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

Embedded Linux Conference Europe, Lyon

October 28, 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 *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 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
Linux Kernel development workflow

2. Methodology
Linux Kernel development workflow

2. Methodology
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

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
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>
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
2. Methodology
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;
+ if (rq->wake_list)
+ return 0;
+ if (!llist_empty(&rq->wake_list))
+ 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

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

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 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);
+ 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

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

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 all 400 tasks to a single CPU. This
was because of select idle sibling()'s interaction with the
[...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>

[...]

#ifdef CONFIG_SMP
cpu_rq(cpu)
return
rq
rq->curr
rq->idle
rq->nr_running
rq->wake_list
struct

[...]
#else
#endif
ifdef
if
CONFIG_SMP
cpu_rq(cpu)
llist_empty
return
rq
rq->curr
rq->idle
rq->nr_running
rq->wake_list
rq->wake_list
struct

return 0;
+ 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

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

diff --git a/kernel/sched.c b/kernel/sched.c
index 1874c74..4cdc91c 100644
−
5138,7 +5138,20 @@EXPORT_SYMBOL(task_nice);
∗
/
−
int idle_cpu(int cpu)
+ 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;
/

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
Interested in the techniques? More details in:

Ramsauer, Duda, Bulwahn, Mauerer
The List is Our Process!
October 28, 2019 10 / 25

2. Methodology

The List is the Process: Reliable Pre-Integration Tracking of Commits on Mailing Lists

Ralf Ramsauer, Daniel Lehmoh, and Wolfgang Mauerer
Technical University of Applied Sciences Regensburg
niffler@tuh.edu, lehmoh@tuh.de, mauei@tuh.de

ABSTRACT

Modifications to open-source software (OSS) are often performed by a large community of developers and maintained by a project management team. In this paper, we present an empirical study of how maintainers of OSS projects monitor the evolution of pull requests (PRs) via Mailing Lists (MLs). Our study is based on a detailed analysis of 76 OSS projects for which we collected PR history, details about PRs, and information about their maintainers. We analyze the information found in mailing lists to study the processes involved in the handling of PRs and their evolution. Our study shows that MLs are an indispensable tool to handle PRs in OSS projects. MLs are also an interesting source for data analysis and can be used to enhance other tools and workflows. Our work contributes to the understanding of how maintainers of OSS projects monitor the evolution of PRs and can be used as a basis for future research on how to improve the efficiency and effectiveness of PR handling processes in OSS projects.

1. INTRODUCTION

Software projects have come a long way since they first emerged as a formal notion in the 1960s. With the advent of large-scale software development, the need for efficient and effective tools for版本 tracking and communication has become more pressing than ever. In this context, Mailing Lists (MLs) have emerged as a valuable tool for software developers to communicate, discuss, and coordinate the development of software projects. MLs provide a platform for developers to collaborate, share ideas, and work together on projects. They serve as a central hub for communication and collaboration, facilitating the flow of information and ideas among team members. MLs play a crucial role in the software development process, enabling developers to work together effectively and efficiently.

In this paper, we present an empirical study of how maintainers of OSS projects monitor the evolution of pull requests (PRs) via Mailing Lists (MLs). Our study is based on a detailed analysis of 76 OSS projects for which we collected PR history, details about PRs, and information about their maintainers. We analyze the information found in mailing lists to study the processes involved in the handling of PRs and their evolution. Our study shows that MLs are an indispensable tool to handle PRs in OSS projects. MLs are also an interesting source for data analysis and can be used to enhance other tools and workflows. Our work contributes to the understanding of how maintainers of OSS projects monitor the evolution of PRs and can be used as a basis for future research on how to improve the efficiency and effectiveness of PR handling processes in OSS projects.

2. APPROACH

Our study is based on a dataset of 76 OSS projects that are maintained on GitHub. We collected data from the projects’ repositories, including PR history, details about PRs, and information about their maintainers. We used a combination of automated and manual methods to extract and analyze the data. For automated data extraction, we used tools to parse the project’s repository and extract relevant information. For manual data analysis, we manually inspected the data and identified patterns and trends.

Our study is based on a dataset of 76 OSS projects that are maintained on GitHub. We collected data from the projects’ repositories, including PR history, details about PRs, and information about their maintainers. We used a combination of automated and manual methods to extract and analyze the data. For automated data extraction, we used tools to parse the project’s repository and extract relevant information. For manual data analysis, we manually inspected the data and identified patterns and trends.

We provide a case study on the open-source software project GNU Emacs, where we demonstrate how maintainers of OSS projects can use Mailing Lists to monitor the evolution of PRs and how these tools can be used to improve the efficiency and effectiveness of PR handling processes in OSS projects.

We provide a case study on the open-source software project GNU Emacs, where we demonstrate how maintainers of OSS projects can use Mailing Lists to monitor the evolution of PRs and how these tools can be used to improve the efficiency and effectiveness of PR handling processes in OSS projects.

We provide a case study on the open-source software project GNU Emacs, where we demonstrate how maintainers of OSS projects can use Mailing Lists to monitor the evolution of PRs and how these tools can be used to improve the efficiency and effectiveness of PR handling processes in OSS projects.

Since this is an ongoing project, we are not ready to share the details of our findings yet. However, we are planning to present our results at a future conference. Our future work will focus on extending and refining our analysis methods to cover a wider range of OSS projects. We will also explore how Mailing Lists can be used to enhance other tools and workflows in the software development process.
Data Acquisition

- Dumps from gmane.org etc.
- kernel.org public inboxes
  - some lists, prehistoric data
  - https://lore.kernel.org/lists.html
  - Some lists are imports from gmane.org :-(
- Our own collection
Data Acquisition

- Dumps from gmane.org etc.
- kernel.org public inboxes
  - some lists, prehistoric data
  - https://lore.kernel.org/lists.html
  - Some lists are imports from gmane.org :-(
- 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
Date: Tue, 27 Mar 22001 13:42:39 +0200 (Westeuropäische Sommerzeit)
X-Mailer: Microsoft Outlook Express 6.00.2900.3028
We analyse...

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

- v2.6.39..linus/master
- ≈610K commits
- ≈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

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
1.15M Patches
2. Methodology

- 3M Mails
  - 1.15M Patches
  - 1.85M Non-Patches

- 1.11M Linux Kernel Patches
  - 1.15M Patches
  - Other

- 1.11M Linux Kernel Patches
2. Methodology

- 3M Mails
  - 1.15M Patches
  - 1.85M Non-Patches

- 1.11M Linux Kernel Patches
  - ~80% Actual Patches
  - 1.15M Patches

- Stable Review
  - Git Pull
  - Bots

Ramsauer, Duda, Bulwahn, Mauerer
The List is Our Process!
October 28, 2019
2. Methodology

3M Mails
- 1.15M Patches
- 1.85M Non-Patches

1.11M Linux Kernel Patches
- ~80% Actual Patches

882K Linux Kernel Patches

Ramsauer, Duda, Bulwahn, Mauerer
The List is Our Process!

October 28, 2019
2. Methodology

3M Mails

1.15M Patches
1.85M Non-Patches

1.11M Linux Kernel Patches

~80% Actual Patches

882K Linux Kernel Patches

794K Relevant Patches

Boots
Stable
Review
Ignored Patches

Research Question

Are there specific characteristics for 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
Ignored Patches

By the Numbers...

lore.kernel.org lists 2011-2018: ø2.5% ignored patches
  - 2011: ø3.9%
  - 2015: ø2.1%
  - 2018: ø1.6%
3. Evaluation

Evolution of ignored patches

![Chart showing the evolution of ignored patches from 2011 to 2019. The chart displays the number of patches per week, categorized as total and ignored, with a clear increase over the years.]

Ramsauer, Duda, Bulwahn, Mauerer

The List is Our Process!

October 28, 2019
Evolution of ignored patches

3. Evaluation

Evolution of ignored patches

<table>
<thead>
<tr>
<th>Date</th>
<th>Number of patches per week</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>10</td>
</tr>
<tr>
<td>2012</td>
<td>30</td>
</tr>
<tr>
<td>2013</td>
<td>100</td>
</tr>
<tr>
<td>2014</td>
<td>300</td>
</tr>
<tr>
<td>2015</td>
<td>1000</td>
</tr>
<tr>
<td>2016</td>
<td>3000</td>
</tr>
<tr>
<td>2017</td>
<td>10000</td>
</tr>
<tr>
<td>2018</td>
<td>30000</td>
</tr>
<tr>
<td>2019</td>
<td>100000</td>
</tr>
</tbody>
</table>

Overall

- Ramsauer, Duda, Bulwahn, Mauerer
- The List is Our Process!
- October 28, 2019
Evolution of ignored patches

Overall

Date

Total number of ignored patches per week

Evolution of ignored patches

Overall
Evolution of ignored patches

<table>
<thead>
<tr>
<th>Date</th>
<th>Ratio of ignored patches</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>1%</td>
</tr>
<tr>
<td>2012</td>
<td>2%</td>
</tr>
<tr>
<td>2013</td>
<td>3%</td>
</tr>
<tr>
<td>2014</td>
<td>4%</td>
</tr>
<tr>
<td>2015</td>
<td>5%</td>
</tr>
<tr>
<td>2016</td>
<td>6%</td>
</tr>
</tbody>
</table>

3. Evaluation
3. Evaluation

Evolution of ignored patches

- linux-arm-kernel@lists.infradead.org
- linux-mips@vger.kernel.org
- linux-wireless@vger.kernel.org
- netdev@vger.kernel.org
Evolution of ignored patches

The List is Our Process!

October 28, 2019 18 / 25
Does it matter *when* a patch is sent?

![Probability of patch ignored distribution by development stage](image.png)

**Insights**
- Largely independent of the development stage
- Slightly higher chance of ignorance during merge window

Distribution of ratio ignored/total patches grouped by Linux kernel development stage.
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

……

private

public

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

- ...has been included in Linus’ git repository
- ...has never been sent to any public mailing list
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 off-list patch commits
Off-list Patches

The obvious

- Reverting patches is discussed on mailing list, the reverting 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 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
   - Rework patch
   - Submit patch

2. Patch is on mailing list
   - Path evolution relationship

3. Patch is in git repository
   - Observable
   - Non-observable

2011-05–2018-12

Evolution of ignored patches

Date
Ratio of ignored patches

3. Evaluation

Evolution of ignored patches
Thank you!

{ralf.ramsauer,wolfgang.mauerer}@othr.de
sebastian.duda@fau.de
lukas.bulwahn@gmail.com