Quantifying Upstream Integration with PaStA

Ralf Ramsauer
Daniel Lohmann, Wolfgang Mauerer

Digitalisation Laboratory,
Technical University of Applied Sciences Regensburg (OTH)

LTSI Workshop @ ELC17
Motivation

- Body of source code provides **basic building blocks**
- Custom **modifications**
- No obligatory **mainline integration**
- Massive **out-of-tree development**
- Diverging software branches
- **Maintenance** of patch stacks requires substantial effort
Quantitative observation of patch stacks

Our Goals

- Actionable criteria for Long-Term maintenance
- Guideline for successful development
- Quantify maintenance effort/costs

Evolution of Patch Stacks

- Track patches
- Living code base
- Identify trouble spots
Quantitative observation of patch stacks

**Our Goals**
- Actionable criteria for **Long-Term maintenance**
- Guideline for **successful development**
- Quantify maintenance effort/costs

**Patch Stack Analysis (PaStA)**
- Identify similar patches on patch stacks
- Measure mainline integration
- Visualise the evolution of patch stacks
- Support development

**Evolution of Patch Stacks**
- Track patches
- Living code base
- Identify trouble spots
Quantitative observation of patch stacks

Our Goals
- Actionable criteria for **Long-Term maintenance**
- Guideline for **successful development**
- Quantify maintenance effort/costs

Patch Stack Analysis (PaStA)
- Identify similar patches on patch stacks
- Measure mainline integration
- Visualise the evolution of patch stacks
- Support development

⇒ mine git repositories!
Vanilla Mainline Linux

Preempt_RT patched Linux
Take mainline project as foundation
Apply releases of the patch stacks on separate branches
Apply releases of the patch stacks on separate branches
Apply releases of the patch stacks on separate branches
Apply releases of the patch stacks on separate branches
Determine similar patches on patch stacks
Determine similar patches on patch stacks
Determine similar patches on patch stacks

Methodology
Determine similar patches on patch stacks
Patches with no successors
Patches with no successors
Patches with no successors: Were they integrated?
Patches with no successors: Were they integrated?
Patches with no successors: Were they integrated?
Patches with no successors: Were they integrated?
Patches with no successors: Were they backported?
How to detect similar patches?
How to detect similar patches?

- Locality Sensitive Hash Functions [?] [x]
- AST based clone detection [?] [?] [x]
- Levenshtein String Distances [✓]
Example of similar patches

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

commit 908a32873728d92df36e0c7cd633046d35e93a8a9
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 iqi 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 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;
+    */
+    #endif
+    return rq->curr == rq->idle && !rq->nr_running;
+  }

  /*
  */

commit 908a32873728d92df36e0c7cd633046d35e93a8a9
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 iqi 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
+    return 1;
  }

  /*
  */
Example of similar patches

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[...]
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 all 400 tasks to a single
cpu. This is because of select_idle_sibling()’s interaction
with the new ipi based wakeup scheme.
Signed-off-by: Thomas Gleixner <tglx@linutronix.de>
Signed-off-by: Peter Zijlstra <a.p.zijlstra@chello.nl>
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.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 1874c74..4cdc91c 100644

On --rt we observed hackbench waking all 400 tasks to a single
cpu. This was because of select_idle_sibling()’s interaction
with the new ipi based wakeup scheme.

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>

Di/uniFB00similarity: 0.875
### PaStA by the Numbers

**Preempt_RT (Jul ‘11 - Aug ‘16):**

- 582 patch stack releases
- 183,000 patches
- 1,098 patch groups
- 156 forward ports
- 219 backports
Case Study Preempt_RT: Evolution of Patch Stacks

Flow of Patches

inflow
- backports and new patches
outflow
- dropped and upstream patches
invariant
- patches that remain on the stack
Case Study Preempt_RT: Mainline Integration

Distribution of integration times (in days)
Positive: forward ports
Negative: backports

Preempt_RT: Duration of mainline integration
Summary

- **Detect similar patches** with semi-automatic algorithm
- Determine **patch flow** between different releases of the patch stack
- Quantify **mainline integration** of a given patch stack
- **Support development** of patch stacks
Future Work

- Measure *invasiveness* of a patch stack
- Use results to *identify patches with high maintenance effort*
- Derive *successful development and maintenance strategies*
Thank you!

ralf.ramsauer@othr.de
Get a fresh clone at github.com/lfd

PGP: 8F10049B