DCMIX: Generating Mixed Workloads for the Cloud Data Center

Xingwang Xiong[1], Lei Wang[1], Wanling Gao[1], Rui Ren[1], Ke Liu[1], Chen Zheng[1], Yu Wen[2], and Yi Liang[3]⋆

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{xiongxingwang, wl, gaowanling, renrui, liuke, zhengchen}@ict.ac.cn
[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[3] College of Computer Science, Beijing University of Technology, Beijing, China
yliang@bjut.edu.cn

Abstract. To improve system resource utilization, consolidating multi-tenants' workloads on the common computing infrastructure is a popular way for the cloud data center. The typical deployment of the modern cloud data center is co-locating online services and offline analytics applications. However, the co-locating deployment inevitably brings workloads' competitions for system resources, such as the CPU and the memory resources. These competitions result in that the user experience (the request latency) of the online services cannot be guaranteed. More and more efforts try to assure the latency requirements of services as well as the system resource efficiency. Mixing the cloud workloads and quantifying resource competition is one of the prerequisites for solving the problem. We proposed a benchmark suite—DCMIX as the cloud mixed workloads, which covered multiple application fields and different latency requirements. Furthermore the mixture of workloads can be generated by specifying mixed execution sequence in the DCMIX. We also proposed the system entropy metric, which originated from some basic system level performance monitor metrics as the quantitative metric for the disturbance caused by system resource competition. Finally, compared with the Service-Standalone mode (only executing the online service workload), we found that 99^{th} percentile latency of the service workload under the Mixed mode (workloads mix execution) increased 3.5 times, and the node resource utilization under that mode increased 10 times. This implied that mixed workloads can reflect the mixed deployment scene of cloud data center. Furthermore, the system entropy of mixed deployment mode was 4 times larger than that of the Service-Standalone mode, which implied that the system entropy can reflect the disturbance of the system resource competition. We also found that the isolation mechanism has some efforts for mixed workloads, especially the CPU-affinity mechanism.

Keywords: cloud computing data centers · system entropy · benchmark.

⋆ Yi Liang is the corresponding author.

1 Introduction

Today, more and more data centers are being used to provide cloud computing services, whatever the public cloud or the private cloud. To implement the economy of scale of cloud computing, consolidating more tenants' workloads is the basic idea [17]. Furthermore, the higher system resource utilization can bring more profits, so deploying diverse multi-tenant workloads on the same physical node is a popular way for the cloud data center. And typically, online services and offline analytics applications are co-located on shared resources [10]. However, the co-locating deployment inevitably brings workloads' competitions for system resources, such as CPU and memory resources within the same node. These competitions always result in high response latency of online service workload and further lead to poor user experience.

More and more previous work tries to assure the user experience as well as the system efficiency, such as Intel's Cache Allocation Technology [8], Linux Containers Technology [11], Labeled von Neumann Architecture [1], et al. Benchmarks measure the systems and architectures quantitatively, so the cloud data center benchmark is one of the prerequisites for solving the problem. There are two main challenges: first, the benchmark suite should reflect the application characteristic of cloud data center as well as the mixed execution pattern of cloud data center. Second, we need a metric to quantify the resource competition of mixed execution workloads.

In this paper, we propose DCMIX—a cloud data center benchmark suite covering multiple cloud application fields and the mixed workloads' execution mechanisms. DCMIX has 17 typical cloud data center workloads, which covered four typical application fields and the latencies of workloads range from microseconds to minutes. Furthermore, DCMIX can generate mixed execution sequence of workloads by the user customization, and it supports the mixture of serial execution and parallel execution. Then we propose system entropy as the joint entropy of system resource performance data, to reflect system resource competitions. We chose four system level metrics (CPU utilization, memory bandwidth, disk I/O bandwidth, and network I/O bandwidth) as the basic elements of the system entropy, and the system entropy is the joint entropy of them. The elements of the system entropy can easily get by monitoring the target node without third party application's participation, which is more suited for the public cloud scenes.

Finally, we conduct a series of experiments under five different modes on the X86 platform, which are Service-Standalone (only online services), Analytics-Standalone (only offline analytics applications), Mixed (workloads mix without any isolation setting), Mixed-Tied (workloads mix under the CPU-affinity setting), and Mixed-Docker (workloads mix under Linux containers). Compared with the Service-Standalone mode, we found that the latency of the service workload under the mixed mode increased 3.5 times, and the node resource utilization under that mode increased 10 times. Furthermore, the system entropy of the Mixed mode was 4 times larger than that of the Service-Standalone mode.

We also found that the isolation mechanisms have some efforts under the mixed mode, especially the CPU-affinity mechanism.

2 Related Work

Related work is summarized from two perspectives: cloud data center benchmarks and the system entropy.

For cloud data center benchmarks, we classify cloud data center benchmarks into two categories from the perspective of the co-locating deployment. The first one is generating multiple workloads individually, such as CALDA [12], Hibench [7], BigBench [5], BigDataBench 4.0 [4, 16], TailBench [9], and CloudSuite [3]. These benchmarks don't consider the co-locating deployment, and they provide multiple typical cloud data center workloads. CALDA provides Cloud OLAP workloads; Hibench provides Hadoop/Spark data analytics workloads; TailBench provides diverse tail latency sensitive service workloads; CloudSuite and Bigdatabench provide multiple workloads of the data center; BigBench provides an end-to-end data center workload. The second one is mixed workloads. SWIM [2] and CloudMix [6] build a workload trace to describe the realistic workloads mixed by mining production trace, and then run synthetic operations according to the trace. However, how to generate real workloads on the basis of mixture is still an open question.

In the area of the system entropy, the information entropy, also called Shannon entropy, is often used to quantify the degree of uncertainty of which information is produced by a stochastic source of data. Google [13] applied entropy for the system monitor, which is used to assess the stability of the profiling and sampling. BDTune [14] applied the relative entropy, which is the relative value of performance metrics on different data center nodes, to troubleshoot anomalous nodes in the data center. How to quantify resource competition in the cloud data center is still an open question.

3 DCMIX

Fig.1 shows the framework of DCMIX, there are four main modules: Workloads, User interface, Mixed workloads generator, and Performance monitor. DCMIX contains two types of workloads: online services and data analytic workloads, and they are all deployed on the target system. User interface is the portal for user, and users can specify their workload mix requirements, including workloads and mixture patterns. Mixed workloads generator can generate online services' requests and submit data analytics jobs to the target system. Performance monitor can monitor the performance data of the target system, and the system entropy is calculated by these original monitor data.

3.1 Workloads

DCMIX contains two types of workloads: online services and data analytic workloads. As shown on Fig.2, these workloads have different application fields and

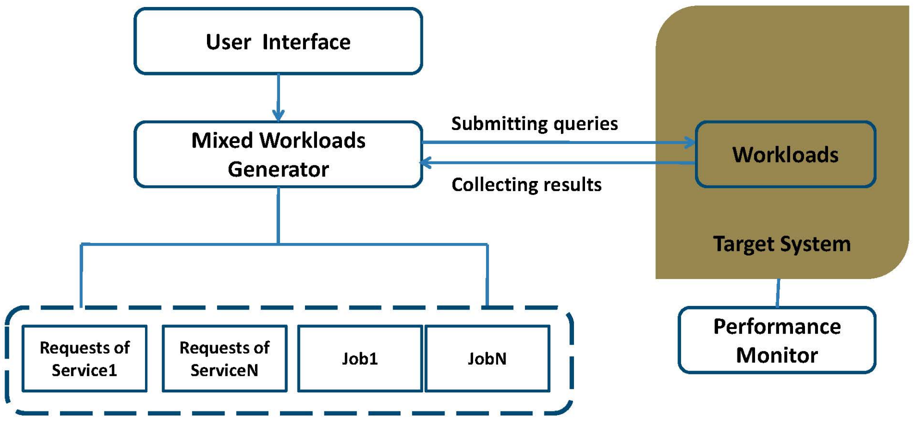

Fig. 1. The DCMIX Framework

different user experience (latency). DCMIX's application fields are big data, artificial intelligence, high-performance computing, transaction processing databases, et al. The latencies of DCMIX workloads range from microseconds to minutes.

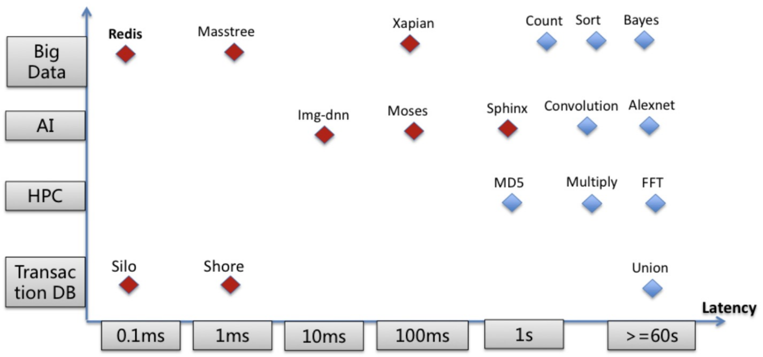

Fig. 2. The DCMIX Workloads

The details of workloads are shown on Table 1. DCMIX Workloads are from two famous benchmark suites, which are Bigdatabench 4.0 [4,16] and TailBench [9].

Table 1. The DCMIX Workloads

Workloads	Application Type	Domain	Latency Requirement
Count [16]	offline analytics application	Big Data	larger than 10s
Sort [16]	offline analytics application	Big Data	larger than 10s
Bayes [16]	offline analytics application	Big Data	larger than 10s
Convolution [16]	offline analytics application	AI	larger than 10s
Alexnet [16]	offline analytics application	AI	larger than 10s
MD5 [16]	offline analytics application	HPC	larger than 10s
Multiply [16]	offline analytics application	HPC	larger than 10s
FFT [16]	offline analytics application	HPC	larger than 10s
Union [16]	offline analytics application	Transaction DB	larger than 10s
Redis	online service	Big Data	less than 0.1ms
Xapian [9]	online service	Big Data	1∼ 100ms
Masstree [9]	online service	Big Data	1∼10ms
Img-dnn [9]	online service	AI	1∼20ms
Moses [9]	online service	AI	1∼100ms
Sphinx [9]	online service	AI	1∼10s
Silo [9]	online service	Transaction DB	less than 0.1ms
Shore [9]	online service	Transaction DB	1∼10ms

3.2 Mixed Workload Generator

Mixed workloads generator can generate the mixed workloads through submitting queries (service requests queries and data analytics job submitting queries). Mixed workloads generator supports the mixture execution of serial execution and parallel execution. Serial execution means that the workload must start up after the previous workload complete. Parallel execution means that multiple workloads start up at the same time.

Moreover, in the workload generator configuration file, users can set request configurations for each workload. For online-services, we provided request intensity, number of requests, number of warmup requests, etc.; for offline-analytics, we provide path of the data set, threads number of jobs, etc. Table 2 lists the parameters in the workload generator configuration file.

4 System Entropy

System entropy is used to reflect system resource disturbances, i.e., the uncertainty associated with resources usage.

Although the concept of system entropy has been proposed [18], there is no formal definition and corresponding calculation method. In this section, we defined the concept of system entropy as the joint entropy S of system resource performance data, to reflect system resource competition. The definition of System Entropy is based on the Shannon entropy. Shannon entropy is often used to quantify the degree of uncertainty of which information is produced by a stochastic source of data. The measure of Shannon entropy associated with each

Table 2. Parameters in Workload Generator Configuration

Parameter Name	**Description**
WarmupReqs	For online services. The number of requests for warm-up.
Reqs	For online services. The total number of requests (not include Warmup Reqs).
QPS	For online services. The average request rate, i.e., queries per second.
ServerThreads	For online services. The number of server threads for processing requests.
ClientThreads	For online services. The number of client threads for generating requests.
ServerIP	For online services. The IP address of the server.
ServerPort	For online services. The TCP/IP port used by the server.
JobThreads	For offline analytics workloads. The number of threads for executing jobs.
DataPath	For offline analytics workloads. The path of data set.

possible data value is the negative logarithm of the probability mass function for the value [15].

We chose four architecture-independent system metrics, which are CPU utilization, memory bandwidth utilization, disk I/O utilization, and network I/O bandwidth utilization, as elements of the system entropy. And the system entropy is the sum of these four elements' entropies. In other words, we measure system uncertainty with variations of the four most common system resource utilization.

As shown in Formula 1, S is the variable of system entropy, S contains four elements. C is the CPU utilization, which is defined as the percentage of time that the CPU executing at the system or user level. M is the memory bandwidth utilization, which is the occupied memory bandwidth divided by the peak memory bandwidth. D is the disk I/O utilization, which is the occupied disk I/O bandwidth divided by the peak disk I/O bandwidth. N is the network I/O utilization, which is the occupied network I/O bandwidth divided by the peak network I/O bandwidth.

$$S = (C, M, D, N) \tag{1}$$

As shown in Formula 2, the entropy of S is the joint entropy of (C, M, D, N), and we assume that these elements are independent of each other, so the calculation of $H(S)$ is the sum of them.

$$H(S) = H(C) + H(M) + H(D) + H(N) \tag{2}$$

The principle of system entropy is according with the information entropy. According to the information entropy calculation formula given by Shannon, for any discrete random variable X, its information entropy is defined as Formula 3 [15].

$$H(X) = -\sum_{x \in X} p(x) * \log_2 p(x) \tag{3}$$

So, the entropy of each element can be obtained by Formula 3. And we take C as the example to describe the calculation of $p(x)$. As shown on Formula 4, $p(c)$ is the probability of C, the $Num(c)$ is the count of the value is c in the sample, and n is the total number of the sample.

$$p(c) = \frac{Num(c)}{n} \tag{4}$$

5 Experiment and Experimental Analysis

5.1 Experimental Configurations and Methodology

Experimental Configurations We used two physical nodes for experiments, one is the target node (Server node) and the other is the workload generator node (Client node). The operating system of the Server node is Linux Ubuntu 16.04. The Server node is equipmented with Intel Xeon E5645 processor and 96GB memory. The detailed configurations are summarized in Table 3.

Table 3. The Configuration of the Server Node

CPU	Intel(R) Xeon(R) E5645 2.40G
Memeory	96GB DDR3 1333MHz bandwidth:8GB/s
Network	Ethernet 1G bandwidth:943Mbits/s
Disk	SATA 1T bandwidth:154.82MB/s
OS	Ubuntu 16.04 and the kernel is 4.13.0-43-generic
GCC	4.3
Redis	4.2.5

We chose four workloads in the experiments, they are Redis (the online service workload), Sort (the offline analytics workload), Wordcount (the offline analytics workload), and MD5 (the offline analytics workload). Redis is a single thread in-memory database, which has been used in the cloud widely. Sort and Wordcount are multi-threaded big data workloads, which is implemented with OpenMP in our experiment. MD5 is a multi-threaded HPC workload, which is also implemented with OpenMP. Four workloads are deployed on the Server node. And we deployed the workload generator on the Client node. We generated the mixed workloads with the parallel execution mode, in which four workloads start up at the same time and run together. For the offline analytics workloads, we submitted jobs of Sort, Wordcount and MD5 with 8GB data scale. For the online service workload, the client request intensity of Redis is 50,000 requests per second, and follows the exponential distribution.

Experimental Methodology We conduct the experiment under five different modes, which were Service-Standalone, Analytics-Standalone, Mixed, Mixed-Tied, and Mixed-Docker. For the Service-Standalone mode, we only run the Redis workload on the physical machine. For the Analytics-Standalone mode, we run all of offline workloads on the physical machine. For the Mixed mode, we co-located Redis and offline workloads on the physical machine without any isolation setting, but the total thread number is according with the total hardware thread number of the target platform. For the Mixed-Tied mode, we run Redis and offline workloads on separated cores through the CPU affinity setting. Different with the Mixed mode, we run Redis on one core, while run the other offline workloads on the other cores. For the Mixed-Docker mode, Redis and offline workloads were executed in two separate Docker containers (Redis run on one container, and offline workloads run on the other container).

Metrics The evaluation metrics cover the spectrum of user-observed metrics, system level metrics, and micro-architectural metrics. As for user-observed metrics, we chose the average latency and the tail latency. In terms of system level metrics, we chose CPU utilization, memory bandwidth utilization, disk bandwidth utilization, and network I/O bandwidth utilization.

5.2 Experiment Results and Observations

The user-observed metric Fig.3 shows the latency of Redis. From Fig.3, we have the following observations:

First, the tail latency is severe, even in the Service-Standalone mode. In the Service-Standalone mode, we only run Redis (the single thread workload) on the multi-core node (Intel Xeon processor), the 99^{th} latency (0.367 ms) is 2 times to the average latency (0.168 ms), and 99.9^{th} latency (0.419 ms) is 2.5 times to the average latency. This implied that the state-of-practice system architecture, i.e., CMP micro-architecture and time-sharing OS-architecture, would incur the high tail latency.

Second, mixed deployment without any isolation mechanism also incurs the high latency. In the Mixed mode, the average latency is 0.429 ms (2.6 times to the Service-Standalone mode) and 99.9^{th} latency is 16.962 ms (27 times to the Service-Standalone mode). Although, the thread number accords with the total hardware thread number of the target platform, the interfere of mixed deployment should incur the high latency of online services.

Third, the CPU affinity setting can relieve the competition. The average latency of Mixed-tied is 0.173 ms and 99.9^{th} latency is 1.371 ms. So in our condition, the CPU affinity setting can relieve the competition efficiently.

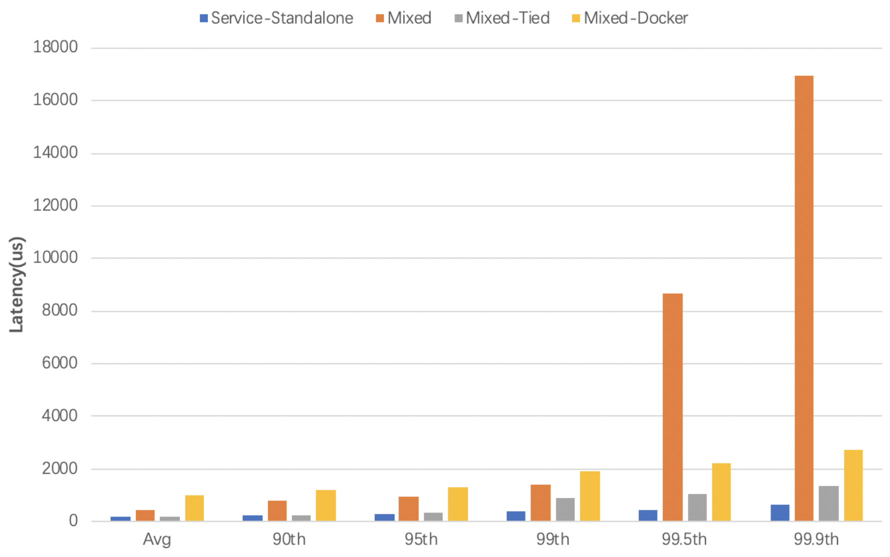

Fig. 3. The Request Latency of the Redis

Fourth, the average latency of Mixed-Docker is 0.977 ms and 99.9^{th} latency is 2.75 ms. The container can relieve the tail latency, but make the average latency higher.

The system level metrics for the system Fig.4 presents the resource utilization of server node. From Fig.4, we find that mixed deployment can prompt the resource utilization. The CPU utilization of the Service-Standalone mode is only 4%, while the mixed deployment can achieve 46%-55%.

Fig.5 shows the system entropy of server node. From Fig.5, we find that the system entropy of the Service-Standalone mode is only 5.9, while that of the Analytics-Standalone, the Mixed mode, the Mixed-Tied mode, and the Mixed-Docker mode are 20,23,22,25 respectively. Furthermore, the system entropy of the Mixed-tied mode is the minimum among all of the mix modes.

The architecture level metrics for the system Fig.6 shows the micro-architecture metrics of server node. From Fig.6, we find larger L1I cache misses and L2 cache misses under the Service-Standalone mode, smaller L1I cache misses and L2 cache misses under Analytics-Standalone mode, and that the micro-architecture metrics have minor variations among three mixed modes. In other words, the micro-architecture metrics can not reflect the disturbance caused by system resource competition.

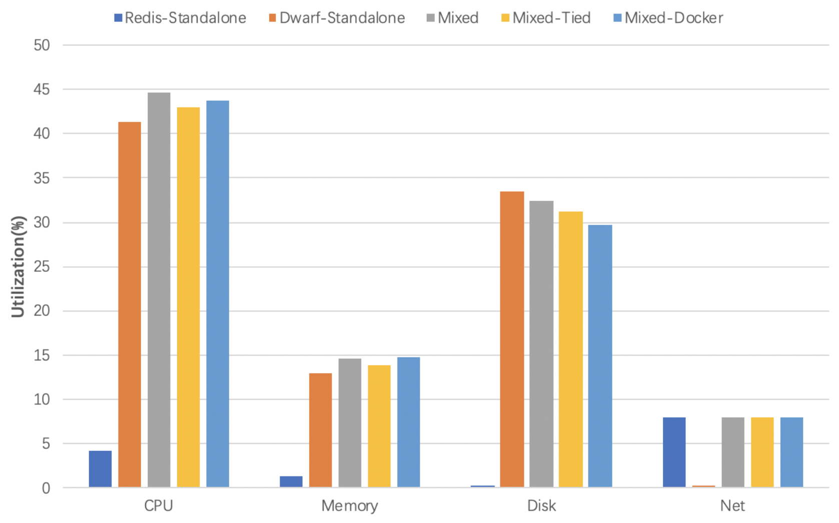

Fig. 4. The System Level Metrics of the Server Node

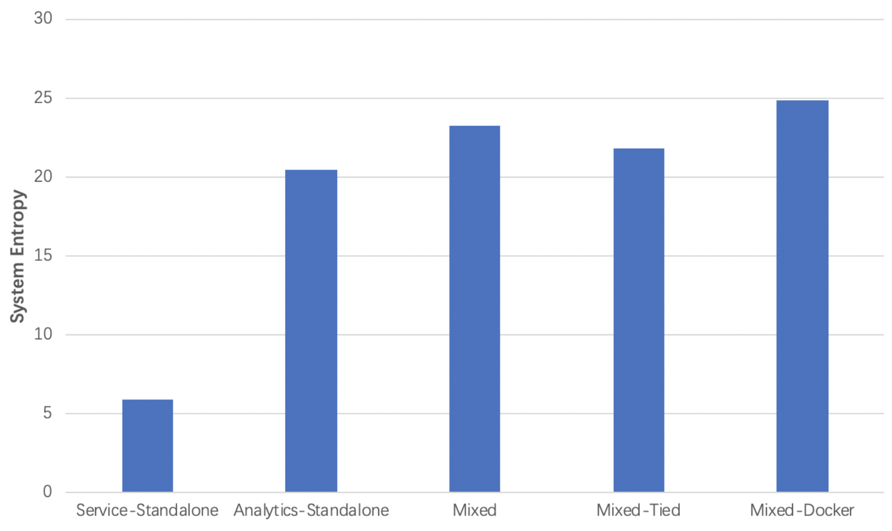

Fig. 5. The System Entropy of the Server Node

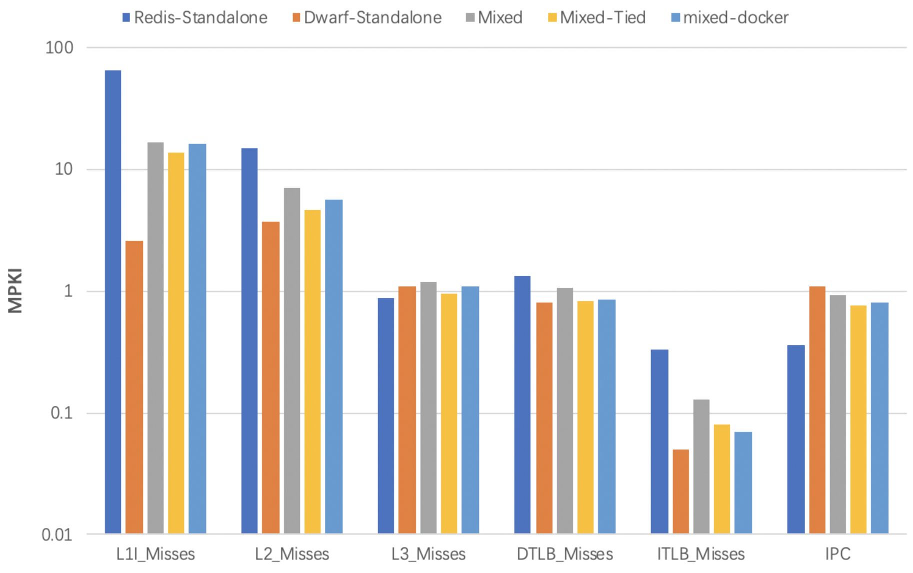

Fig. 6. The Architecture Metrics of the Server Node

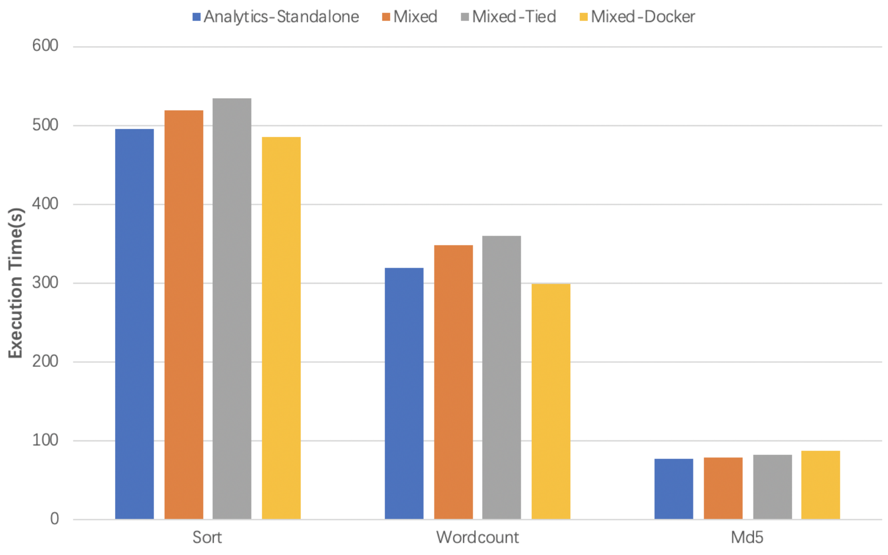

Fig. 7. Offline Analytics Application Execution Time

Offline analytics application execution time Fig.7 shows offline analytics application execution time under four different modes. From Fig.7, we find that the execution time of Sort under the Analytics-Standalone mode is 495s, and that under the Mixed mode, the Mixed-Tied mode, and the Mixed-Docker mode are 519s, 534s, 486s respectively. Interference has less impact on offline analytics applications than that on the online services.

5.3 Summary

Mixed workloads Compared with the Service-Standalone mode, we found the latency of the service workload under the Mixed mode increased 3.5 times, and the node resource utilization under that increased 10 times. This implied that mixed workloads can reflect the mixed deployment scene.

Tail latency of the service workload The state-of-the-practice system architecture, i.e., CMP micro-architecture and time-sharing OS-architecture, should incur the high tail latency, even in the Service-Standalone mode.

The system entropy for the server node The system entropy of the Mixed mode was 4 times larger than that of the Service-Standalone mode, and its tendency was corresponding to latency among different mixed modes. This implied that the system entropy can reflect the disturbance caused by system resource competition.

Isolation mechanisms State-of-the-practice isolation mechanisms have some efforts under the mixed workloads, especially the CPU-affinity mechanism.

Impacts for offline workloads Compared with execution time under the Analytics-Standalone mode, there is only a slight increase in execution time of offline analytics applications under the mixed modes. So we can see that the root cause of long latency of online services under the co-locating deployment is not insufficient resources, but the short-term disorder competitions.

6 Conclusion

In this paper, we proposed DCMIX as the cloud data center benchmark suite. We also defined the system entropy to quantify resource competition in the cloud data center. Through the experiment, we found that DCMIX can reflect the mixed deployment scene in the cloud data center and the system entropy can reflect the disturbance of the system resource competition.

7 Acknowledgment

This work is supported by the National Key Research and Development Plan of China Grant No.2016YFB1000201.

References

1. Bao, Y.G., Wang, S.: Labeled von neumann architecture for software-defined cloud. Journal of Computer Science and Technology **32**(2), 219–223 (Mar 2017). https://doi.org/10.1007/s11390-017-1716-0, https://doi.org/10.1007/s11390-017-1716-0
2. Chen, Y., Alspaugh, S., Katz, R.: Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. Proc. VLDB Endow. **5**(12), 1802–1813 (Aug 2012). https://doi.org/10.14778/2367502.2367519, http://dx.doi.org/10.14778/2367502.2367519
3. Ferdman, M., Adileh, A., Kocberber, O., Volos, S., Alisafaee, M., Jevdjic, D., Kaynak, C., Popescu, A.D., Ailamaki, A., Falsafi, B.: Clearing the clouds: A study of emerging workloads on modern hardware p. 18 (2011)
4. Gao, W., Zhan, J., Wang, L., Luo, C., Zheng, D., Wen, X., Ren, R., Zheng, C., Ye, H., Dai, J., et al.: Bigdatabench: A scalable and unified big data and ai benchmark suite. Under review of IEEE Transaction on Parallel and Distributed Systems (2018)
5. Ghazal, A., Rabl, T., Hu, M., Raab, F., Poess, M., Crolotte, A., Jacobsen, H.A.: Bigbench: Towards an industry standard benchmark for big data analytics. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. pp. 1197–1208. SIGMOD '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2463676.2463712, http://doi.acm.org/10.1145/2463676.2463712
6. Han, R., Zong, Z., Zhang, F., Vazquez-Poletti, J.L., Jia, Z., Wang, L.: Cloudmix: Generating diverse and reducible workloads for cloud systems. In: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD). pp. 496–503 (June 2017). https://doi.org/10.1109/CLOUD.2017.123
7. Huang, S., Huang, J., Dai, J., Xie, T., Huang, B.: The hibench benchmark suite: Characterization of the mapreduce-based data analysis. In: 2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010). pp. 41–51 (March 2010). https://doi.org/10.1109/ICDEW.2010.5452747
8. Intel Corporation: Improving Real-Time Performance by Utilizing Cache Allocation Technology (April 2015), http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/cache-allocation-technology-white-paper.pdf
9. Kasture, H., Sanchez, D.: Tailbench: a benchmark suite and evaluation methodology for latency-critical applications. In: 2016 IEEE International Symposium on Workload Characterization (IISWC). pp. 1–10 (Sept 2016). https://doi.org/10.1109/IISWC.2016.7581261
10. Liu, Q., Yu, Z.: The elasticity and plasticity in semi-containerized co-locating cloud workload: A view from alibaba trace. In: Proceedings of the ACM Symposium on Cloud Computing. pp. 347–360. SoCC '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3267809.3267830, http://doi.acm.org/10.1145/3267809.3267830
11. Merkel, D.: Docker: Lightweight linux containers for consistent development and deployment. Linux J. **2014**(239) (Mar 2014), http://dl.acm.org/citation.cfm?id=2600239.2600241
12. Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., DeWitt, D.J., Madden, S., Stonebraker, M.: A comparison of approaches to large-scale

data analysis. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. pp. 165–178. SIGMOD '09, ACM, New York, NY, USA (2009). https://doi.org/10.1145/1559845.1559865, http://doi.acm.org/10.1145/1559845.1559865

13. Ren, G., Tune, E., Moseley, T., Shi, Y., Rus, S., Hundt, R.: Google-wide profiling: A continuous profiling infrastructure for data centers. IEEE Micro **30**(4), 65–79 (July 2010). https://doi.org/10.1109/MM.2010.68
14. Ren, R., Jia, Z., Wang, L., Zhan, J., Yi, T.: Bdtune: Hierarchical correlation-based performance analysis and rule-based diagnosis for big data systems. In: 2016 IEEE International Conference on Big Data (Big Data). pp. 555–562 (Dec 2016). https://doi.org/10.1109/BigData.2016.7840647
15. Shannon, C.E.: A mathematical theory of communication. Bell system technical journal **27**(3), 379–423 (1948)
16. Wang, L., Zhan, J., Luo, C., Zhu, Y., Yang, Q., He, Y., Gao, W., Jia, Z., Shi, Y., Zhang, S., Zheng, C., Lu, G., Zhan, K., Li, X., Qiu, B.: Bigdatabench: A big data benchmark suite from internet services. In: 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA). pp. 488–499 (Feb 2014). https://doi.org/10.1109/HPCA.2014.6835958
17. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications **1**(1), 7–18 (May 2010). https://doi.org/10.1007/s13174-010-0007-6, https://doi.org/10.1007/s13174-010-0007-6
18. Zhiwei, X., Chundian, L.: Low-entropy cloud computing systems. SCIENTIA SINICA Informationis **47**(9), 1149 (2017). https://doi.org/https://doi.org/10.1360/N112017-00069, http://engine.scichina.com/publisher/Science China Press/journal/SCIENTIA SINICA Informationis/47/9/10.1360/N112017-00069