The List is Our Process!
An analysis of the kernel’s email-based development process

Ralf Ramsauer*, Sebastian Duda†, Lukas Bulwahn‡, Wolfgang Mauerer*§

* Technical University of Applied Sciences Regensburg
† Friedrich-Alexander University Erlangen-Nürnberg
‡ Hobbyist, active in LF ELISA Project, employed at BMW AG
§ Siemens AG, Corporate Research and Technology, Munich

Linux Plumbers Conference, Lisbon
September 9, 2019
Our Overall Goal

Formalising and assessing the Linux Kernel development process
Our Overall Goal
Formalising and assessing the Linux Kernel development process

Outside / Inside Motivation
- Safety-Critical Development
- Development Process Assessment
- Monitoring (cf. CHA OSS)
- Fundamentals of Software Engineering
Motivation from *outside* the community

**Academic Research**
- Applying statistical (big data, ML, etc.) methods on software development projects
- Linux kernel serves as a prime example for their evaluation

**Commercial Users**
- Measuring progress of certain long-term kernel developments, e.g., mainlining the PREEMPT_RT patch
- Measuring and monitoring many business-critical open-source projects at scale (CHAOSS)
- Fulfilling the certification requirement "Development process assessment" in regulated environments (safety- and security-related certified systems)
Motivation from *inside* the community

### Interest of the kernel community itself

- D. Williams, *Towards a Linux Kernel Maintainer Handbook*, LPC 2018
- J. Corbet, Change IDs for kernel patches, [https://lwn.net/Articles/797613/](https://lwn.net/Articles/797613/)
- [Ksummit-discuss] [MAINTAINERS SUMMIT] Patch version changes in commit logs?
- [Ksummit-discuss] Allowing something Change-Id (or something like it) in kernel commits
Towards a Linux Kernel Maintainer Handbook

Dan Williams
Linux Plumbers 2018

1. Motivation

Subsystem Profile

- Describe maintenance of your sub-system as a “bus-factor”
  document
- Document the policies that each subsystem typically decides
differently

Patches or Pull requests
- Last day for new feature submissions
- Last day to merge features
- Non-author Ack / Review Tags Required
- Test Suite

Trusted Reviewers
- Resubmit Cadence
- Time Zone / Office Hours
- Checkpatch / Style cleanups
- Off-list review

Others?
### Subsystem Profile

- Describe maintenance of your sub-system as a “bus-factor” document
- Document the policies that each subsystem typically decides differently

<table>
<thead>
<tr>
<th>Patches or Pull requests</th>
<th>Trusted Reviewers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last day for new feature submissions</td>
<td>Resubmit Cadence</td>
</tr>
<tr>
<td>Last day to merge features</td>
<td>Time Zone / Office Hours</td>
</tr>
<tr>
<td>Non-author Ack / Review Tags Required</td>
<td>Checkpatch / Style cleanups</td>
</tr>
<tr>
<td>Test Suite</td>
<td>Off-list review</td>
</tr>
</tbody>
</table>

Others?
Subsystem Profile

- Describe maintenance of your sub-system as a “bus-factor” document
- Document the policies that each subsystem typically decides differently

<table>
<thead>
<tr>
<th>Patches or Pull requests</th>
<th>Trusted Reviewers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last day for new feature submissions</td>
<td>Resubmit Cadence</td>
</tr>
<tr>
<td>Last day to merge features</td>
<td>Time Zone / Office Hours</td>
</tr>
<tr>
<td>Non-author Ack / Review Tags Required</td>
<td>Checkpatch / Style cleanups</td>
</tr>
<tr>
<td>Test Suite</td>
<td>Off-list review</td>
</tr>
</tbody>
</table>

Others?
Towards a formal model of the development process

1. patch is created

2. patch is on mailing list

3. patch is in git repository

Patch evolution relationship

submit patch

rework patch

......

integrate patch

private

public
Linux Kernel development workflow
2. Methodology

Linux Kernel development workflow

PATCH
2. Methodology

Linux Kernel development workflow

![Diagram of git and patch process]
Linux Kernel development workflow
Linux Kernel development workflow

Message-Id: <1531137835-21581-1-git@1wt.eu>
Linux Kernel development workflow

Message-Id: <1531137835-21581-1-git@1wt.eu>

commit 2f6e24d3151fb9967774f9721b288f216f3180df
Linux Kernel development workflow

Message-Id:
v1: <1531137835-21581-1-git@1wt.eu>
v2: <6739637657-68462-1-git@1wt.eu>
v3: <9717683099-75474-1-git@1wt.eu>

commit
2f6e24d3151fb9967774f9721b288f216f3180df
2. Methodology

Linux Kernel development workflow

Message-Id:

v1: <1531137835-21581-1-git@1wt.eu>
v2: <6739637657-68462-1-git@1wt.eu>
v3: <9717683099-75474-1-git@1wt.eu>

commit
2f6e24d3151fb9967774f9721b288f216f3180df89876f275e8d5e2912d9c238cd888b52065cf25
Linux Kernel development workflow

Message-Id: 
v1: <1531137835-21581-1-git@1wt.eu>
v2: <6739637657-68462-1-git@1wt.eu>
v3: <9717683099-75474-1-git@1wt.eu>

commit 2f6e24d3151fb9967774f9721b288f216f3180df
89876f275e8d562912d9c238cd888b52065cf25
PaStA - Patch Stack Analysis

- Detects *similar* patches across different branches
- Quantify mainlining efforts of off-tree developments (Preempt_RT, vendor kernels, ...)
- Works with mailing lists!

Source: toplock.net.au
Example of similar patches

commit 91824d74d6d85f58c63a66b8f2c7993ae246181b
Author: Thomas Gleixner <tglx@linutronix.de>
Date: Mon Sep 12 21:45:49 2011 +0200

sched−cure−utter−idle−accounting−madness.patrch

Signed−off−by: Thomas Gleixner <tglx@linutronix.de>

diff --git a/kernel/sched.c b/kernel/sched.c
index 205499a..1121a97 100644
−−− a/kernel/sched.c
+++ b/kernel/sched.c
@@ −5037,7 +5037,13 @@ EXPORT_SYMBOL(task_nice);
    */
    int idle_cpu(int cpu)
    {
−    return cpu_curr(cpu) == cpu_rq(cpu)->idle;
+    struct rq *rq = cpu_rq(cpu);
+    if (rq->curr != rq->idle)
+       return 0;
+    if (rq->nr_running)
+       return 0;
+    #ifdef CONFIG_SMP
+    if (!llist_empty(&rq->wake_list))
+       return 0;
+    #endif
+    return 1;
    }

/∗∗

commit 908a3283728d92df36e0c7cd63304fd35e93a8a9
Author: Thomas Gleixner <tglx@linutronix.de>
Date: Thu Sep 15 15:32:06 2011 +0200

sched: Fix idle_cpu()

On −rt we observed hackbench waking al 400 tasks to a single
cpu. This is because of select_idle_sibling()'s interaction
with the new ipi based wakeup scheme.

[...snip...]
Signed−off−by: Thomas Gleixner <tglx@linutronix.de>
Signed−off−by: Peter Zijlstra <a.p.zijlstra@chello.nl>
Link: http://lkml.kernel.org/n/tip−3o30p18b2 [...]
Signed−off−by: Ingo Molnar <mingo@elte.hu>

diff --git a/kernel/sched.c b/kernel/sched.c
index 1874c74..4cdc91c 100644
−−− a/kernel/sched.c
+++ b/kernel/sched.c
@@ −5138,7 +5138,20 @@ EXPORT_SYMBOL(task_nice);
    */
    int idle_cpu(int cpu)
    {
−    return cpu_curr(cpu) == cpu_rq(cpu)->idle;
+    struct rq *rq = cpu_rq(cpu);
+    if (rq->curr != rq->idle)
+       return 0;
+    if (rq->nr_running)
+       return 0;
+    #ifdef CONFIG_SMP
+    if (!llist_empty(&rq->wake_list))
+       return 0;
+    #endif
+    return 1;
    }

/∗∗
Example of similar patches

commit 91824d74d6d85f58c63a66b8f2c7993ae246181b
Author: Thomas Gleixner <tglx@linutronix.de>
Date: Mon Sep 12 21:45:49 2011 +0200


Signed−off−by: Thomas Gleixner <tglx@linutronix.de>

diff —− git a/kernel/sched.c b/kernel/sched.c
index 205499a..1121a97 100644
− − − a/kernel/sched.c
+++ b/kernel/sched.c
@@ −
− 5037,7 +5037,13 @@EXPORT_SYMBOL(task_nice);
∗
/
int idle_cpu(int cpu)
−
return cpu_curr(cpu) == cpu_rq(cpu)
− >idle;
−
+ struct rq ∗rq = cpu_rq(cpu);
+ if (rq−>curr != rq−>idle)
+ return 0;
+ if (rq−>nr_running)
+ return 0;
+ #ifdef CONFIG_SMP
+ if (!llist_empty(&rq−>wake_list))
+ return 0;
+ #endif
+ return 1;
+}

sched: Fix idle_cpu()

On−rt we observed hackbench waking all 400 tasks to a single
cpu. This is because of select_idle_sibling()’s interaction
with the new ipi based wakeup scheme.
[...snip...]
Signed−off−by: Thomas Gleixner <tglx@linutronix.de>
Signed−off−by: Peter Zijlstra <a.p.zijlstra@chello.nl>
Link: http://lkml.kernel.org/n/tip−3o30p18b2[...]
Signed−off−by: Ingo Molnar <mingo@elte.hu>

commit 908a3283728d92df36e0c7cd63304fd35e93a8a9
Author: Thomas Gleixner <tglx@linutronix.de>
Date: Thu Sep 15 15:32:06 2011 +0200

On—rt we observed hackbench waking all 400 tasks to a single
cpu. This is because of select_idle_sibling()’s interaction
with the new ipi based wakeup scheme.
[...snip...]
Signed−off−by: Thomas Gleixner <tglx@linutronix.de>
Signed−off−by: Peter Zijlstra <a.p.zijlstra@chello.nl>
Link: http://lkml.kernel.org/n/tip−3o30p18b2[...]
commit 91824d74d6d85f58c63a66b8f2c7993ae246181b
Author: Thomas Gleixner <tglx@linutronix.de>
Date: Mon Sep 12 21:45:49 2011 +0200

sched
−
cure
−
utter
−
idle
−
accounting
−
madness.patrch

Signed−off−by: Thomas Gleixner <tglx@linutronix.de>

diff − − git a/kernel/sched.c b/kernel/sched.c
index 205499a..1121a97 100644
− − − a/kernel/sched.c
+++ b/kernel/sched.c
@@ −5037,7 +5037,13 @@EXPORT_SYMBOL(task_nice);
∗
/
int idle_cpu(int cpu)
−
return cpu_curr(cpu) == cpu_rq(cpu)
−
>idle;
+ struct rq ∗rq = cpu_rq(cpu);
+ 
+ #ifdef CONFIG_SMP
+ return rq
−
>curr == rq
−
>idle && !rq
−
>nr_running && !rq
−
>wake_list;
+ 
+ #else
+ return rq
−
>curr == rq
−
>idle && !rq
−
>nr_running;
+ 
+ #endif
+ return 1;
+ }
/
∗∗

commit 908a3283728d92df36e0c7cd63304fd35e93a8a9
Author: Thomas Gleixner <tglx@linutronix.de>
Date: Thu Sep 15 15:32:06 2011 +0200

sched: Fix idle_cpu()

On −rt we observed hackbench waking all 400 tasks to a single

cpu. This is because of select_idle_sibling()’s interaction

with the new ipi based wakeup scheme.

[...snip...]
Signed−off−by: Thomas Gleixner <tglx@linutronix.de>
Signed−off−by: Peter Zijlstra <a.p.zijlstra@chello.nl>
Link: http://lkml.kernel.org/n/tip−3o30p18b2 [...]
Signed−off−by: Ingo Molnar <mingo@elte.hu>

diff − − git a/kernel/sched.c b/kernel/sched.c
index 1874c74..4cdc91c 100644
− − − a/kernel/sched.c
+++ b/kernel/sched.c
@@ −5138,7 +5138,20 @@EXPORT_SYMBOL(task_nice);
∗
/
int idle_cpu(int cpu)
−
return cpu_curr(cpu) == cpu_rq(cpu)
−
>idle;
+ struct rq ∗rq = cpu_rq(cpu);
+ 
+ #ifdef CONFIG_SMP
+ if (!llist_empty(&rq
−
>wake_list))
+ return 0;
+ #endif
+ 
+ #ifdef CONFIG_SMP
+ if (!llist_empty(&rq
−
>wake_list))
+ return 0;
+ 
+ #endif
+ 
+ return 0;
+ +
+ return 1;
+ }
/
∗∗
Example of similar patches

commit 91824d74d6d85f58c63a66b8f2c7993ae246181b
Author: Thomas Gleixner <tglx@linutronix.de>
Date: Mon Sep 12 21:45:49 2011 +0200

sched
−
cure
−
utter
−
idle
−
accounting
−
madness.patrch

Signed−off by: Thomas Gleixner <tglx@linutronix.de>

diff − − git a/kernel/sched.c b/kernel/sched.c
index 205499a..1121a97 100644
− − − a/kernel/sched.c
+++ b/kernel/sched.c
@@ −5037,7 +5037,13 @@EXPORT_SYMBOL(task_nice);
∗

int idle_cpu(int cpu)
−
return cpu_curr(cpu) == cpu_rq(cpu)
−
>idle;
+ struct rq ∗rq = cpu_rq(cpu);
+ #ifdef CONFIG_SMP
+ return rq
−
>curr == rq
−
>idle && !rq
−
>nr_running && !rq
−
>wake_list;
+ #else
+ return rq
−
>curr == rq
−
>idle && !rq
−
>nr_running;
+ #endif
+ return 1;
}/∗∗

Di/uniFB00similarity: 0.875
2. Methodology

Legend

- green nodes: patches on MLs
- orange nodes: commits in repository
- edges: similarity of patches/commits
  - dashed: similarity below thres
  - solid: similarity above thres
2. Methodology

- **Legend**
  - green nodes: patches on MLs
  - orange nodes: commits in repository
  - edges: similarity of patches/commits
    - dashed: similarity below thres
    - solid: similarity above thres
Interested in the techniques? More details in:

Observing Custom Software Modifications: A Quantitative Approach of Tracking the Evolution of Patch Stacks

Ralf Ramsauer
daniel lohnmann
Friedrich Alexander University
Bavaria, Germany
rolf.ramsauer@fau.de
lohnmann@fau.de
September 9, 2019

Abstract

The impact of software change on existing systems is manifold and complex. The underlying reason for this broad influence is the necessity to maintain existing systems while being compatible with the original software. In the context of large software projects, this is a daunting task, especially when the software is not open source. This paper presents a method to identify the evolution of software change and patch stacks. The method is based on a series of experiments that compare the evolution of software change on existing systems. The results show that the evolution of software change is complex and challenging. The method is a valuable tool for identifying the evolution of software change on existing systems.

1. Introduction

The rapid development of software systems has led to the development of new tools and techniques to analyze the evolution of software change. This paper presents a method to identify the evolution of software change on existing systems. The method is based on a series of experiments that compare the evolution of software change on existing systems. The results show that the evolution of software change is complex and challenging. The method is a valuable tool for identifying the evolution of software change on existing systems.

2. Methodology

The method is based on a series of experiments that compare the evolution of software change on existing systems. The results show that the evolution of software change is complex and challenging. The method is a valuable tool for identifying the evolution of software change on existing systems.

3. Results

The results show that the evolution of software change is complex and challenging. The method is a valuable tool for identifying the evolution of software change on existing systems.

4. Conclusion

The rapid development of software systems has led to the development of new tools and techniques to analyze the evolution of software change. This paper presents a method to identify the evolution of software change on existing systems. The results show that the evolution of software change is complex and challenging. The method is a valuable tool for identifying the evolution of software change on existing systems.
Data Acquisition

▷ Dumps from gmane.org etc.
▷ kernel.org public inboxes
  ▷ some lists, prehistoric data
  ▷ https://lore.kernel.org/lists.html
▷ Our own collection
  ▷ 200 lists, since May ’19
  ▷ https://github.com/linux-mailinglist-archives
Let there be chaos

- Broken encoding
- BĀse64
- MUAs
- Bots
- HTML
- Automated mails
- non-Linux patches
- Stable reviews
- Malformed recipients
- ...
2. Methodology

Message-Id: <74851t0$h3103wn0$Delldi Fri, 9 Mar 71685 18:45:56 +0000
Date: Mon, 08 Aug 05 04:01:15 ?x? ????????
Date: Tue, 27 Mar 22001 13:42:39 +0200 (Westeuropäische Sommerzeit)
2. Methodology

X-Mailer: Microsoft Outlook Express 6.00.2900.3028
We analyse...

- v2.6.39..linus/master
- \(\approx 610K\) commits
- \(\approx 3M\) mails
- Lists: All Public Inboxes from lore.kernel.org
We analyse...

- v2.6.39..linus/master
- \( \approx 610K \) commits
- \( \approx 3M \) mails
- Lists: All Public Inboxes from lore.kernel.org
2. Methodology

2011-05-2018-12

3M Mails: 1.15M Patches, 1.85M Non-Patches
2. Methodology

2011-05-2018-12

3M Mails

1.15M Patches

1.85M Non-Patches

1.15M Patches
2. Methodology

3M Mails

- 1.15M Patches
- 1.85M Non-Patches

- 1.11M Linux Kernel Patches
- Other 1.15M Patches
2. Methodology
2. Methodology

3M Mails

1.15M Patches
1.85M Non-Patches

1.11M Linux Kernel Patches
1.15M Patches

~80% Actual Patches
1.11M Linux Kernel Patches

882K Linux Kernel Patches
2. Methodology

3M Mails

- 1.15M Patches
- 1.85M Non-Patches

1.11M Linux Kernel Patches

- ~80% Actual Patches
- Stable Review
- Bots
- git pull

882K Linux Kernel Patches

794K Relevant Patches
Increasing Levels of Data Aggregation

Our main work

- Focus on the first two levels and *selectively* interpret observations to *describe* behaviour
- We do **NOT** try and **NOT** intend to judge behaviour
Ignored Patches

Definition

A patch on a ML is *ignored* if . . .

- . . . the thread of the patch has no responses from persons other than the author
- . . . the patch was not accepted upstream
- . . . all related patches (e.g., revisions in other series) were ignored

Research Question

Are there specific characteristics for ignored patches?
Ignored Patches

By the numbers...

- lore.kernel.org lists 2011-2018: ø2.5%
  - 2011: ø3.9%
  - 2015: ø2.1%
  - 2018: ø1.6%
- vs. ALL mailing lists: (data 2019-05 – today)
  - ø3.3%
Evolution of ignored patches

The List is Our Process!

Ramsauer, Bulwahn, Duda, Mauerer

3. Evaluation
Evolution of ignored patches

<table>
<thead>
<tr>
<th>Date</th>
<th>Number of patches</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td></td>
</tr>
<tr>
<td>2016</td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td></td>
</tr>
<tr>
<td>2019</td>
<td></td>
</tr>
</tbody>
</table>

Evolution of ignored patches
Evolution of ignored patches

Number of ignored patches

Date

Ramsauer, Bulwahn, Duda, Mauerer

The List is Our Process!

September 9, 2019
Evolution of ignored patches

Date

Ratio of ignored patches

1% 2% 3% 4% 5% 6%
Does it matter *when* a patch is sent?

**Insights**

- Largely independent of the development stage
- Slightly higher chance of ignorance during merge window
Towards a formal model of the development process

1. patch is created
2. patch is on mailing list
3. patch is in git repository

- submit patch
- rework patch
- integrate patch

Patch evolution relationship

private
public
Off-list Patches

Definition

An off-list patch is a patch that...

- has been included in Linus’ git repository
- has never been sent to any public mailing list

Results

- Identified 80 commits with PaStA heuristics from v5.1-rc1..v5.1 (≈1800 commits)
- Manually assessed 60 commits and identified 24 o/uniFB00-list patch commits

Ramsauer, Bulwahn, Duda, Mauerer
The List is Our Process!
September 9, 2019
### Definition

An *off-list patch* is a patch that...
- ...has been included in Linus’ git repository
- ...has never been sent to any public mailing list

### Results

- Identified 80 commits with PaStA heuristics from v5.1-rc1..v5.1 (~1800 commits)
- Manually assessed 60 commits and identified 24 off-list patch commits
Off-list Patches

The obvious
- Reverting patches is discussed on mailing list, the revertig patch is not sent.
- Very few patches from maintainers are actually off-list patches

The less obvious
- Some off-list patches are clearly some security-related issues
- Patches from some subsystem maintainers are often off-list
commit c7084edc3f6d67750f50d4183134c4fb5712a5c8
Author: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
Date: Fri Apr 5 15:39:26 2019 +0200

tty: mark Siemens R3964 line discipline as BROKEN

The \texttt{n_r3964} line discipline driver was written in a different time, when
SMP machines were rare, and users were trusted to do the right thing.
Since then, the world has moved on but not this code, it has stayed
rooted in the past with its lovely hand-crafted list structures and
loads of "interesting" race conditions all over the place.

After attempting to clean up most of the issues, I just gave up and am
now marking the driver as BROKEN so that hopefully someone who has this
hardware will show up out of the woodwork (I know you are out there!) and
will help with debugging a raft of changes that I had laying around
for the code, but was too afraid to commit as odds are they would break
things.

Many thanks to Jann and Linus for pointing out the initial problems in
this codebase, as well as many reviews of my attempts to fix the issues.
It was a case of whack-a-mole, and as you can see, the mole won.

Reported-by: Jann Horn <jannh@google.com>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
Signed-off-by: Linus Torvalds <torvalds@linux-foundation.org>
2. Methodology

Towards a formal model of the development process

1. patch is created
2. patch is on mailing list
3. patch is in git repository

Linux Kernel development workflow

Evolution of ignored patches

4. Conclusion
What is useful to measure?

How to improve the process?
5. BACKUP

BACKUP
What fraction of patches correctly addresses MAINTAINERS

Definition
A patch correctly addresses MAINTAINERS if...
- it addresses one ML as listed by get_maintainers.pl
- **OR:** it addresses one maintainers as listed by get_maintainers.pl
What fraction of patches correctly addresses MAINTAINERS

<table>
<thead>
<tr>
<th>Date</th>
<th>Ratio of correctly addressed maintainers</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012</td>
<td>0.80</td>
</tr>
<tr>
<td>2013</td>
<td>0.85</td>
</tr>
<tr>
<td>2014</td>
<td>0.90</td>
</tr>
<tr>
<td>2015</td>
<td>0.95</td>
</tr>
<tr>
<td>2016</td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td></td>
</tr>
<tr>
<td>2019</td>
<td>0.95</td>
</tr>
</tbody>
</table>