reviewing what's happening, participating in design activities
- Itanium users tend to be disconnected from mainline kernel development
  - This has opportunity costs

Example: CPU scheduler changes

- Kernel's core CPU scheduler is about to be rewritten
- Itanium users have an interest in scheduler behavior
- There may be missed opportunities for improvement
- We might cause regressions which affect Itanium
  - We already know of one
  - It was reported just this month
    - Against kernel 2.6.9!
- The scheduler changes are being driven by desktop users, focused on interactivity
  - Probably only a group of 10-20 people
    - Very noisy people

Example: containerization

- The kernel containerization project is still ramping up
- A very large project
- Disparate groups with disparate requirements
  - Workload consolidation
  - Virtual servers
  - Resource management
  - High availability
- Requirements are still being discussed
  - They are very complex
- Some infrastructure has already been merged
- Large changes are perhaps only a month away
- Designers and decision makers need to hear from all stakeholders *now*

Example: RAS

- Various RAS initiatives have appeared over the years
  - Then disappeared
- OS and hardware vendors understand the need for RAS
  - So do the bulk of the kernel development team
    - But it makes us fall asleep
- Previous RAS contributions have come from non-core individuals
  - And met a somewhat critical reception
- Kernel developers rely on personal experience when judging requirements
  - Few of us have first-hand experience with such systems
- But we also rely upon input from end-users
  - When it is available
- We need help understanding RAS requirements
- How RAS is used, why it is important

Kernel lobbying 101

- This doesn't only apply to companies who employ kernel developers
  - Nor only to tech companies
- We love to hear from end-users about their problems
- End users (esp. those from large orgs) are reticent
  - If they do have issues, they work them through their vendors
  - But each vendor controls only a small fraction of the kernel team
  - The vendor has to make a business decision with your requirement
  - And knowledge is compartmentalized
    - one particular vendor might not have suitably skilled engineers for the affected area
  - When the kernel team as a whole hears about these issues (if they do), they are second-hand, and filtered
    - We believe vendors, but it is better coming from the end-users
  - Others need info to make merge decisions (eg: me)

Kernel lobbying 101

- Distro engineers may end up developing your feature, but it impacts all kernel maintainers
- No single company controls more than 10% of kernel development resources
    - Getting that company on board only solves a fraction of the problem

Kernel lobbying 101 (cont'd)

- We use email almost exclusively
  - Don't expect to make progress using face-to-face communication
- Lesson: be brave, email us
- If you're not that brave, email an individual first
- I can help
- Identify an interface individual in your organization
- To get resources allocated to your issue, you need to make noise about it
- The more noise (and the more serious the issue), the better the response will be
- All merge decisions involve cost/benefit tradeoffs
  - Part of that decision involves asking “who stands to gain?”
  - “who” includes “how many such people are there?”
  - If your interests are not represented, mistakes can be made

Kernel lobbying 101 (cont'd)

- Lobbying is not purely for your advantage
- Kernel developers can make mistakes when analyzing requirements
  - Receiving input from a broad set of sources minimizes the error rate
- Kernel developers' personal experience is not sufficient to cover all industry requirements
  - We need educating
- It is better if a specialized user's requirement is generally useful, or can be made so

Testing kernel.org kernels

- This is to your advantage
- Testing 2-year-old vendor kernel: expensive, slow to get fixes
  - Doesn't help others
- Enables you to evaluate new kernel features
  - Ensure that they work optimally for your application
- This is the reason you are using open source
  - Immediate interaction with core developers
  - Value needs to flow in both directions
- Do you have a business case for doing regular kernel testing?
  - Probably not
  - As if I care

Reporting bugs

- It's your fault that the bug got into a vendor kernel anyway
  - You should have tested the kernel during its development
- Option a: report to vendor
  - Slow, can cost
  - Costs the vendor too
  - Others are being affected
  - We'd prefer that the bug never was there in the first place
  - It's still your fault
  - Vendor then needs to remember to forward-port the fix into mainline
    - Can take ages, and might even get lost

Reporting bugs (cont'd)

- Option b:
  - Verify that the bug is still present in mainline kernel. If so:
    - Tell the vendor!
      - They will realize that they just need to find the right patch to backport
      - They'll probably act more promptly
  - If the bug is in mainline, report it to kernel.org team instead.
    - Chances are, it'll get fixed quickly
    - Then the vendor can backport it promptly

Itanium and the Linux Kernel