Closer: A new design principle for the privileged virtual machine OS

Djob Mvondo, LIG, University of Grenoble Alpes
Dr. Boris Teabe, IRIT, University of Toulouse
Pr. Alain Tchana, AVALON, ENS Lyon
Pr. Daniel Hagimont, IRIT, University of Toulouse
Pr. Noel De Palma, LIG, University of Grenoble Alpes

MASCOTS'19, 22-10-19, Rennes

Outline

Context

System Virtualization

System virtualization enables several operating systems (OSes) to run on a physical server. The **hypervisor or Virtual Machine Monitor** must ensure :

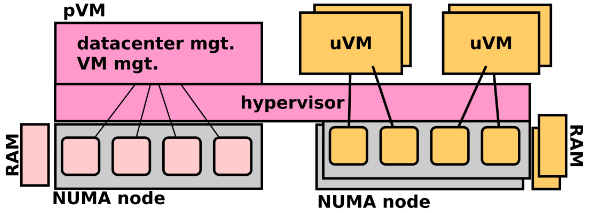
- Resource sharing
- Isolation
- Partitioning

Context

System Virtualization - Hypervisor

In most hypervisors, to minimise the TCB, the trend is to grant a set of privileges to a particular VM, refered to as **privileged VM (pVM)**. It is responsible for :

- VM management tasks i.e *start, stop, checkpointing, etc ...*
- Multiplexing I/O devices to VMs (*split driver model*)[2]
- Hosting monitoring tools e.g., OpenStack Nova Compute



[2]depends on the virtualization type

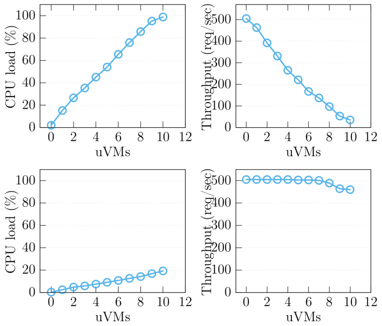
Problem statement

Resource provisioning

Delegating those tasks to a pVM raises an important concern :

How to provision resources to the pVM, especially on NUMA machines ?

Sizing (cpu+memory), NUMA placement, Billing to User VMs



Problem statement

Resource provisioning

Delegating those tasks to a pVM raises an important concern :

How to provision resources to the pVM, especially on NUMA machines ?

Sizing (cpu+memory), **NUMA placement**, Billing to User VMs

	Latency(ms)	**Throughput (Reqs/sec)**
Co-localised	8.621	521
No colocation	13.431	491

Problem statement

Resource provisioning

Delegating those tasks to a pVM raises an important concern :

How to provision resources to the pVM, especially on NUMA machines ?

Sizing (cpu+memory), NUMA placement, **Billing to User VMs**

Related Work

- Oracle VM Server proprosed a static solution for the memory allocation of its pVM. $pVM_{mem} = 502 + int(phys_{mem} \times 0.0205)$[1].
- D. Gupta et al. Middleware'06 [2] tries to account CPU time used by the pVM on behalf of VMs.
- B. Teabe. et al. SCC'16 [3] tries to quantify and charge on VMs resources used by the pVM on their behalf.
- Existing approaches either propose static solutions or fail to handle NUMA.

[1] Oracle Docs. Changing the dom0 memory size. `http://tiny.cc/lckuez`

[2] Enforcing performance isolation across virtual machines in Xen

[3] Billing the CPU Time Used by System Components on Behalf of VMs

Related Work

- Oracle VM Server proprosed a static solution for the memory allocation of its pVM. $pVM_{mem} = 502 + int(phys_{mem} \times 0.0205)$[1].
- D. Gupta et al. Middleware'06 [2] tries to account CPU time used by the pVM on behalf of VMs.
- B. Teabe. et al. SCC'16 [3] tries to quantify and charge on VMs resources used by the pVM on their behalf.
- Existing approaches either propose static solutions or fail to handle NUMA.

The main issue with current pVMs is that they rely on standard Oses (e.g., Linux) whose resource manager does not consider pVM's particularities (correlation with uVMs).

[1] Oracle Docs. Changing the dom0 memory size. `http://tiny.cc/lckuez`
[2] Enforcing performance isolation across virtual machines in Xen
[3] Billing the CPU Time Used by System Components on Behalf of VMs

Contributions

Closer principle

To mitigate these issues, we introduce a new principle **Closer** which ensures three main rules :

- **On-demand** : pVM's Resources should be allocated on demand according to uVMs activities thus avoiding resource waste by a static allocation.

Contributions

Closer principle

To mitigate these issues, we introduce a new principle **Closer** which ensures three main rules :

- **On-demand** : pVM's Resources should be allocated on demand according to uVMs activities thus avoiding resource waste by a static allocation.
- **Pay-per-use** : The resources allocated to a pVM's task which is correlated with a uVM activity should be charged to the concerned uVM thus reducing unpredictability and DDos attacks.

Contributions

Closer principle

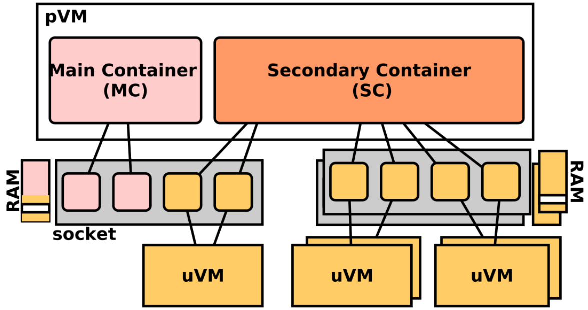
To mitigate these issues, we introduce a new principle **Closer** which ensures three main rules :

- **On-demand** : pVM's Resources should be allocated on demand according to uVMs activities thus avoiding resource waste by a static allocation.
- **Pay-per-use** : The resources allocated to a pVM's task which is correlated with a uVM activity should be charged to the concerned uVM thus reducing unpredictability and DDos attacks.
- **Locality** : The resources allocated to a pVM's task which is correlated with a uVM activity should be located as close as possible (i.e. on the same NUMA socket) to the uVM activity thus improving uVM's performance.

Contributions

pVM redesign

Conceptually, we divide the pVM in a **main container (MC)** and a **secondary container (SC)**. The MC is in charge of administrative tasks and the SC concerns all uVMs related activities.



Contributions

pVM redesign - MC Resource Management

Since the resources needed by the MC do not depend on the uVMs intensity, they are constant and can be estimated as they only depend on the DataCenter administration system (e.g., OpenStack). We carried out evaluations for three datacenter management frameworks.

	#vCPus	RAM(Gb)
OpenStack	2	2
OpenNebula	2	1.85
Eucalyptus	1	1.5

Contributions

pVM redesign - SC Resource Management

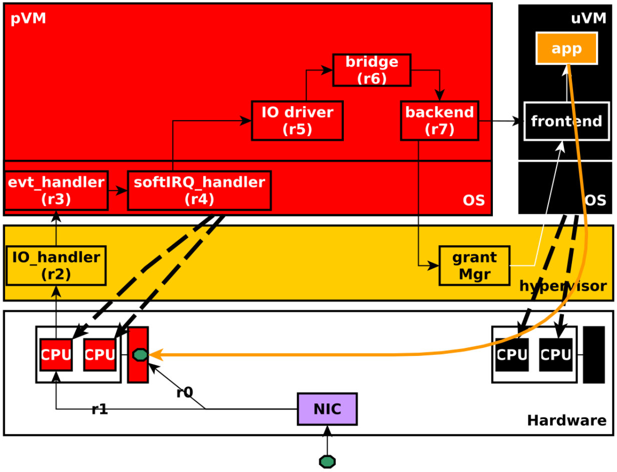
At boot, the SC has as many vCPUs as CPUs on the system (excluding those of the MC) and each vCPU is mapped to one CPU. For each I/O task **T** needs to be scheduled (on behalf of a **uVM**), we apply the following algorithm:

```
  Input  : T: an I/O task or VM administration task that
           should run within pVM
1 targetuVM=Identification of the target uVM
2 targetCPU=Pick a CPU which runs targetuVM
3 Schedule T on the pVM's vCPU which is pinned on
    targetCPU
```

By doing so, each pVM task **T** for a uVM **U** is charged on U and we benefit from the node/socket locality.

Contributions

SC Resource Management - Packet Reception



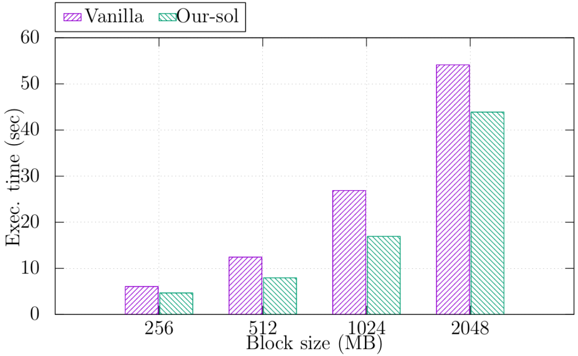
Performance Evaluation

We prototyped our solution on **Xen 4.7** with **Linux kernel 4.10.3**. Our evaluation testbed has the following characteristics :

Cores	26 x1,7 GHz
NUMA sockets	2, equidistant to the NIC & disk
Memory	65 GB
Ethernet	Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet
pVM & uVM's OS	Ubuntu Server 16.04 , Linux Kernel 4.10.3

Performance Evaluation

Disk I/O Improvement, up to 22% improvement
dd performance with varying block sizes
We compare our solution with native Xen



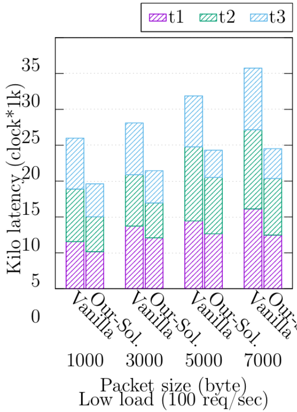
Performance Evaluation

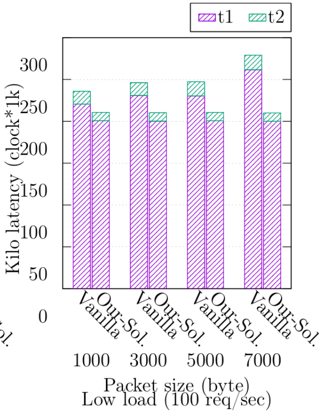
Network I/O Improvement, up to 36.5% improvement

t_1 : the treatment duration before the backend invocation,

t_2: the time taken by the backend to inform the frontend,

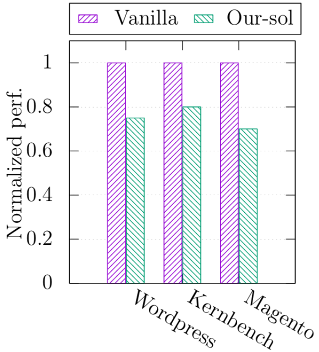
t_3 : the time taken by the frontend to transfer the packet to the application's buffer





Performance Evaluation

Macro-Benchmarks, up to 25% improvement
25% for wordpress, 21% for Kernbench, and 30% for Magento



For more evaluation results, please refer to the paper

Conclusion

We present **Closer** a principle which tries to bridge the gap between standard pVM OSes and the correlation with uVMs. It ensures three rules:

- **On-demand**
- **Pay-per-use**
- **Locality**

The results obtained proves we can have **optimial resource provisioning** and **improved performance** with little overhead both for administrative tasks (such as migration) or uVMs apps.

Conclusion

We present **Closer** a principle which tries to bridge the gap between standard pVM OSes and the correlation with uVMs. It ensures three rules:

- **On-demand**
- **Pay-per-use**
- **Locality**

The results obtained proves we can have **optimial resource provisioning** and **improved performance** with little overhead both for administrative tasks (such as migration) or uVMs apps.

Thank you for your attention ☺