





Analysis and mitigation of writeback cache lock-ups in Linux

Alba Mendez Orero

Project Proposal and Work Plan

Document: workplan.odt	Project Proposal and Work Plan Mitigation of writeback cache lock-ups in Linux	 
Date: 2020-03-01		
Rev: 01		
Page 2 of 12		



REVISION HISTORY AND APPROVAL RECORD

Revision	Date	Purpose
1	2020-03-01	Initial version

DOCUMENT DISTRIBUTION LIST


Name	E-mail
Alba Mendez Orero	me@alba.sh
Juan Jose Costa	jcosta@ac.upc.edu

WRITTEN BY:		REVIEWED AND APPROVED BY:	
Date	2020-03-01	Date	2020-03-04
Name	Alba Mendez Orero	Name	Juan Jose Costa
Position	Project author	Position	Project Supervisor

Document: workplan.odt	Project Proposal and Work Plan Mitigation of writeback cache lock-ups in Linux	 
Date: 2020-03-01		
Rev: 01		
Page 3 of 12		

0. CONTENTS

0.	Contents.....	3
1.	Project overview and goals.....	4
2.	Project background.....	5
3.	Project requirements and specifications.....	6
4.	Work Plan.....	7
4.1.	Work Breakdown Structure.....	7
4.2.	Work Packages, Tasks and Milestones.....	7
4.3.	Time Plan (Gantt diagram).....	8
4.4.	Meeting and communication plan.....	9
5.	Generic skills.....	10



Document: workplan.odt	Project Proposal and Work Plan Mitigation of writeback cache lock-ups in Linux	
Date: 2020-03-01		
Rev: 01		
Page 4 of 12		

1. PROJECT OVERVIEW AND GOALS

As usual in most modern systems, the Linux kernel caches disk writes to optimize throughput and latency. Heavy I/O tasks writing at a fast speed (such as downloads, backups, or extractions) may eventually saturate the cache, forcing the kernel to throttle I/O. This often results in a lagging system. Introducing some level of fairness in the throttling would improve this.

This research project, which is carried out at the Department of Computer Architecture, aims to develop a Proof of Concept that will (partly) introduce this fairness:

- First, the existent (unfair) I/O throttling will be measured. Its impact over system performance and general responsivity will then be analyzed.
- Then, the Proof of Concept will be designed and developed to isolate the throttling, so that only the offending tasks are affected by it.
- As an *optional goal*, a proper patch to the Linux kernel can be developed in order to provide actual fairness.
- Measures will be taken again, and the PoC will be tested on production systems.
- The resulting improvement in system performance / responsivity will be analyzed and general conclusions will be drawn.

Document: workplan.odt	Project Proposal and Work Plan Mitigation of writeback cache lock-ups in Linux	 
Date: 2020-03-01		
Rev: 01		
Page 5 of 12		

2. PROJECT BACKGROUND

2.1. *The problem*

The idea for this independent project comes from several attempts of diagnosing system unresponsiveness from the author, often in production environments. This problem occurs frequently, as the only requisite is to have an application that writes lots of data to the disk.

The writeback cache is always enabled by default, because it greatly improves performance:

- It defers writes to the disk, allowing the application to continue running—in fact, writes are almost instantaneous from the application's point of view.
- Writes are applied to the disk in bulks, allowing the I/O scheduler to work more efficiently.



However, it is also unfair because it treats all writes equally. So, when the cache gets full, *all* applications that perform I/O are allegedly slowed down (I/O throttling), affecting the whole system.

To avoid this problem, an application that wants to write lots of data may explicitly enable `O_DIRECT`, a POSIX flag which causes these writes to bypass the writeback cache, preventing the cache from filling (and therefore, avoiding I/O throttling). However this is far from a solution, because:

- The user must modify the application source code in order to enable it (users don't normally know about this source code).
- Applications don't usually know whether they're going to cause the cache to fill; it's hard to predict if the bottleneck will be at the disk writing speed. It's the kernel that knows.
- Using an option that does an unwanted thing (it disables the cache, reducing performance) just to prevent a system-dependent side effect (I/O throttling) doesn't feel right at all.

Linux also avoids disabling the writeback cache on an entire filesystem, by remounting it in sync mode. However this is also not ideal; it degrades system performance as indicated above.

Linux also allows to adjust the threshold at which the cache is considered full (i.e. the queue size). Threshold, bandwidth and I/O scheduling can also be tweaked per block device. However, neither of this prevents the throttling from triggering; at most, it can isolate it to applications using that block device, but this doesn't usually help.

Document: workplan.odt	Project Proposal and Work Plan Mitigation of writeback cache lock-ups in Linux	 
Date: 2020-03-01		
Rev: 01		
Page 6 of 12		

2.2. *Previous efforts*

This I/O throttling has been a well known source of unresponsiveness. In 2010, modifications were made to the Linux kernel where (among other things) I/O is now *rate-limited* early to avoid reaching the actual limit where full throttling begins. From [patch 143dfe86](#):

[...] long block time in `balance_dirty_pages()` hurts desktop responsiveness [...]

Users will notice that the applications will get throttled once crossing the global (background + dirty)/2=15% threshold, and then balanced around 17.5%. Before patch, the behavior is to just throttle it at 20% dirtyable memory in 1-dd case.



Users will notice a more responsive system during heavy writeback.

"killall dd" will take effect instantly.

Some other changes have also been made in respect to throttling since then; however throttling still has a substantially noticeable effect on responsiveness on the mainline kernel at the time of this writing.

It's important to note that throttling appears to have been task-fair at earlier versions of the Linux kernel; however that doesn't seem to be the case today.

On the other side, subtle control features have been made available (memory cgroups, BIO annotation support), but aren't —to the author's knowledge— being actively used to address I/O throttling fairness. This project attempts to build on these features. However throttling is a complex process, so we first need to understand & measure it in detail.

Document: workplan.odt	Project Proposal and Work Plan Mitigation of writeback cache lock-ups in Linux	 
Date: 2020-03-01		
Rev: 01		
Page 7 of 12		

3. PROJECT REQUIREMENTS AND SPECIFICATIONS

Project requirements:

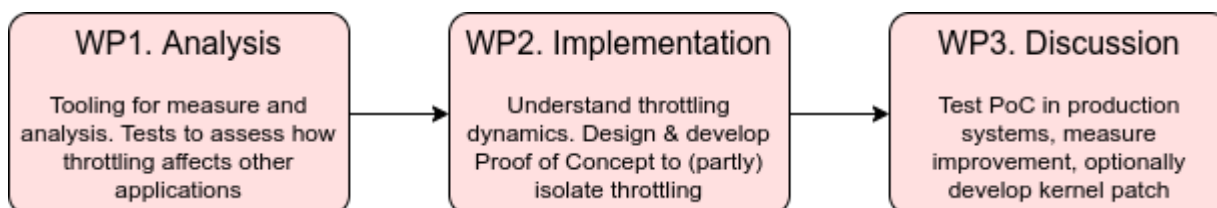
- Tooling to measure and analyze when and how I/O throttling occurs, and what tasks are affected.
- The measuring process should be as less invasive as possible, so it can be used on a production system and measurement itself doesn't alter the results.
- The developed Proof of Concept must be able to isolate the I/O throttling to a small part of the system, ideally at process- or task- level.
- The Proof of Concept should be developed preferably in the form of a *userspace daemon*.
- Optionally, a proper kernel patch may be developed.

Project specifications:

- General specifications (both PoC and measurement tooling):
 - o The kernel must be compiled with accounting and tracing support.
 - o In order to implement fairness, we first need to track the origin of block writes. If these writes come from a filesystem (which will usually be the case), then explicit support is required from the FS in order to track the origin. According to the kernel's documentation, only ext2, ext4 and btrfs have this support implemented. **Implementing tracking support on filesystems is out of scope.** Thus, this isn't guaranteed to work on other filesystems.
- PoC operation specifications:
 - o Maximum time from (a) start of offending writes to (b) throttling isolation: 5 seconds
 - o The kernel must be compiled with cgroup support.
 - o If multiple processes are writing to the same inode, and at least one of them is performing offending writes, throttling isolation is not guaranteed.
 - o Custom kernel patches might be required.

4. WORK PLAN

4.1. Work Breakdown Structure



Critical review initially planned to happen shortly after starting WP2.



Note: As this is a research project, the shcedule / structure may change at a later time.

4.2. Work Packages, Tasks and Milestones

Work Packages:

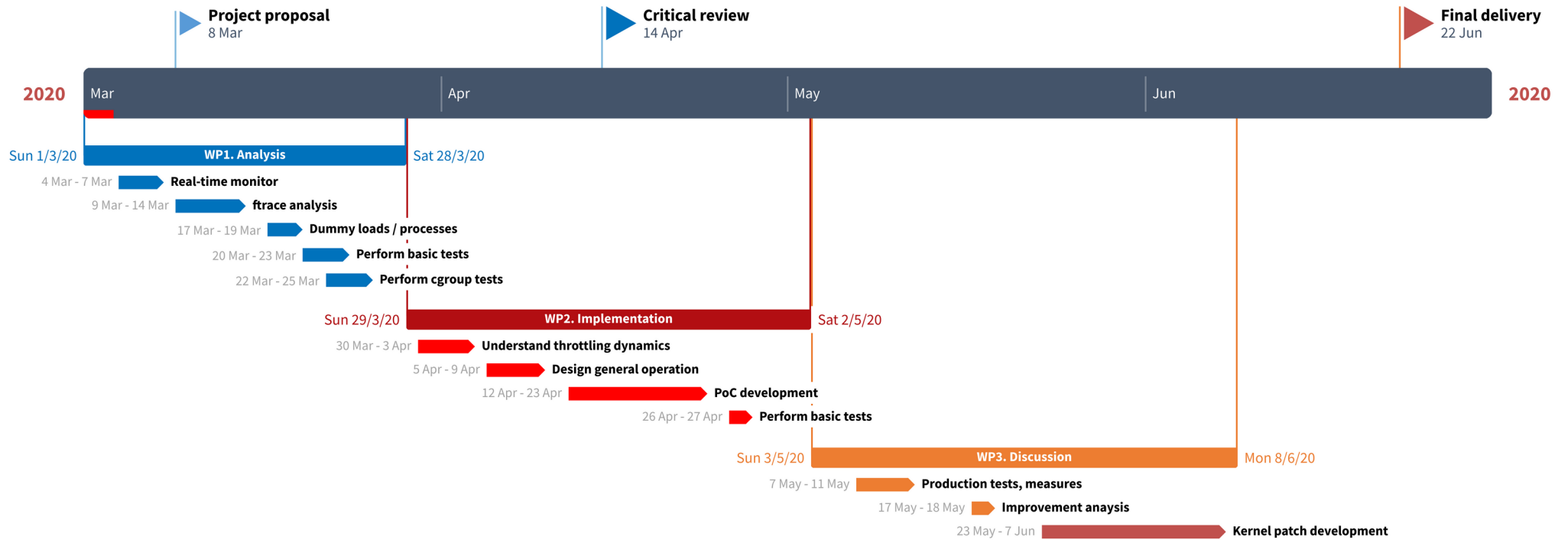
Project: Analysis	WP ref: WP1	
Major constituent: measurement & analysis	Sheet 1 of 1	
Short description: Develop necessary tools to non-invasibly measure and analyze the dynamics of the I/O throttling and how it affects other processes.	Planned start date: 2020-03-01 Planned end date: 2020-03-28	
Internal task T1: real-time monitor Internal task T2: ftrace analysis Internal task T3: dummy loads / processes Internal task T4: perform basic tests Internal task T5: perform cgroup tests	Deliverables: None	Dates: None



Project: Implementation	WP ref: WP2	
Major constituent: design & development	Sheet 1 of 1	
Short description: Understand throttling dynamics. Design & develop Proof of Concept to (partly) isolate throttling	Planned start date: 2020-03-29 Planned end date: 2020-05-02	
Internal task T1: understand throttling dynamics Internal task T2: design general operation, validate it Internal task T3: PoC development Internal task T4: perform basic tests	Deliverables: Critical review	Dates: None

Document: workplan.odt	Project Proposal and Work Plan Mitigation of writeback cache lock-ups in Linux	 
Date: 2020-03-01		
Rev: 01		
Page 9 of 12		

Project: Discussion	WP ref: WP3	
Major constituent:	Sheet 1 of 1	
Short description: Test the PoC in production systems, measure improvement, optionally develop proper kernel patch	Planned start date: 2020-05-03 Planned end date: 2020-06-08	
Internal task T1: production test, measures Internal task T2: improvement analysis Internal task T3: [Optional] kernel patch development	Deliverables: Final memory, source code	Dates: None

4.3. Time Plan (Gantt diagram)



Document: workplan.odt	Project Proposal and Work Plan Mitigation of writeback cache lock-ups in Linux	 
Date: 2020-03-01		
Rev: 01		
Page 11 of 12		

4.4. *Meeting and communication plan*

- Planned meetings with the supervisor:

Meeting	Date
Project Proposal and WorkPlan approval	2020-03-03
Critical Review (midterm)	2020-04-03
Final Review	2020-06-15

In addition to the above, the student and the supervisor plan to meet about every two weeks, to review progress on the project and address any difficulties.

5. GENERIC SKILLS

The following generic skills will be promoted and assessed during the development of the project:
(Mark at least three, being GS4 one of them)

Be aware that if you have some of the third level generic skills not scored yet with A or B, you can work them in your TFG in order to obtain your Bachelor degree with the set of generic skills completely acquired.

#	Generic Skill	Assessed
1	Innovation and entrepreneurship	
2	Societal and environmental context	
3	Communication in a foreign language	
4	Oral and written communication	X
5	Teamwork	
6	Survey of information resources	
7	Autonomous learning	X
8	Ability to identify, formulate and solve engineering problems	X
9	Ability to Conceive, Design, Implement and Operate complex systems in the ICT context	
10	Experimental behaviour and ability to manage instruments	X