ARTICLE IN PRESS

Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Revisiting swapping in mobile systems with SwapBench

Xiao Zhu [a,b], Duo Liu [a,b,*], Liang Liang [c,**], Kan Zhong [a,b], Linbo Long [a,b], Meikang Qiu [d], Zili Shao [e], Edwin H.-M. Sha [a,b]

[a] Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China
[b] College of Computer Science, Chongqing University, Chongqing, China
[c] College of Communication Engineering, Chongqing University, China
[d] Computer Science Department, Pace University, USA
[e] Department of Computing, The Hong Kong Polytechnic University, Hong Kong

HIGHLIGHTS

- An evaluation framework is proposed to appraise different swapping schemes.
- SwapBench is implemented on Android platform and validated with micro benchmark.
- Evaluation of application launch time with different swap schemes is given.
- Evaluation of application switch delay with different swap schemes is given.
- The impact of swap size and application memory access feature is discussed.

ARTICLE INFO

Article history:
Received 16 September 2015
Received in revised form
29 April 2016
Accepted 22 May 2016
Available online xxxx

Keywords:
Swapping
Mobile
User experience
Evaluation

ABSTRACT

Mobile systems such as smartphones and tablets are re-adopting swapping – a mature but rarely used OS feature – to extend memory capacity without adding more DRAM, especially low-end devices. This resurgence of swapping in mobile systems has inspired both traditional "off-the-rack" schemes and new approaches based on compression and new hardware. Their vastly different designs, however, make them difficult for system designers to measure, compare and revise. In this paper, we first propose an evaluation framework, SwapBench, to appraise swap schemes and focus on two important but overlooked metrics: application launch and switch. And cross-validation with microbenchmarks shows that SwapBench is accurate. Then, we present the first comprehensive evaluation from three dimensions: system architecture, application launch time and application switch delays, to understand and summarize the impacts of swapping in mobile systems. Finally, based on the findings from SwapBench, we give our conclusion and suggestions of different approaches to swapping in mobile systems.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Mobile applications – which were lightweight – are becoming increasingly heavy [1], because of their rich functionality made possible by recent high-performance mobile processors and large embedded low power DRAM. Although main memory capacity has been growing in mobile devices,[1] it is difficult (if not impossible) to always satisfy the increasing memory requirements from systems and apps. The situation is worse on low-end devices, which are often seen in developing markets and are usually equipped with 512 MB or smaller main memory for cost reasons. When the system is under memory pressure, running applications will be terminated to make room for new allocations. This design deteriorates user experience but is found in most major mobile OSes, including Android [2,3] and iOS. An important rationale behind this design decision is the lack of a (high-performance) swap space.

Almost all major mobile OSes, including Android and iOS, support or could be easily modified to support swapping, and adding a swap space is an effective (though probably not efficient) solution to reduce unexpected process terminations. Carefully chosen memory pages could be moved out temporarily to stable

* Corresponding author at: College of Computer Science, Chongqing University, Chongqing, China.
** Corresponding author.
E-mail addresses: liuduo@cqu.edu.cn (D. Liu), liangliang@cqu.edu.cn (L. Liang).

[1] For example, the recent Google Nexus 6 smartphone is equipped with 3 GB of DRAM. Specs at http://www.google.com/nexus/6.

http://dx.doi.org/10.1016/j.future.2016.05.026
0167-739X/© 2016 Elsevier B.V. All rights reserved.

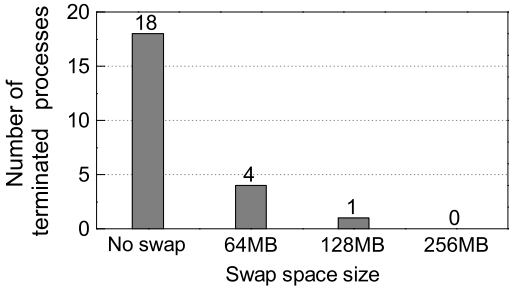

Fig. 1. The number of process terminations with flash-based swapping when running ten popular applications on a Nexus 5 smartphone under memory pressure. Even a small (64 MB) swap space can avoid most terminations.

storage (e.g., flash, RAM disk) to satisfy new allocations when the system is lacking for free memory. As shown in Fig. 1, when running a list of ten popular applications (see Section 4 for details) on a Nexus 5 smartphone,[2] the number of processes terminated under memory pressure drops significantly with a flash-backed swap space.

However, a naive adoption of swapping in mobile systems will cause severe performance degradation due to poor performance of flash memory, especially when the swap space is backed by an SD card [4]. Mainstream systems, including Android and iOS, deliberately avoided swapping – even at the cost of terminating running applications [2,5] – to ensure a smooth user experience.

Despite these warnings on degraded performance of using swapping in mobile systems, there have been active attempts by enthusiasts to enable it manually [6] or using applications [7–10]. On the vendor and academia sides, a recent trend is to customize swapping specifically for mobile systems to have better performance, using compression (e.g., zRAM [3], which is included in recent Android releases) or emerging hardware such as byte-addressable non-volatile memories (NVRAM) [5].

Although the implementation of traditional swapping in mobile OSes are almost identical to that in server/desktop systems, swapping has different and sometimes more profound implications for mobile applications. Moreover, the implementation of recent proposals has diverged a lot from traditional swapping. For example, NVM-Swap [5] completely eliminates I/O and makes swapping pure `memcpy`s; zRAM uses compression to "swap to zRAM", which affects not only performance, but also the device's energy behavior. This makes it more difficult to compare and revise different swapping schemes: I/O numbers are not an accurate metric any more. Without appropriate metrics and tools, the performance characteristics and impacts on system design of these different swapping schemes remain largely unknown to today's mobile system designers.

In this paper, we first propose a comprehensive evaluation framework called SwapBench to appraise swapping schemes. Except for application I/O performance during normal execution, SwapBench focuses on measuring another two fundamental performance metrics that will be affected by swapping the most: application *launch time* and *switch delay*, which measure the time needed to start an application and switch between multiple running applications, respectively. Not only swapping has significant impact on the two metrics, but also they can directly determine the system's "smoothness" when the user interacts with multiple applications. User experience is significantly affected by those factors [11,12]. Besides, in this paper, utilizing SwapBench, we evaluate the difference of application launch time and switch delays with and without all the swap schemes and give a comprehensive analysis based on our extensive experiments.

The main contributions of this paper are listed below:

- We analyze different swapping schemes on architecture-wise and propose a comprehensive evaluation framework called SwapBench to appraise different swapping schemes. With SwapBench, mobile system designers will be able to benchmark, compare and revise existing swap schemes.
- We implement SwapBench on Android platform and validate the accuracy of SwapBench with microbenchmarks. With SwapBench, (1) we explain the relationship between application launch time and different swap schemes and give some analysis; (2) we present the overall impacts on application switch delays of different approaches to swapping in mobile systems and analyze the reason behind.
- Besides, we discuss the factors that would impact application switch delays including size of swap area, application memory access feature (by application category).

In the rest of this paper, we first give an overview of existing swapping schemes in Section 2, and then describe the design of SwapBench in Section 3. Next, we give the methodology of our evaluation in Section 4. Then, we discuss the preliminary results about application launch time drew from SwapBench on evaluating different swapping schemes in Section 5. Then, we present the evaluation and discussion of application switch delays in Section 6. Finally, we give the related work in Section 7 and conclude the paper in Section 8.

2. Swapping in mobile systems

The traditional meaning of "swapping" refers to copying a whole process' memory to some predefined stable storage area known as the "swap area" to free up memory space when the system is under memory pressure [13]. In this way, memory space is "extended" because swapping happens without any application involvement. With virtual memory, however, modern OSes implement swapping in a more flexible way: swap memory pages, instead of the whole process' memory to disk.

Swapping is a mature technique and has been around for server and desktop systems for a long time. But it is relatively rare to see it in mobile systems, due to severe performance overhead caused by flash memory [4,14]. There is a recent trend of re-adopting swapping in mobile devices, made popular by smartphone applications [7–10], popular guide from smartphone forums [6], compression based approaches such as zRAM [3] which is shipped with new Android releases, as well as emerging hardware based schemes such as NVM-Swap [5].

In this section, we first give an overview on different swapping techniques that are available for today's mobile devices, and then discuss their impacts on system architecture. Throughout the paper we use Android as an example to discuss various swapping schemes as it is arguably the most popular mobile OS so far. Our evaluation will also be based on the swapping schemes that are described in this section.

2.1. "Off-the-rack" solutions

Flash based swap. For Linux based systems (e.g., Android based smartphones and tablets), swapping has long been a built-in but disabled OS feature. The only difference between swapping in a mobile device and a server is that usually internal flash storage or external SD card is the only available device to serve as the swap area. Memory pages are moved between DRAM and flash, as shown in Fig. 2(a). The major drawback is that swapping to flash might severely degrade system performance, due to poor write performance found in integrated internal flash, and especially SD cards (which are usually much slower than the internal flash storage) [4]. Besides, swapping to flash might also shorten flash's lifetime.

[2] Specs at http://www.google.com/nexus/5.

Please cite this article in press as: X. Zhu, et al., Revisiting swapping in mobile systems with SwapBench, Future Generation Computer Systems (2016), http://dx.doi.org/10.1016/j.future.2016.05.026

Download English Version:

https://daneshyari.com/en/article/4950378

Download Persian Version:

https://daneshyari.com/article/4950378

Daneshyari.com