The Art of CPU-Pinning: Evaluating and Improving the Performance of Virtualization and Containerization Platforms

Davood GhatrehSamani, Chavit Denninart
Joseph Bacik†
Mohsen Amini Salehi‡
High Performance Cloud Computing Lab (HPCC)
School of Computing and Informatics
University of Louisiana Lafayette

UNIVERSITY of LOUISIANA LAFAYETTE

HPCC lab.

Introduction

- Execution platforms
  1. Bare-Metal (BM)
  2. Hardware Virtualization (VM)
  3. OS Virtualization (containers, CN)
- Choosing a proper execution platform, based on the imposed overhead
  - Container on top of VM (VMCN) is not studied and compared to other platforms in depth

Introduction

- Overhead behavior and trend
  - Different execution platforms (BM, VM, CN, VMCN)
  - Different workload patterns (CPU intensive, IO intensive, etc.)
  - Increasing compute resources
  - Compute tuning applied (CPU pinning)
- Cloud solution architect challenge:
  - Which Execution platform suits what kind of workload

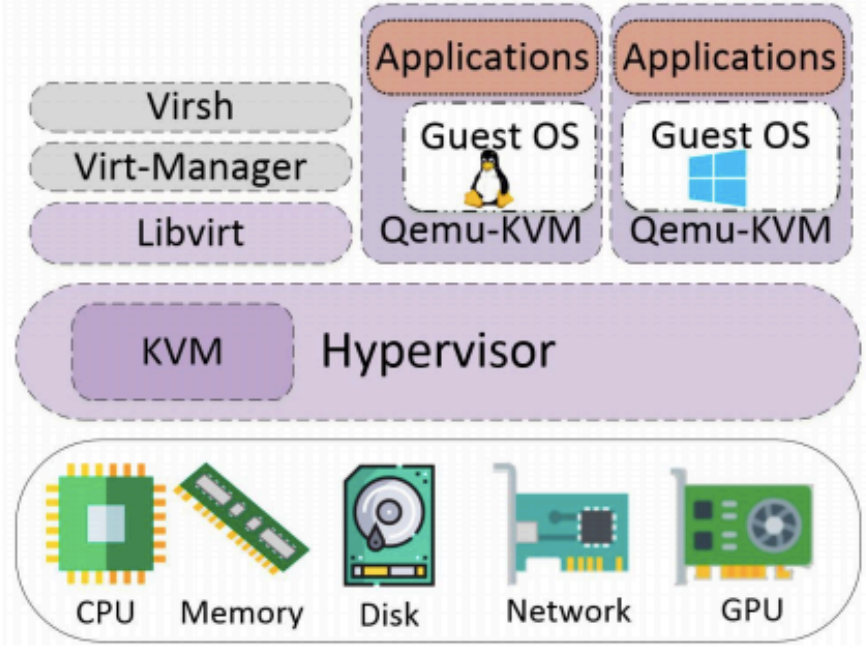
Hardware Virtualization (VM)

- Operates based on a hypervisor
- VM: a process running in the hypervisor
- Hypervisor has no visibility to VM's processes
- KVM: a popular open-source hypervisor
  AWS Nitro C5 VM type

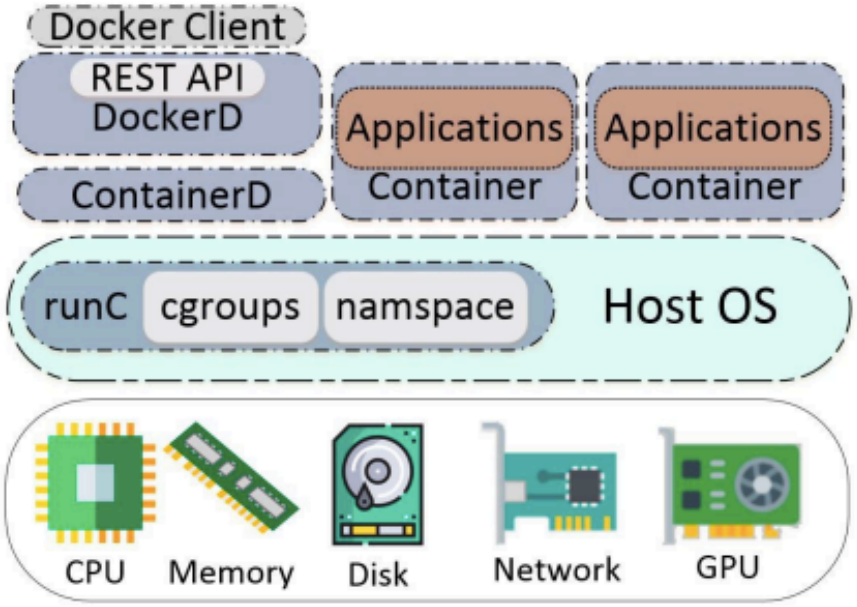
OS Virtualization (Container)

- Lightweight OS layer virtualization
- No resource abstraction (CPU, Memory, etc.)
- Host OS has complete visibility to the container processes
- Container = name space + cgroups
- Docker: The most widely adopted container technology

VM vs Container



(a) Modules of KVM hypervisor. Each VM, called Qemu-KVM, has a full-stack of the deployed applications and an operating system. Libvirt provides necessary APIs for managing KVM.

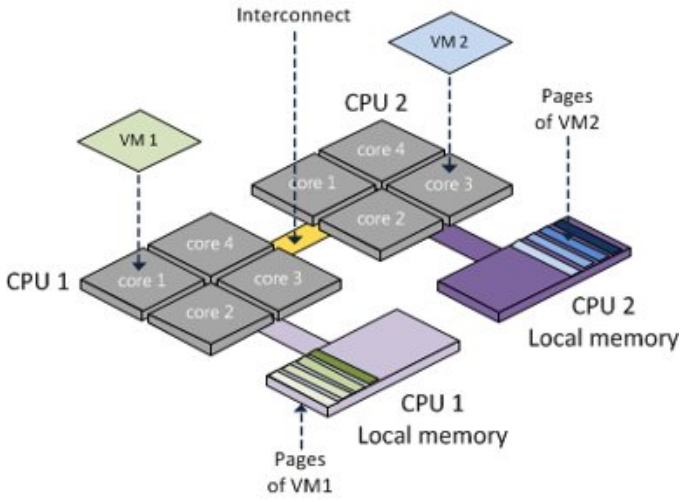


(b) Main modules of Docker. Containers are coupling of namespace and cgroups modules of the host OS kernel. Docker daemon interacts with Container daemon (ContainerD) and runC kernel module to manage containers.

HPCO lab.

CPU Provisioning in Virtualized Platform

- Default: Time sharing
  - Linux: Completely Fair Scheduler (CFS)
  - All CPU cores are utilized even if there is only one VM in the host or the workload is not heavy
  - Each CPU quantum different set of CPU cores
  - Called Vanilla mode in this study
- Pinned: Fixed set of CPU cores for all quantum
  - Override default host/hypervisor OS scheduler
  - Process is distributed only among those designated CPU cores



HPCO lab.

Execution Platforms

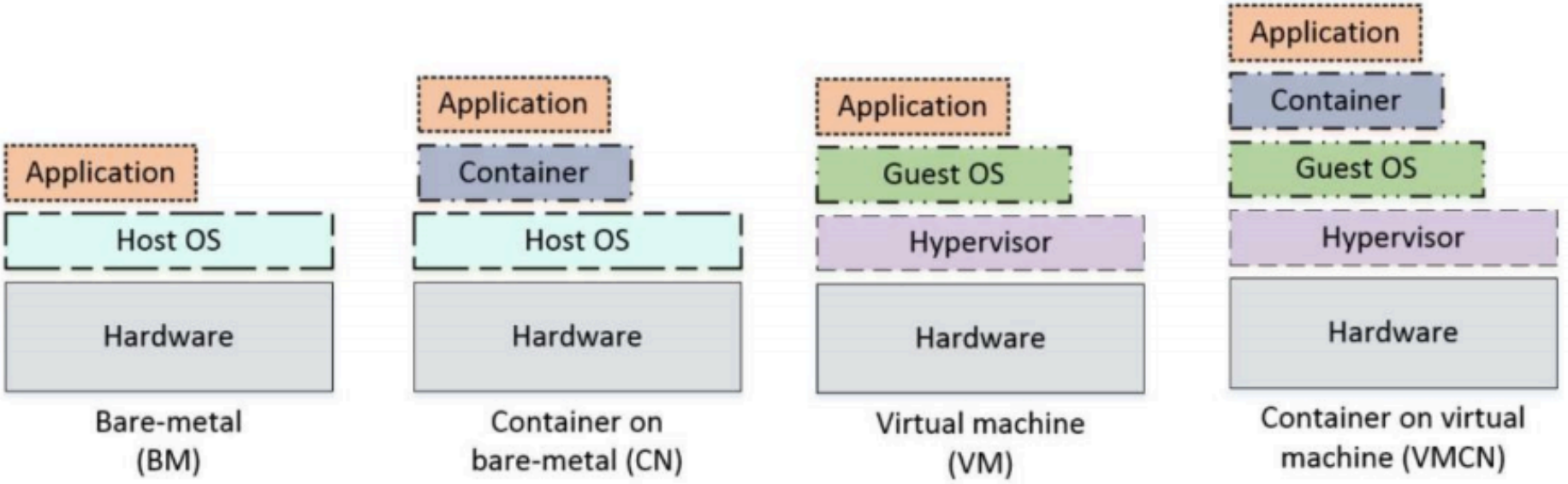


Fig. 2. The four execution platforms used for performance evaluation of different application types.

Abbr.	Platform	Specifications
BM	Bare-Metal	Ubuntu 18.04.3, Kernel 5.4.5
VM	Virtual Machine	Qemu 2.11.1, Libvirt 4 Ubuntu 18.04.3, Kernel 5.4.5
CN	Container on Bare-Metal	Docker 19.03.6, Ubuntu 18.04 image
VMCN	Container on VM	As above

Table 3. Characteristics of different execution platforms used in the evaluations. First column shows the abbreviation of the execution platform used henceforth in the paper.

HPCC lab.

Application types and measurements

Type	Version	Characteristic
FFmpeg	3.4.6	CPU-bound workload
Open MPI	2.1.1	HPC workload
WordPress	5.3.2	IO-bound web-based workload
Cassandra	2.2	Big Data (NoSQL) workload

Table 1. Specifications of application types used for evaluation.

- Measured performance metric
  - Total Execution Time
- Overhead ratio
  - $\frac{Average\ execution\ time\ offered\ by\ the\ platform}{Average\ execution\ time\ of\ bare-metal}$
- Performance monitoring and profiling tools
  - BCC (BPF Compiler Collection: cpudist, offcputime), iostat, perf, htop, top

Configuration of instance types

Instance Type	No. of Cores	Memory (GB)
Large	2	8
xLarge	4	16
2xLarge	8	32
4xLarge	16	64
8xLarge	32	128
16xLarge	64	256

Table 2. List of instance types used for evaluation.

- Host server: DELL PowerEdge R830
  - 4×Intel Xeon E5-4628Lv4
    - Each processor is 1.80, 35 MB cache and 14 processing cores (28 threads)
    - 112 homogeneous cores
  - 384 GB memory
    - 24×16 GB DDR4 DRAM
  - RAID1 (2×900 GB HDD 10k) storage.

Motivation

- In depth study of container on top of VM (VMCN)
- Comparing different execution platforms (BM, VM, CN, VMCN) all to gather
- Real life applications with different workload patterns
- Finding an overhead trend by increasing resource configurations
- Involving CPU pinning in the evaluation

Contribution to this work

- Unveiling
  - PSO (Platform Size Overhead)
  - CHR (Container to Host core Ratio)
- Leverage PSO and CHR to define overhead behavior pattern for
  - Different resource configurations
  - Different workload types
- A set of best practices for cloud solution architects
  - Which execution platform suits what kind of workload

HPCC lab.

Experiment and analysis: Video Processing Workload Using FFmpeg

- FFmpeg: Widely used video transcoder
  - Very high processing demand
  - Multithreaded (up to 16 cores)
  - Small memory footprint
  - Representative of a CPU-intensive workload
- Workload:
  - Function: codec change from AVC (H.264) to HEVC (H.265)
  - Source video file: 30 MB HD video
  - Mean and confidence interval across 20 times of execution for each platform collected

HPCO lab.

Experiment and analysis: Video Processing Workload Using FFmpeg

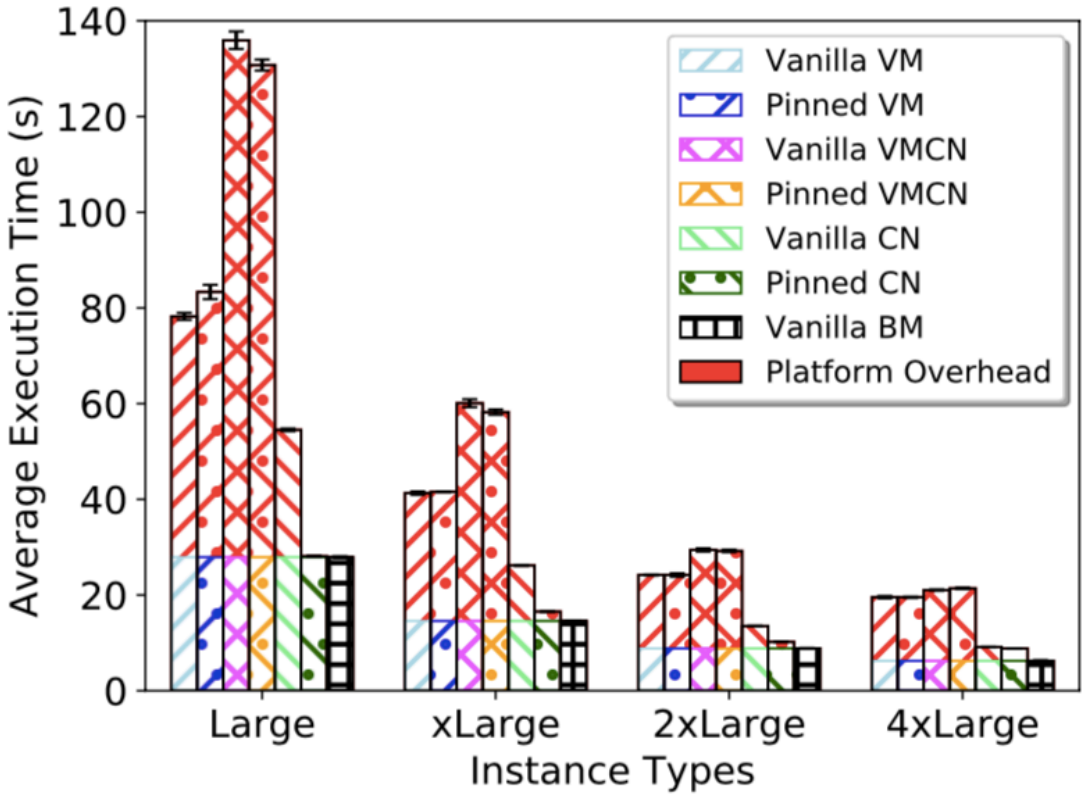


Fig. 3. Comparing execution time of FFmpeg on different execution platforms under varying number of CPU cores. Horizontal axis indicates the number of CPU cores in form of different instance types.

HPCC lab.

Experiment and analysis: Parallel Processing Workload Using MPI

- MPI: Widely-used HPC platform
  - Multi-threaded
  - Resource usage footprint highly depends on the MPI program
- Workload
  - Applications: MPI_Search, Prime_MPI
  - Compute intensive, however, communication between CPU cores dominates the computation
  - Mean and confidence interval across 20 times of execution for each platform collected

HPCC lab.

Experiment and analysis: Parallel Processing Workload Using MPI

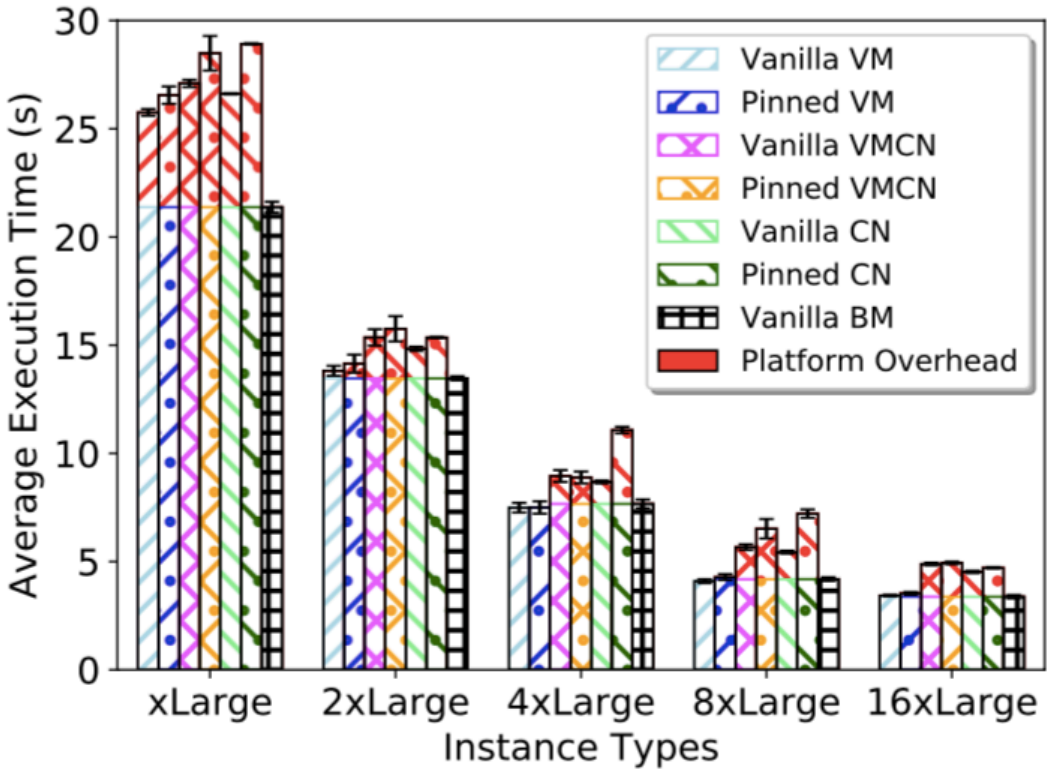


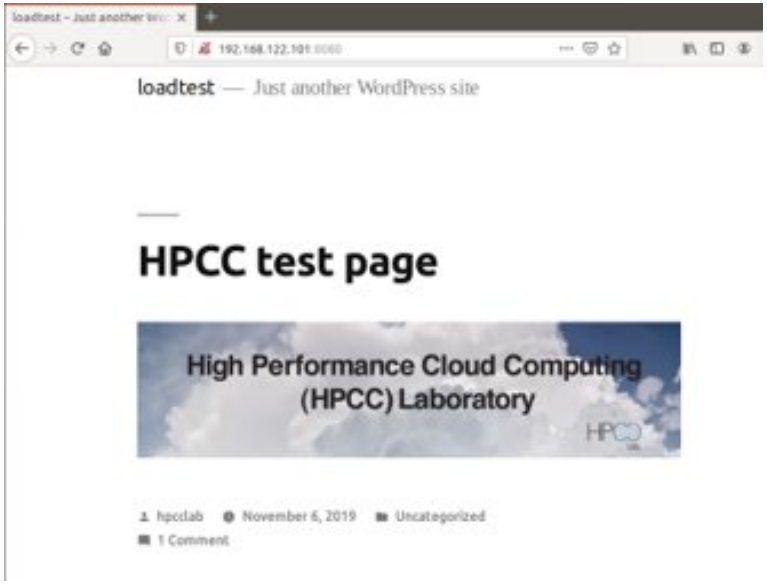
Fig. 4. Comparing execution time of MPI_search on different execution platforms. Horizontal axis represents the number of CPU cores in the form of different instance types. Vertical axis shows the mean execution time (in seconds).

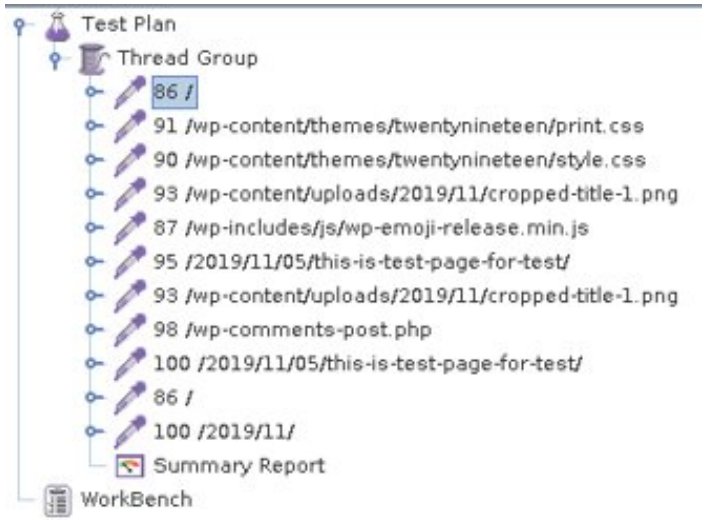
HPC lab.

Experiment and analysis: Web-based Workload Using WordPress

- WordPress
  - PHP-based CMS: Apache Web Server+MySQL
  - IO intensive (network and disk interrupts)
- Workload
  - A simple website is setup on WordPress
  - Browsing behavior of a web user is recorded
  - 1,000 simultaneous web users are simulated
    - Apache Jmeter
  - Each experiment is performed 6 times
  - Mean execution time (response time) of these web processes is recorded

Experiment and analysis: Web-based Workload Using WordPress





Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Through...	KB/sec	Avg. Byte
86 /	1801	10862	122	68454	17129.33	1.00%	22.2/sec	93.15	4295.
91 /wp-c...	852	11418	1	66424	20092.81	0.59%	11.7/sec	17.65	1549.
90 /wp-c...	820	2570	14	56116	7744.90	0.00%	12.0/sec	357.79	30508.
93 /wp-c...	1640	1729	1	55801	3102.94	0.06%	23.1/sec	4808.51	213611.
87 /wp-in...	820	32	1	1022	48.84	0.00%	12.2/sec	59.28	4973.
95 /2019...	820	3954	1163	35273	3164.17	0.00%	11.7/sec	68.53	6017.
98 /wp-c...	820	846	161	2703	306.66	100.00%	12.1/sec	37.21	3159.
100 /201...	820	2389	1132	3934	391.55	0.00%	11.9/sec	71.78	6156.
100 /201...	820	3659	69	34834	2192.79	0.00%	12.7/sec	76.09	6157.
TOTAL	9213	4684	1	68454	10979.14	9.16%	113.4/sec	4880.34	44078.

HPCC lab.

Experiment and analysis: Web-based Workload Using WordPress

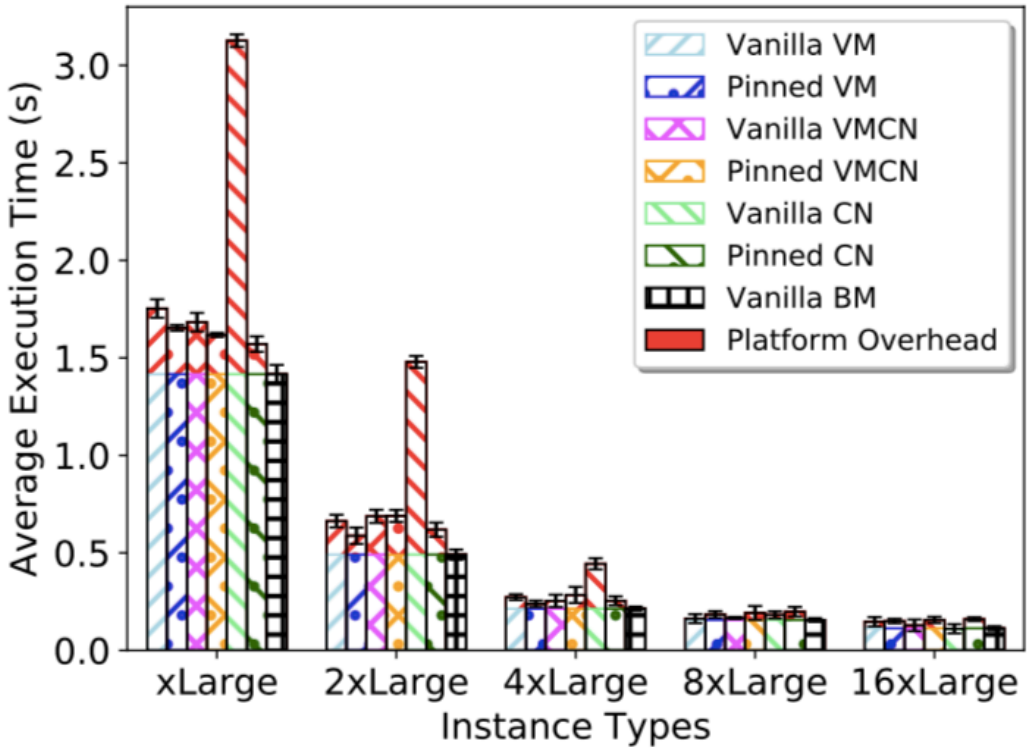


Fig. 5. Comparing mean response time (aka execution time) of 1,000 web processes on different execution platforms (WordPress evaluation). The horizontal axis represents the number of CPU cores in the form of different instance types and the vertical axis shows the mean execution time (in seconds).

HPCC lab.

NoSQL Workload using Apache Cassandra

- Apache Cassandra:
  - Distributed NoSQL, Big Data platform
  - Demands compute, memory, and disk IO.
- Workload
  - 1,000 operations within one second
    - Cassandra-stress
    - 25% Write, 85% Read
    - 100 threads, each one simulating one user
  - Each experiment is repeated 20 times
  - Average execution time (response time) of all the synthesized operations.

HPCC lab.

NoSQL Workload using Apache Cassandra

```
Results:
Op rate                   :    9,995 op/s  [READ: 7,513 op/s, WRITE: 2,485 op/s]
Partition rate            :    9,995 pk/s  [READ: 7,513 pk/s, WRITE: 2,485 pk/s]
Row rate                  :    9,995 row/s [READ: 7,513 row/s, WRITE: 2,485 row/s]
Latency mean              :    1.1 ms [READ: 0.9 ms, WRITE: 1.6 ms]
Latency median            :    0.6 ms [READ: 0.6 ms, WRITE: 0.6 ms]
Latency 95th percentile   :    1.1 ms [READ: 1.0 ms, WRITE: 2.6 ms]
Latency 99th percentile   :   15.6 ms [READ: 5.2 ms, WRITE: 29.8 ms]
Latency 99.9th percentile :   62.1 ms [READ: 51.9 ms, WRITE: 69.1 ms]
Latency max               :   81.1 ms [READ: 75.3 ms, WRITE: 81.1 ms]
Total partitions          :    558,733 [READ: 419,816, WRITE: 138,917]
Total errors              :          0 [READ: 0, WRITE: 0]
Total GC count            : 5
Total GC memory           : 7.972 GiB
Total GC time             :    0.1 seconds
Avg GC time               :   26.6 ms
StdDev GC time            :    3.5 ms
Total operation time      : 00:00:55

END
```


HPC lab.

NoSQL Workload using Apache Cassandra

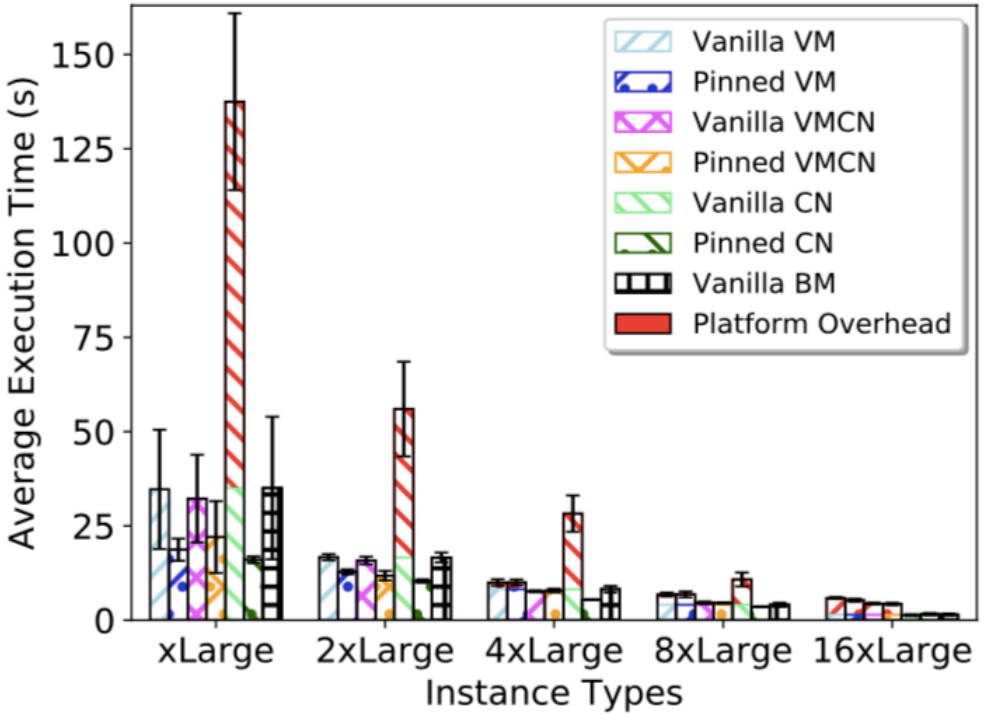


Fig. 6. Comparing mean execution time (aka response time) of Cassandra workload (in seconds) on different execution platforms. Horizontal axis represents the number of CPU cores in the form of different instance types. Note that the execution time for the Large instance type is out of range and unchartable.

HPC lab.

Cross-Application Overhead Analysis

- Platform-Type Overhead (PTO)
  - Resource abstraction (VM)
  - Constant trend
  - Pinning is no helpful
- Platform-Size Overhead (PSO)
  - Diminished by increasing the number of CPU cores
  - Specific to containers
  - Just reported by IBM for Docker (Websphere tuning)
  - Pinning is helpful alot

HPCO lab.

Parameters affecting PSO

1. Container Resource Usage Tracking
   - cgroups
2. Container-to-Host Core Ratio (CHR)
   - $\text{CHR} = \frac{\text{Assigned cores to the container}}{\text{Total number of host cores}}$
3. IO Operations
4. Multitasking

HPCO lab.

Impact of CHR on PSO

- Lower value of CHR imposes a larger overhead (PSO)
- Application characteristics define the value of CHR
  - CPU intensive
    - $0.14 < CHR < 0.28$
  - IO intensive: higher
    - $0.28 < CHR < 0.57$

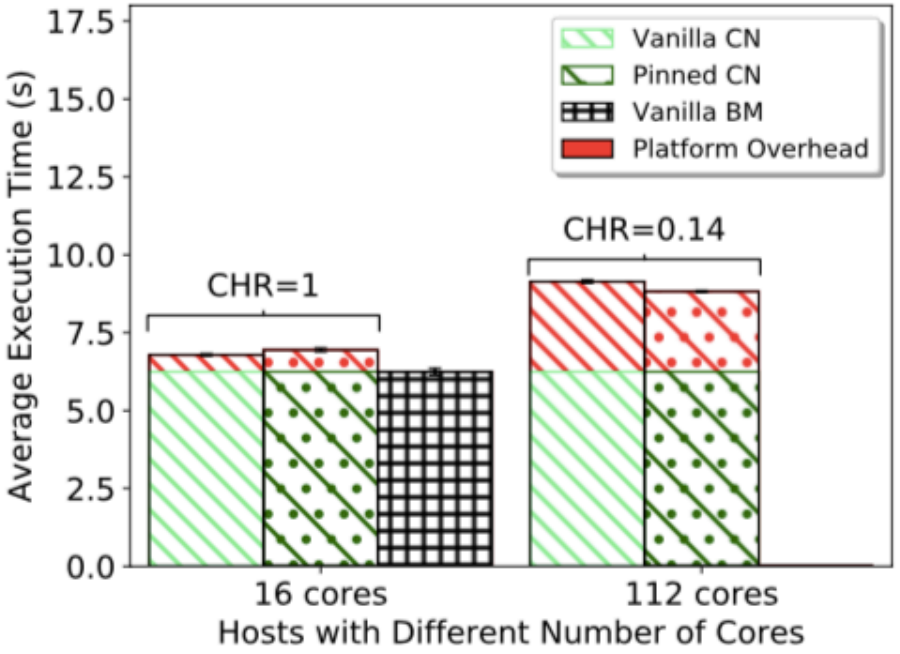


Fig. 7. Evaluating the impact of CHR on the overhead of a vanilla and a pinned CN platform on two homogeneous hosts with 16 and 112 cores. The vertical axis shows the mean execution time (in seconds) and the horizontal axis shows the host's number of cores.

HPCO lab.

Container Resource Usage Tracking

- OS scheduler allocates all available CPU cores to the CN process
- cgroups collects usages cumulatively
- Each scheduling event has different CPU allocation for that CN
- cgroups is an atomic (kernel space) process
- Container has to be suspended while aggregating resource
- OS scheduling enforces process migration, cgroups enforce resource usage tracking -- Synergistic

HPC lab.

The Impact of Multitasking on PSO

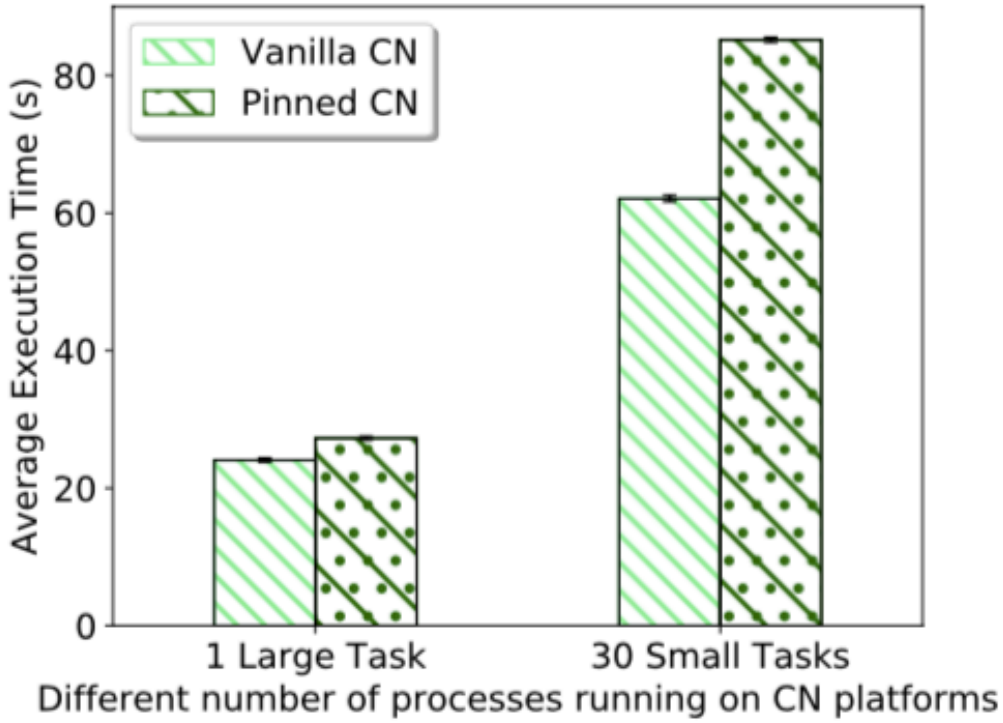


Fig. 8. Comparing the impact of number of processes on the imposed overhead of 4×Large CN instance. The vertical axis shows the mean executing times (in Seconds) and the horizontal axis shows processing of a source video file in two cases: one large video versus partitioning it into 30 small videos.

HPC lab.

The Impact of IO operations on PSO

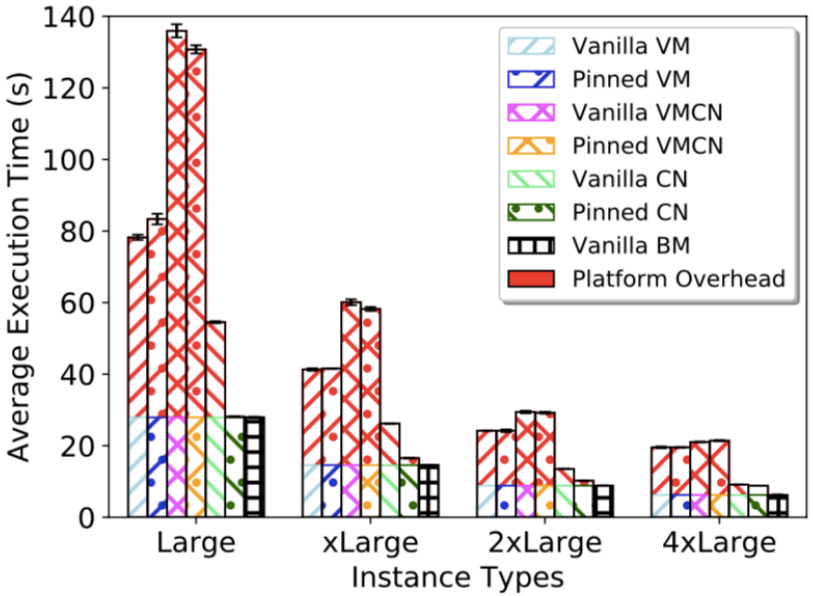


Fig. 3. Comparing execution time of FFmpeg on different execution platforms under varying number of CPU cores. Horizontal axis indicates the number of CPU cores in form of different instance types.

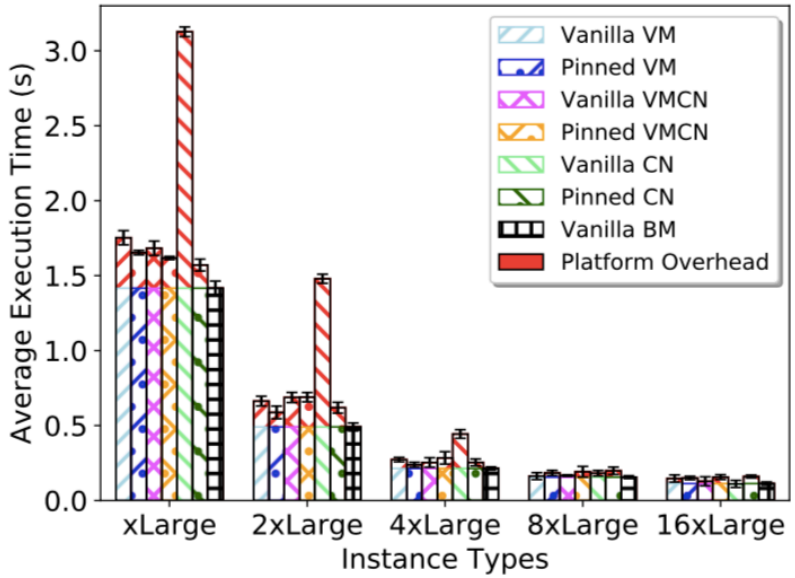


Fig. 5. Comparing mean response time (aka execution time) of 1,000 web processes on different execution platforms (WordPress evaluation). The horizontal axis represents the number of CPU cores in the form of different instance types and the vertical axis shows the mean execution time (in seconds).

- CPU pinning can mitigate this kind of overhead

HPCO lab.

Summary

1. Application characteristic is decisive on the imposed overhead
2. CPU pinning reduce the overhead for IO-bound applications running on containers.
3. CHR plays a significant role on the overhead of containers
4. Containers may induce higher overhead in comparing to VMs
5. Containers on top of VMs (called VMCN) impose a lower overhead for IO intensive applications

HPCO lab.

Best Practices

1. Avoid small vanilla containers
2. Use pinning for CPU-bound containers
3. Not worthwhile to use pinning for CPU-bound VMs
4. Use pinning for IO intensive workloads
5. CPU intensive applications: $0.07 < CHR < 0.14$
6. IO intensive applications: $0.14 < CHR < 0.28$
7. Ultra IO intensive applications: $0.28 < CHR < 0.57$

HPCO lab.