浙 江 大 学

硕士学位论文开题报告

（专业学位）

论文题目： 资源感知的混部系统的设计与实现

姓　名： 李天硕

学　号： 22251075

专　业： 软件工程

院　别： 软件学院

二零××　年　×　月

本人的论文进度计划在最后一个章节

目录

1 课题来源及类型

云计算逐渐成为企业部署及运维企业服务的主要模式，在云计算技术的支撑下，企业极大地提升了 IT 运维效率。云计算技术明显地改变了互联网应用架构设计与发展出了一条新的商业模式。

互联网行业的头部企业，出于成本考虑，通常会自建数据中心，数据中心通常由上万台服务器组成，成本巨大。在行业快速发展的时期，企业通常不会关注成本投入，而如今全球经济受到疫情，俄乌冲突等因素影响的背景下，企业开始频繁提出“降本增效”的成本收缩策略，数据中心的成本投入越来越受到重视，据统计，全球数据中心的 CPU 资源利用率仅为 10%至 20%[1]。其实早在 2015 年 Google 于研究成果 Borg[2]提出混部的概念后(混部是改变传统的应用单机部署模式，利用容器技术将应用部署到同一台物理机，提升资源利用率)，国内的头部互联网公司已经关注到数据中心成本节约的问题。

目前，国内大多数互联网都采用 Google 开源的 Kubernetes 开源编排系统实现混部平台，而 Kubernetes 作为设计极具通用性的框架，其原生调度策略已经难以满足企业目前的应用部署需求[5]，在实际生产落地实现混部平台时仍存在诸多缺陷：

Kubernetes 仅能感知 CPU，内存容量，硬盘空间资源，而其中仅 CPU 资源受到竞争对应用性能会受到影响。

Kubernetes 无应用资源画像能力，仅依靠用户定义的资源需求进行容器编排调度，这将引起集群跨节点的资源负载不均衡，集群易出现高负载节点，存在较高的服务质量风险。

2 课题的意义及现状分析

如今大数据应用被广泛采用，主要业务场景包含但不限于日志统计，离线数据分析等，这些应用通常被称为批处理应用(离线应用)。这类应用的特点为: (1)延迟不敏感。(2)资源占用高。

批处理应用领域著名的框架有 Spark[6]，Flink[7]；批处理应用的分发与资源

管理由一些调度器处理，主要的知名开源调度编排系统有 Mesos[8]，Yarn[9]。

与批处理应用相对应的是在线应用，这类应用的特性为: (1)延迟敏感。(2)资源需求周期性强。

早期，在线应用都是以二进制形式，单机部署的。资源浪费严重，且动态伸缩灵活性很差。Openstack[10]出现充分利用虚拟化的特性，将物理机器池化成虚拟机的粒度，提高了动态伸缩的灵活性，不过此时应用的部署还是单机粒度。为了推动进一步的资源节省，Docker[11]与 Kubernetes[12]应运而生。Docker 基于轻量虚拟化的机制，实现应用更细粒度的部署，标准化的形式实现应用的交付与生命周期管理。在 Docker 的基础上，Google 根据运营了十多年的编排系统 Borg 进行开源，取名为 Kubernetes；Kubernetes 提供了一个易于拓展且标准化的容器编排系统。

然而，如今开源的编排系统，都没有真正地实现负载感知。开放的调度决策中，仅考虑业务方为应用定义的任务资源需求与节点资源状态进行调度决策。但是这存在两个缺陷: (1)用户定义的资源需求是静态的，应用实际的资源需求却是动态的。(2)用户定义的资源需求是不准确的，业务方通常不会在意应用究竟需求多少资源，业务方通常会直接选择最大可选资源，如图 0.1　某集群一段时间内 CPU 资源使用量与请求量对比图。所以采用目前开源的编排系统实现混部平台将引起的问题包括: (1)集群中节点的负载不均衡，极易出现高负载节点，集群存在应用服务质量受影响的风险。(2)集群资源浪费。

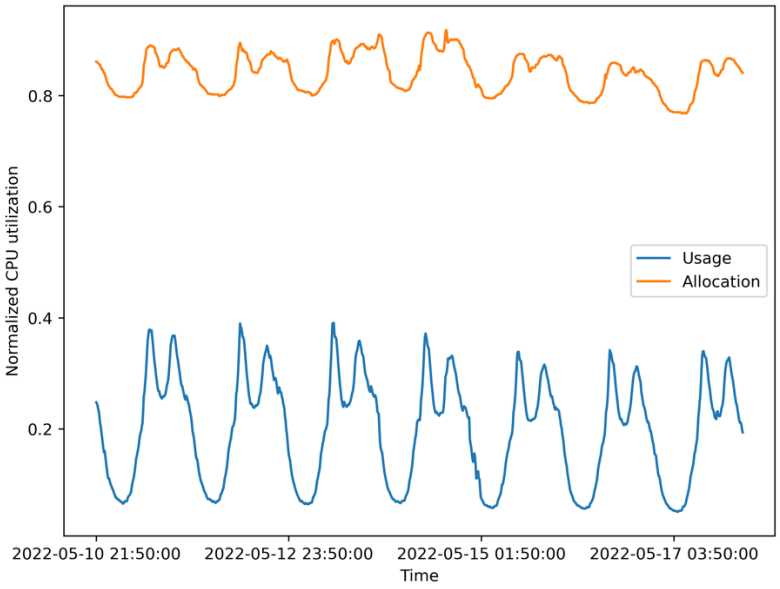

图 0.1　某集群一段时间内 CPU 资源使用量与请求量对比图

在研究领域，为了感知应用的实际资源需求，即构建应用资源画像。autopilot[13]采用基于半衰期的移动窗口算法优化机器资源的超卖，他们的方法比起人工设置资源需求，将应用的实际资源使用量与为该应用分配的资源量的差距从 46%缩短到了 23%；peak oracle[14]，　Pawel Janus[15], 3Sgima[16]都采用概率分布的方法构建应用资源画像，优化超卖策略，或者是结合 Bin-Packing 算法[17]进一步优化调度策略。随着机器学习，神经网络等技术在学术界研究的深入，一些研究也将这些技术采用到应用资源画像的构建，如 Janardhanan et al.[18]使用 LSTM 预测应用 CPU 使用量的时间序列，Eli Cortez[19]使用 XGBoost 预测应用的资源使用量与生命周期。不过，对于生产落地，企业更愿意采用统计或者其他传统的可解释性强的方法。

除了未实现真正的负载感知外，目前开源的编排系统在调度时考虑的资源维度较少，仅考虑了 CPU Cores，内存容量，硬盘大小等。根据本文的调研与实验验证，在资源受到竞争时对应用性能会产生明显影响的资源包括但不限于：(1)CPU Core。(2)内存子系统资源[20],[21]，包括 LLC[22]，内存控制器。来源于 Google 的研究表明，在他们的验证的在线服务中，单独受到 LLC 竞争引起的性能损失最高达 11%，而同时收到 LLC 与内存控制器两类资源竞争引起的性能损失最高达 22%[23]。

3 课题的研究目标、研究内容和拟解决的关键问题

3.1 课题研究目标

本文考虑到在云原生领域，现有的编排调度系统存在的几点问题：

1. 默认调度算法对应用负载需求无感知，集群资源负载不均衡。
2. 编排系统对底层共享资源竞争无感知（本文主要关注内存子系统压力）。
3. 编排系统的调度仿真能力缺失。

本文针对以上的主要问题，结合本文参与的平台已有能力，对某公司现有调度系统提出了增强的解决方案，一定程度上解决了上述存在的问题。另外，本文考虑的问题是普遍存在的，解决方案也是较为通用的方法，所以在公司进行生产环境落地后，未来有机会将会考虑将这部分能力开放到社区中。

3.2 拟解决的关键问题

为了填补 Kubernetes 框架的缺陷，本文在所处的基于 Kubernetes 调度平台之上，设计与实现了一套解决方案，增强集群调度能力。本文的主要工作：(1)硬件指标采集工具，该工具补全了容器的内存子系统压力相关指标与 cycles-per-instructions(CPI)的监控，前者丰富调度系统的资源感知，后者用于集群数据分析。(2)通用的内存子系统压力定义，包括 last level cache(LLC)与内存控制器资源的压力定义。(3)应用资源画像与基于画像的调度策略，本文采用通用且易生产落地的方法构建了应用资源画像，并且基于画像设计了基于高斯估计的调度优化策略，为集群跨节点资源负载(cpu/cache/memory controller)的均衡度提升最高达(38%,28%,32%)。(4)调度仿真工具，实现对 Kubernetes 组件的端到端的调度验证。

3.3 课题设计方案

基于 K8S 调度器的缺陷与内存子系统压力相关的需求分析，现给出系统的软件实现总架构，其中一些功能实现依赖于公司平台现有的能力，如图 4.1 为本文系统方案的设计，图中的灰色部分是本文的主要工作，其他部分则主要依赖平台的已有能力。主要包括内存子系统压力定义，系统监控，资源画像，调度策略，以及调度仿真工具五个部分。

内存子系统压力定义：本文根据相关研究工作调研，提出了内存子系统压力的量化定义，弥补了现有调度系统对这部分资源感知的缺失，并将该定义纳入到集群调度的考虑，优化集群中跨节点的内存子系统压力不均衡问题。另外，本文也通过单机实验与集群数据分析验证了该指标的有效性。

系统监控：本文在监控系统方面的工作，是扩展了目前公司与原生 Kubernetes 缺失的硬件性能指标的数据监控，开发了硬件指标采集工具。该工具采集的指标将用于：(1)作为应用性能指标的代理，助于本文的数据分析。(2)组合定义量化应用与主机内存子系统压力的量化指标，丰富了资源感知的能力。Node Exporter/cAdvisor/falcon-agent 等工具为平台已具备的能力，这些工具能够监控节点/容器的资源使用量（CPU/内存/网络 IO 等）。

资源画像：基于一些可解释性强，易于生产落地的统计算法，对集群所有应用构建画像，画像将作为调度策略的数据支撑。主要内容包括应用容量推荐与应用资源消耗特征，两个功能都应用统计的方法，构建出应用画像，在集群调度的资源超卖与集群资源均衡化方面发挥作用，实现调度策略的优化。

调度策略：调度策略方面考虑了公司现有调度方案的不足，引入应用资源画像的信息作为输入(画像缓存到 PortraitCache)，设计了基于画像数据的调度策略，充分优化了调度策略中的节点预选（Predicates）与节点优选（Priorities）两类策略。进一步优化资源浪费，集群负载不均衡等问题。

调度仿真工具：由于 K8S 中缺乏对调度算法与调度系统的验证，这将导致新的调度方案上线存在一定风险，为了实现调度策略的优化程度及有效性。本文实现的调度仿真工具可模拟虚拟节点，从线上集群生成生产实际的调度序列，将调度序列输入到仿真集群中，达成调度仿真的目的。

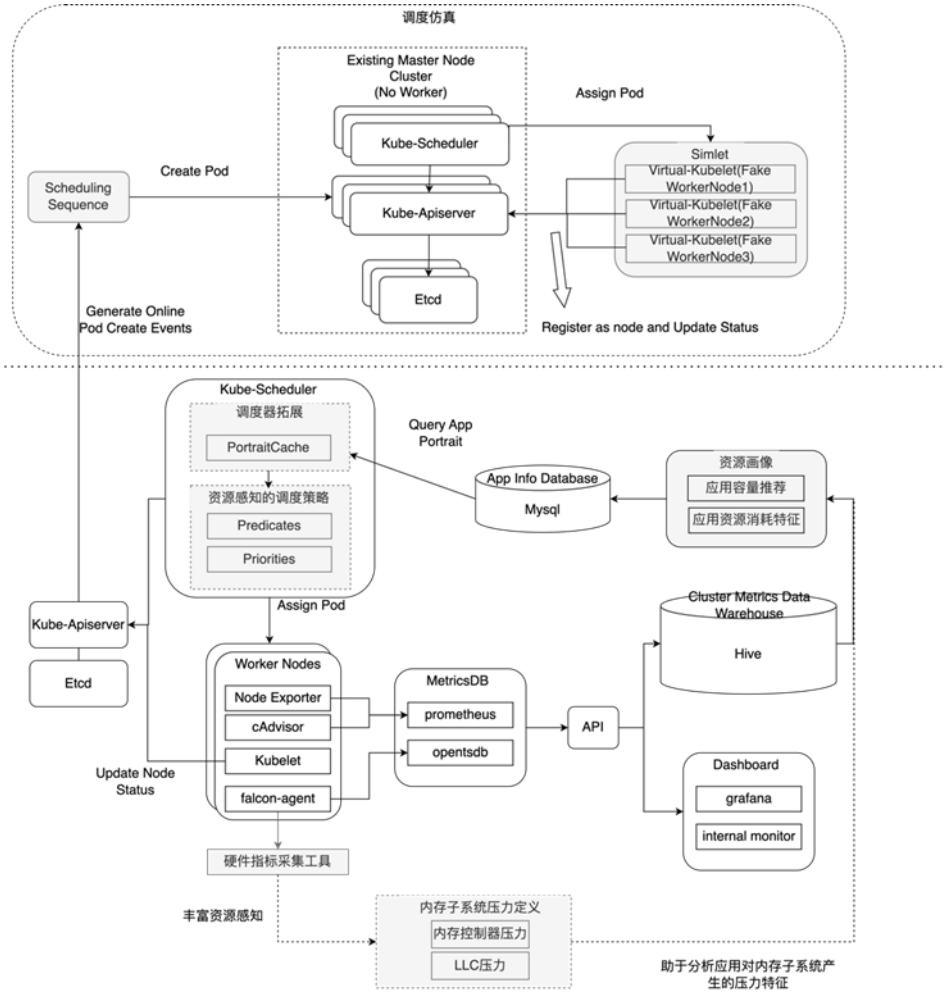

图 4.1　系统设计架构图

3.4 课题可行性分析

混合部署技术在云服务供应商之间的使用已经比较成熟，在众多互联网厂商也开始普遍使用，混合部署的研究也有一定的基础。本文将通过研究与实际生产环境数据分析及实验，提出了通用的内存子系统压力量化指标，使得调度系统可对内存子系统部分资源感知；应用半衰期滑动窗口与直方统计及概率分布此类易于落地的通用传统方法，对应用构建资源画像，深度分析了 Kubernetes 默认优选与预选策略的劣势，基于画像数据与概率分布理论实现了 Kubernetes(K8S)的调度策略，优化了集群资源负载分布不均衡的问题。在本文方法落地实践证明有效后，也可为社区提供通用的资源画像与调度优化策略解决方案；实现了基于 Kubernetes 的调度仿真工具，使得新的调度策略可被端到端的形式进行测试，允许调度策略开发人员在策略上线进行预先验证，规避策略上线引起的风险。

4 课题计划进度和预期成果

4.1 计划进度

2022 年 5 月-2022 年 9 月　　写开题报告，前期预研

2022 年 9 月-2022 年 11 月　写文献综述，开始实验与优化

2022 年 11 月-2022 年 12 月　开始写论文初稿

2022 年 11 月-2023 年 1 月　论文修改、定稿

4.2 预期成果

论文一篇；

结合公司环境与项目进行落地实践；

相关代码开源到社区。

5 本人毕业论文工作计划

时间点	工作内容计划
2023 年 6 月-2024 年 3 月	前期知识储备、技术积累，方向调整
2024 年 3 月-2024 年 6 月	查阅方向相关论文，写开题报告
2024 年 7 月-2024 年 9 月	写文献综述，设计实验与算法
2024 年 9 月-2024 年 11 月	实验验证与优化，开始写论文初稿
2024 年 11 月-2025 年 1 月	论文修改、定稿

参考文献

[1] Shehabi A, Smith S, Sartor D, Brown R, Herrlin M, Koomey J et al. United States Data Center Energy Usage Report[J]. Lawrence Berkeley National Laboratory, Berkeley, California, 2016. 65 p.

[2] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg[C]. In Proceedings of the Tenth European Conference on Computer Systems (EuroSys '15). Association for Computing Machinery, New York, NY, USA, Article 18, 1–17. DOI:https://doi.org/10.1145/2741948.2741964.

[3] Shuang Chen, Christina Delimitrou, and José F. Martínez. 2019. PARTIES: QoS-Aware Resource Partitioning for Multiple Interactive Services[C]. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19). Association for Computing Machinery, New York, NY, USA, 107–120. https://doi.org/10.1145/3297858.3304005.

[4] Shaohong Li, Xi Wang, Faria Kalim, Xiao Zhang, Sangeetha Abdu Jyothi, Karan Grover, Vasileios Kontorinis, Nina Narodytska, Owolabi Legunsen, Sreekumar Kodakara, et al. Thunderbolt:Throughput-Optimized ,Quality-of-Service-Aware power capping at scale[C]. In 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), pages 1241–1255, 2020.

[5] 中国信通院．云原生发展白皮书[R]. 北京:中国信息通信研究院, 2020.

[6] Apache. Spark Standalone Mode - Spark 3.3.1 Documentation[EB/OL].https://spark.apache.org/docs/latest/spark-standalone.html, 2022.

[7] Apache. Stateful Computations over Data Streams[EB/OL]. https://flink.apache.org/, 2022.

[8] Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A. D., Katz, R. H., Shenker, S., and Stoica, I. Mesos: A platform for fine-grained resource sharing in the data center[C]. In NSDI (2011), vol. 11, pp. 22–22.

[9] Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., et al. Apache hadoop yarn: Yet another resource negotiator[C]. In Proceedings of the 4th annual Symposium on Cloud Computing (2013), ACM, p. 5.

[10] OpenDev. OepnDev Manual[EB/OL]. https://docs.opendev.org/opendev/infra-manual/latest/index.html, 2022.

[11] Docker Inc.. Docker Engin overview[EB/OL]. https://docs.docker.com/engine/, 2022.

[12] The Linux Foundation. Kubernetes Documentation[EB/OL]. https://kubernetes.io/docs/home/, 2022.

[13] Krzysztof Rzadca, Pawel Findeisen, Jacek Swiderski, Przemyslaw Zych, Przemyslaw Broniek, Jarek Kusmierek, Pawel Nowak, Beata Strack, Piotr Witusowski, Steven Hand, and John Wilkes. 2020. Autopilot: workload autoscaling at Google[C]. In Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys '20). Association for Computing Machinery, New York, NY, USA, Article 16, 1–16. DOI:https://doi.org/10.1145/3342195.3387524.

[14] Noman Bashir, Nan Deng, Krzysztof Rzadca, David Irwin, Sree Kodak, and Rohit Jnagal. 2021. Take it to the limit: peak prediction-driven resource overcommitment in datacenters[C]. In Proceedings of the Sixteenth European Conference on Computer Systems (EuroSys '21). Association for Computing Machinery, New York, NY, USA, 556–573. https://doi.org/10.1145/3447786.3456259.

[15] Pawel Janus and Krzysztof Rzadca. 2017. SLO-aware colocation of data center tasks based on instantaneous processor requirements[C]. In Proceedings of the 2017 Symposium on Cloud Computing (SoCC '17). Association for Computing Machinery, New York, NY, USA, 256–268. https://doi.org/10.1145/3127479.3132244.

[16] J. W. Park, A. Tumanov, A. Jiang, M. A. Kozuch, and G. R. Ganger. 3Sigma: Distribution-based cluster scheduling for runtime uncertainty[C]. In Proceedings of the Thirteenth EuroSys Conference, 2018.

[17] Korte, Bernhard; Vygen, Jens (2006). Bin-Packing[M]. Combinatorial Optimization: Theory and Algorithms. Algorithms and Combinatorics 21. Springer. pp. 426–441. doi:10.1007/3-540-29297-7_18. ISBN 978-3-540-25684-7.

[18] Janardhanan D, Barrett E. CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models[C]. In 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), 2017: IEEE, p 55–60.

[19] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms[C]. In Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17). Association for Computing Machinery, New York, NY, USA, 153–167. DOI:https://doi.org/10.1145/3132747.3132772.

[20] Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. 2011. Bubble-Up: increasing utilization in modern warehouse scale computers via sensible co-locations[J]. In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44). Association for Computing Machinery, New York, NY, USA, 248–259. https://doi.org/10.1145/2155620.2155650.

[21] D. Dasari and V. Nelis. An Analysis of the Impact of Bus Contention on the WCET in Multicores[C]. 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems, 2012, pp. 1450-1457, doi: 10.1109/HPCC.2012.212.

[22] Nikolay A. Simakov, Robert L. DeLeon, Joseph P. White, Thomas R. Furlani, Martins Innus, Steven M. Gallo, Matthew D. Jones, Abani Patra, Benjamin D. Plessinger, Jeanette Sperhac, Thomas Yearke, Ryan Rathsam, and Jeffrey T. Palmer. 2016. A Quantitative Analysis of Node Sharing on HPC Clusters Using XDMoD Application Kernels[C]. In Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale (XSEDE16). Association for Computing Machinery, New York, NY, USA, Article 32, 1–8. https://doi.org/10.1145/2949550.2949553.

[23] Lingjia Tang, Jason Mars, Neil Vachharajani, Robert Hundt, and Mary Lou Soffa. 2011. The impact of memory subsystem resource sharing on datacenter applications[C]. In Proceedings of the 38th annual international symposium on Computer architecture (ISCA '11). Association for Computing Machinery, New York, NY, USA, 283–294. https://doi.org/10.1145/2000064.2000099.

[24] D. Lo, L. Cheng, R. Govindaraju, P. Ranganathan and C. Kozyrakis. Heracles: Improving resource efficiency at scale[C]. 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA), 2015, pp. 450-462, doi: 10.1145/2749469.2749475.

[25] Y. Zhang et al.. LIBRA: Clearing the Cloud Through Dynamic Memory Bandwidth Management[C]. 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 815-826, doi: 10.1109/HPCA51647.2021.00073.

[26] Yaocheng Xiang, Chencheng Ye, Xiaolin Wang, Yingwei Luo, and Zhenlin Wang. 2019. EMBA: Efficient Memory Bandwidth Allocation to Improve Performance on Intel Commodity Processor[C]. In Proceedings of the 48th International Conference on Parallel Processing (ICPP 2019). Association for Computing Machinery, New York, NY, USA, Article 16, 1–12. https://doi.org/10.1145/3337821.3337863.

[27] Intel Inc.. Intel® resource director technology reference manual[EB/OL], https://www.intel.com/content/dam/develop/external/us/en/documents/180115-intel-rdtcascadelake-serverreferencemanual-806717.pdf , 2019.

[28] LIN M, XI J Q, BAI W H. Ant Colony Algorithm for Multi-Objective Optimization of Container-Basyd Microservice Scheduling in Cloud[J]. IEEE Access.2019, 7: 83088-83100.

[29] LV L, ZHANG Y C, LI Y S, et al. Communication-aware container placement and reassignment in large-scale internet data centers [J]. IEEE Journal on Selected Areas in Communications, 2019, 37(3):540-555.

[30] 谢雍生,黄相恒,陈宁江.基于改进DQN算法的容器集群自均衡调度策略[J].计算机科学:1-10.

[31] Xiongchao Tang, Haojie Wang, Xiaosong Ma, Nosayba El-Sayed, Jidong Zhai, Wenguang Chen, and Ashraf Aboulnaga. 2019. Spread-n-share: improving application performance and

cluster throughput with resource-aware job placement[C]. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '19). Association for Computing Machinery, New York, NY, USA, Article 12, 1–15. https://doi.org/10.1145/3295500.3356152.

[32] 王康瑾, 贾统, 李影. 在离线混部作业调度与资源管理技术研究综述[J]. 软件学报, 2020, 31(10):3100-3119.

[33] Calheiros RN, Ranjan R, De Rose CAF, *et al.* CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services[J]. Computer Science, 2009.

[34] Docker Inc. Swarm mode overview[EB/OL]. https://docs.docker.com/engine/swarm/, 2022.

[35] Xiao Zhang, Eric Tune, Robert Hagmann, Rohit Jnagal, Vrigo Gokhale, and John Wilkes. 2013. CPI2: CPU performance isolation for shared compute clusters[C]. In Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys '13). Association for Computing Machinery, New York, NY, USA, 379–391. https://doi.org/10.1145/2465351.2465388.

[36] Priyanka Tembey, Ada Gavrilovska, and Karsten Schwan. 2014. Merlin: Application- and Platform-aware Resource Allocation in Consolidated Server Systems[C]. In Proceedings of the ACM Symposium on Cloud Computing (SOCC '14). Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/2670979.2670993.

[37] Michael Kerrisk. perf_event_open(2) – Linux manual page[EB/OL]. https://man7.org/linux/man-pages/man2/perf_event_open.2.html, 2021.

[38] L. Tang, J. Mars, X. Zhang, R. Hagmann, R. Hundt and E. Tune. Optimizing Google's warehouse scale computers: The NUMA experience[C]. 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA), 2013, pp. 188-197, doi: 10.1109/HPCA.2013.6522318.

[39] J. Vallone, R. Birke and L. Y. Chen. Making Neighbors Quiet: An Approach to Detect Virtual Resource Contention[J]. In IEEE Transactions on Services Computing, vol. 13, no. 5, pp. 843-856, 1 Sept.-Oct. 2020, doi: 10.1109/TSC.2017.2720742.

[40] Jun Xiao, Andy D. Pimentel, and Xu Liu. 2019. CPpf: a prefetch aware LLC partitioning approach[C]. In Proceedings of the 48th International Conference on Parallel Processing (ICPP 2019). Association for Computing Machinery, New York, NY, USA, Article 17, 1–10. https://doi.org/10.1145/3337821.3337895.

[41] M. Casas and G. Bronevetsky. Active Measurement of Memory Resource Consumption[C]. 2014 IEEE 28th International Parallel and Distributed Processing Symposium, 2014, pp. 995-1004, doi: 10.1109/IPDPS.2014.105.

[42] A. Ihre Sherif. An Evaluation of Intel Cache Allocation Technology for Data-Intensive Applications[M]. Dissertation, 2021.

[43] S. Wu, H. Sun, L. Zhou, Q. Gan and H. Jin. vProbe: Scheduling Virtual Machines on NUMA Systems[C]. 2016 IEEE International Conference on Cluster Computing (CLUSTER), 2016, pp. 70-79, doi: 10.1109/CLUSTER.2016.60.

[44] Sergey Blagodurov, Sergey Zhuravlev, and Alexandra Fedorova. 2010. Contention-Aware Scheduling on Multicore Systems[J]. ACM Trans. Comput. Syst. 28, 4, Article 8 (December 2010), 45 pages. https://doi.org/10.1145/1880018.1880019.

[45] The Gonum Authors. Gonum Numerical Packages[EB/OL]. https://www.gonum.org/, 2022.

[46] Cox DR, Cox DR, Cox DR, Cox DR. Renewal Theory[M]. Vol. 1. London: Methuen (1967).

[47] D. Breitgand and A. Epstein. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds[C]. In INFOCOM, 2012 Proceedings IEEE. IEEE, 2012, pp. 2861–2865.

[48] I. Hwang and M. Pedram. Hierarchical virtual machine consolidation in a cloud computing system[C]. In Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on. IEEE, 2013, pp. 196–203.

[49] H. Jin, D. Pan, J. Xu, and N. Pissinou. Efficient vm placement with multiple deterministic and stochastic resources in data centers[C]. in Global Communications Conference (GLOBECOM), 2012 IEEE. IEEE, 2012, pp. 2505–2510.

[50] M. Wang, X. Meng, and L. Zhang. Consolidating virtual machines with dynamic bandwidth demand in data centers[C]. In INFOCOM, 2011 Proceedings IEEE. IEEE, 2011, pp. 71–75.

[51] Github, Inc. VPA Recommender[EB/OL]. https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler/pkg/recommender, 2022.

[52] The Virtual Kubelet authors. Overview-The basic of Virtual Kubelet[EB/OL]. https://virtual-kubelet.io/docs/, 2022.

[53] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. 1991. The NAS parallel benchmarks—summary and preliminary results[C]. In Proceedings of the 1991 ACM/IEEE conference on Supercomputing (Supercomputing '91). Association for Computing Machinery, New York, NY, USA, 158–165. https://doi.org/10.1145/125826.125925.

[54] J. D. McCalpin. Stream: Sustainable memory bandwidth in high performance computers[EB/OL]. http://www.cs.virginia.edu/stream/, Feburary 2005.

[55] Marc Casas and Greg Bronevetsky. 2016. Evaluation of HPC Applications' Memory Resource Consumption via Active Measurement[J]. IEEE Trans. Parallel Distrib. Syst. 27, 9 (September 2016), 2560–2573. https://doi.org/10.1109/TPDS.2015.2506563.

[56] Daniel Molka, Robert Schöne, Daniel Hackenberg, and Wolfgang E. Nagel. 2017. Detecting Memory-Boundedness with Hardware Performance Counters[C]. In Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering (ICPE '17). Association for Computing Machinery, New York, NY, USA, 27–38. https://doi.org/10.1145/3030207.3030223.