vTMM

Paper summary:

The paper proposes a smart design to balance memory pages between fast and slow memory sub-systems in multi-VM co-running virtualized environments. Because stat-of-the-art techniques suffer from severe overheads due to software-based balloning and page table scanning techniques, the paper leverages hardware features for virtualization, notably Intel PML, to overcome these limits.

Strengths:

- The paper focuses on virtualized environment and cloud is nowadays the prevalent execution environment for user applications
- The paper leverages existing hardware features for virtualization

Weaknesses:

- The paper needs to be better motivated, especially the virtualization focus: it is not clear to me in what mean does vTMM specifically differs from the state-of-the-art solutions in the virtualization context (appart from the PML utilization)

Comments to the authors:

++ Motivation

1. Is hybrid memory system a common practice in the cloud? Have you investigated the predominance of such environment with cloud providers?

++ Design

1. Consider increasing the size font in Figure 1 for the smallest strings like "multi-level queue", it is very difficult to read
2. It will be good to number the steps in the design's figure for the reader to have a better overview. In [3.2.1 second paragraph], you talk about an "initialization" step which we don't know where it comes from: is it the VM boot? The vTMM process initialization? etc.
3. In the description, you talk about GPT pages which are, according to my understanding, pages of the (guest) page table. If so, it is not clear to me if you are tracking guest processes' pages or only pages of the PT. Or is it just a mistake?
4. [3.2.3 Intermittent monitoring]: not clear what is the notion of *counter* here, it has not been defined earlier
5. [3.3.1 Page-degree]: *read_count* and *write_count* not defined relatively to the page whose degree is been calculated
6. [3.4 PML-based migration]: doesn't KVM already implement live migration with PML like it is the case with the Xen hypervisor? Or are you the one to introduce it in KVM?
7. [3.5 vTMM memory pool]: how does vTMM behaves when the free remaining memory in the pool is lower than the currently requested memory for the migration round?

++ Evals

1. How is the NUMA CXL simulated?
2. You often invoke pages limitation while I consider they are too many redoundant text throughout the paper; for example, each time you refer to the related work you re-explain what is already presented and explained in section 2.2.
3. [4.3.2 Page tracking overhead]: if I understand Figure 2.b and the text that explains it, does this mean that with a sufficient MWS the stat-of-the-art solution is able to provide same results (or even approximatively closed to) as vTMM? -this is even the case for redis from MWS=600ms in Figure 2.b-. If so, I am not sure to perceive what is the importance of a small MWS: scanning the VM memory more frequently does not incur more degradation?
4. [4.3.3 Multi-level queue]:
    - In Figure 3 is the y-axis the frequency or the cumulated one (CDF)?
    - *We observe that the frequency of hot pages is higher in the distribution when enabling multi-level queue*: you stated in the previous sentences that the two distributions are basically the same, and this is what is even observable in the figure; so why this affirmation?
5. [4.4.2 page migration]: in Table 6, after a migration round, are A/D bits reset? If so, how can the number of VMTraps generated by PML be less than 1? Am I missing something in the experiment explanation?
6. [4.5 Ablation study]: why are the other benchs ommitted here?
7. [4.8 THP support]: I would suggest to put vlines in Table 7 for readability. Similarly in Figure 7, or increase the gap between clusters

++ General Notes

- I would suggest to review the text redoundancy in the document, and save space for more important results in the evaluation or the motivation
- Figures need to be refined: adjust the fonts, the placement in the document (e.g., In Figure 6 the title of bottom figures are attached to the x-axis of top one and it makes it diffucul to detect and read), etc.
- There are many small typos, like duplicate words in sentences or verbs conjugation (e.g., Introduction 7th paragraph: *[...] a shared **a** memory pool.*), but they don't make the text non-understandable anyway. I will not report them all but I suggest the authors proofread the document.
- It may be only on my side, but in all graphs, the gray pattern with bubbles inside does not appear when printing the document on paper

Nephele

Paper summary:

The paper presents a way to porting POSIX applications, such as NGINX or Redis, to unikernel VMs. The alternative proposed by the authors to handle `fork()` syscalls is the cloning of the VM to run the originally `fork`ed process. Nephele, the solution proposed, aims to clone unikernel-based VMs in a way that is transparent to the VMs and that allows IPC communication between the cloned VM and the parent one.

Strengths:

- The paper presents a simple but good idea
- The paper is well written and easy to read and follow

Weaknesses:

- Although the idea is good, it does not present a novel research contribution or effort since it has already been proposed by state-of-the-start solutions such as Kylinx
- The motivation and specification of Nephele should be better enonciated from the Introduction and Problem Statement sections

Comments to the authors:

++ Introduction & Problem Statement

By reading the introduction and problem statement, it is pretty clear what makes the cloning of a unikernel VM different from that of a normal one? I kindly suggest that the authors, from the introduction (and maybe the abstract too), specify that the cloning process takes in place and lieu of the call to `fork()`. This is only perceived when reading the Section 4 that details Nephele.

++ Contribution Nephele

- There is excessive implementation details that do not necessarily ease the comprehension, especially in the description of the Xen internal functioning. What is important is the big picture functioning, otherwise it appears more like an only engineering work or a way to inflate the paper.
- I may have missed it, but what happens when the multi-process at the origin of cloning ends? How do Xen knows about it and how are clones destroyed?
- I understand that to clone a VM, you rely on the domain's creation code, but is this more efficient that than of migration (since it is possible with Xen to perform a localhost migration)? I mean, wouldn't it have provided better cloning times to reuse migration code and bypass the steps of shutting down the parent migrated VM? If not, maybe an experiment in the Evaluation section can clarify this.
- The TCB is quite important (17.5KLOC), I think it may be important to provide a security section that presents the threat model and argues on why the added commands do not present a risk of security or couldn't be leveraged for side-channel attacks.

++ Evaluations

- Figure fonts must be adjusted, the axis and keys are not readable when printed
- In the related work, significant overhead is presented as the limit of Kylinx, so it would have been important to compare to it in the evaluations, since Kylinx is quite similar to Nephele

Aggregate VM

Paper summary:

The paper proposes to aggregate unused resources, of multiples nodes in cloud infrastructures, into a spot VM with the same SLO as Primary VMs.

Strengths:

- The paper presents an idea that can enlarge the applicability and benefit of existing spot VMs

Weaknesses:

- The difference with state-of-the-art is not perceptible
- The paper is, in some paragraphs, difficult to read and follow: too much too long phrases that could be better ponctuated to ease reading

Comments to the authors:

++ Introduction

How does agreggating ressources from multiple machines resolves the problem of eviction? Since the resulting aggreated VM is still a sort of spot VM as its resource is form of unused ones. Moreover, does the decision of evicting the VM not depend on the cloud provider implementation of the spot VM? If so, I am not sure to get how this is related to the design: why does the Aggregate VM's design makes it not evictable compared to existing spot/harvest VMs?

++ Motivation

The conclusion taken from Figure1 on Aggregate VM's workload type is not intuitive. May the auther better explain that Figure?

Light Application Level Virtualization Framework

Paper summary:

The paper proposes *virtualization application*, an alternative mechanism to *virtual machines*. The paper tries exposing host OS to guest applications, so that the latter can run on it and benefit performance of bare-metal execution.

Strengths:

- The idea

Weaknesses:

- The paper writing should be improved as it makes it not easier to read and follow

Comments to the authors:

++ Introduction

- What do you mean by: *the number of VMs is the same as that of the instance, while the container only needs one image*? Which image are you talking about, that of the container/VM? Or that of the hosting OS? Because for each VM/container you have an image, you don't have an image providing many containers right? The sentence is not right clear, nor the information it aims to provide.

++ Background

- 2.3 fiest paragraph: to my understanding, VAPP mainly targets environments with lack of, or limited, virtualization support, such as ARM platform. Yet, what proprtion of such architectures are used by

cloud providers? Maybe a stat on this may help in motivating the contribution
- 2.3 last paragraph: regarding program duplication, it can be reduced without VAPP by storage/memory sharing (i.e., data that may be needed by multiple programs can be placed in a same area that will further be mounted (if disk) or accessed (if RAM) by each program to read the data) and for VMs belonging to the same user. And what is proned by VAPP can also only be valid and fully secure for VMs of the same user

++ Related Work

- For virtualization support to ARM architecture, please consider looking at NEVE, Nested Virtualization Extensions on ARM (published in 2017), that may provide some important background on virtualization support for this architecture

++ Application Virtualization Overview

- Architecture overview: How do you handle multiple VMs communication with the host? In a real environment, there are many VMs and even many applications for the same VM and only a unique host kernel: how do you manage congestion and concurrency? Is VAPP scalable?
- Security:
  - You focused on the security of VMs: what about security of the hypervisor and that of the host kernel?
  - How do the virtual areas, mentionned to mitigate data protection from a VM to another, materialize? Is it disk or RAM space? In both cases, are you not replicating the problem of duplication you are trying to solve.

++ Implementation

- To my understanding, VAPP applies to all processes in the VM right? If so, what is modified from the VM's perspective: is it the applications or the guest kernel? In the introduction you talk about modifications needed in the operating system, which OS is concerned here the host or the guest? Anyway, I guess modifications will be needed at least in the host OS? It is clear while reading.

Typos

- Introduction:
  - 1st paragraph: *Also, it can [...] are not affected the host environments.* => there most be a missing preposition
  - 3rd paragraph:
    - *As the virtual hardware do not* => **does** not
    - *[...] the efficient of the virtual [...]* => **efficiency**
  - 4th paragraph last sentence: *VAPP* is first used here and never defined. By fully reading the introduction one may guess it refers to virtualization application, but you should have defined it before, e.g., at the first occurrence of the term *virtualization application*
- Background:
  - 2.2 paragraphs 2 and 3: *bing* => missing **r**, and *abd* => **and**
- Application Virtualization Overview:
  - 3.1:

    - host userspace component: *an service* => **a** service
    - semi-colon ommitted in the 3 following items
    - virtual machine userspace component: *it convert* => **converts**
- etc. => there are a lot of other minor/major typos and phrase construction mistakes that make it difficult to read and understand. I will not report them all here, please think proofreading the paper.

GENESIS: A Generalizable, Efficient, and Secure Intra-kernel Privilege Separation

Paper summary:

The paper leverages a HW feature that is equivalent in most processors to elaborate on a design for intra kernel isolation that is not specific to the architecture. This feature is named SMAP (Supervisor-Mode Access Prevention) in x86, PAN (Privileged Access Never) in ARM, and SUM (permit Supervisor User Memory) in RISC-V.

Strengths:

- The paper is very well structured with a Threat Model and discusion sections
- The context and the problem are well stated and introduced
- The paper is very well written and easy to follow

Weaknesses:

-

Comments to the authors:

++ Genesis Design

- In the *Kernel Deprivileging* Section, it is stated that Genesis ensures the integrity of the CR0 and CR4 registers because CR0.PG and CR4.WP bits can be cleared by the outer kernel: so this means that the outer kernel however has access to those registers and can intend to modify them at anytime? If then, does Genesis peridically checks over those bits to ensure their integrity? What if the checking is done while the outer kernel has already compromised the Genesis security?

++ Perf Eval

- In the LMBench section, you talk about your version of PerspicuOS. What is this version about? I may have missed it, but it seems it is never mensionned that you customized PerspicuOS. If you did, what are improvements or modifications brought to the native version?

General Comments

- I think the paper could be good candidate for ISCA maybe? Seen it has a lot HW/ISA considerations.

Metropolis: OS Support for Container FaaS Snapshots

Paper summary:

The paper proposes Metropolis, a system for checkpointing container to reduce function invocation in FaaS' clouds.

Strengths:

- The paper treats a hot topic in the cloud which is FaaS

Weaknesses:

- The introduction is difficult to follow and understand. The problem is clearly stated

Comments to the authors:

++ Introduction

- It is difficult to get what is the limitation of CRIU you are trying to expose, what you are comparing to, what improvement you bring to Aurora

++ Design

- The design description is unclear, the components of Metropolis are never defined nor explained in depth. Only the summary execution flow of Metropolis is repeated through out the document
- Figure 2 is never referenced in the paper and has no legend to be understood

++ Evaluation

- In-memory invocation Times: the explains that, the take-away from Table 2 is that Metropolis is not bottlenecked by the restore mechanism, while the first part of the table, that represents the restore path (according to paragraph 3), is the one presenting the higher overhead.
- Figure 5: colors used (deep red and green) are not easily distinguishable

General Comments

- The paper needs to be proofread, some mistakes in some phrases completely make them non-understandablev

DARC: High-dimensional Diffusing Anomaly Detection and Root Cause Location in Cloud Computing Systems ### Paper summary:

The paper proposes a system to detect the root cause of anomalies in cloud systems. Compared to state-of-the-art, the paper builds a system that can use a large range and number of attributes combinaison.

Strengths:

- The paper addresses a real-word and concrete problem in cloud systems
- The proposed system is tested and evaluated on real cloud

Weaknesses:

-

Comments to the authors: