14. **End While**

3 实验与验证

实验包括三个部分，首先是实验环境的介绍，接着对比 SAF 和 SAL 算法在事务型应用场景下，性能的差异性；最后对比两种算法在大数据处理应用场景下，吞吐率的差异性. 从而验证本方法的有效性. 其中 SAF 算法本质是公平调度，可反映 Kubernetes、Omage 等系统的资源调度效果；SAL 算法本质为延迟敏感，可反映 Apollo、Nomad 和 DelaySchedule 的资源调度效果.

4.1 实验环境与负载

如图 4 所示，实验环境由 21 台刀片机组成，每台刀片的配置都是 16 cores, 2.4GHz Intel Xeon CPU 和 24G 内存：其中 10 台刀片作为 HDFS 服务器；10 台刀片安装容器软件，用于部署容器，1 台刀片作为负载发生器，模拟用户进行压力测试，1 台刀片安装本文所述应用敏感的资源调度器. 所有刀片设备之间通过千兆交换机互联.

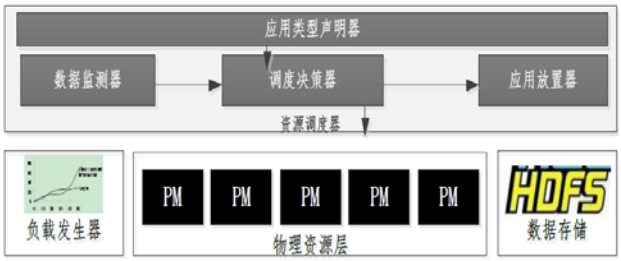

图 4 实验环境

事务型应用采用 TPC-W 基准测试，前端 Web 服务器组件选用 HttpServer 2.0，中间应用服务器组件 tomcat 8.0，后端数据库服务器组件选用 Mysql 5.6，数据库采用默认设置，即 10 000 件商品和 1 440 000 个用户. 上述应用组件均部署在 2CPU 和 2GB 内存的容器中. 其中，工作负载来源于 1998 年法国世界杯网站，分别模拟 50、175、250 和 325 并发用户，图 5 所示.

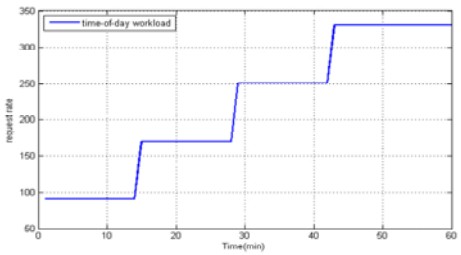

图 5 事物型负载

大数据处理应用采用 WordCount 和 Sort 两类基准测试，数据存在 HDFS 中，前者数据总大小为 10GB，后者数据总大小为 1GB，且应用组件均部署在 2CPU 和 2GB 内存的容器中. 表 1 给出了相关的配置.

表 1 Hadoop 基础测试配置表

类型	#Map	#Reduce
WordCount	5	1
	15	1
	30	1
Sort	5	1
	15	1
	30	1

4.2 事务型应用场景下本方法有效性

本方法调度事务型应用时会采用 SAL 算法，而 Kubernetes、Omage 等系统会默认采用 SAF 算法. 因此，本方法与 Kubernetes、Omage 等系统资源调度算法对比本质可转换为事务型应用在两种调度算法下的性能(响应时间刻画)差异如图 6 所示.

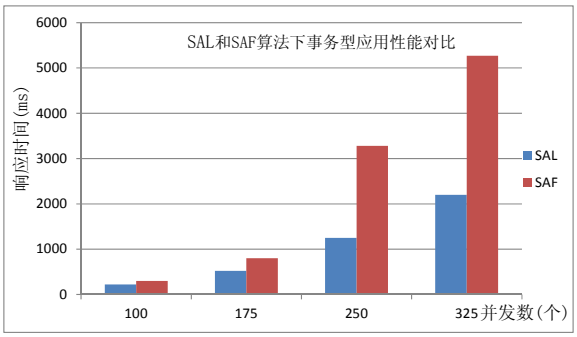

图 6 SAL 和 SAF 算法下事务型用性能对比

随着并发用户的增加，事务型应用在 SAF 算法和 SAL 算法下的响应时间的比例差异越大，最大可达到近 3 倍. 这是因为响应时间度量指标是指用户请求经过 TPC-W 测试基准 Web 服务器、应用服务器、数据库服务器处理，最终返回用户的总延迟时间总和. 采用 SAL 算法，由于应用组件调度在相同容器物理服务上，其延迟时间仅仅为 SAF 算法跨主机的网络延迟是少 1/4.

4.3 大数据处理应用场景下本方法有效性

本方法调度事务型应用时会采用 SAF 算法，而 DelaySchedule 等系统会默认采用 SAL 算法. 因此，本方法与 DelaySchedule 等系统资源调度算法对比本质可转换为事务型应用在两种调度算法下的性能(吞吐

率刻画)差异如图 7 所示.

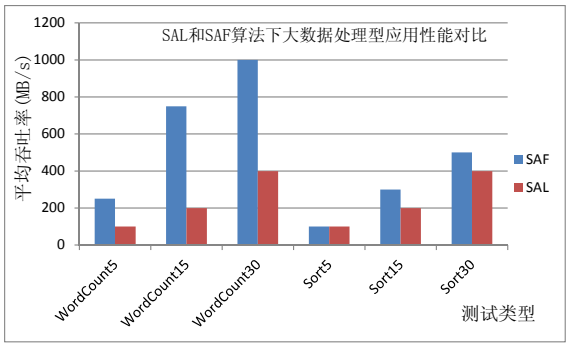

图 7 SAL 和 SAF 算法下事务型应用性能对比

无论是 WordCount 测试基准, 还时 Sort 测试基准, 随着 Map 实例数增加, Map 阶段在 SAF 算法和 SAL 算法下的吞吐率的比例差异越大, 其中 WordCount 非常明显, 最差可相差 2.5 倍. 这是因为实验环境全部是千兆网络, SAL 算法会将应用组件优先调度到相同容器物理服务器上, 而单台容器服务器理论网络吞吐率上限为 125MB/s, 会 Hadoop 应用实例会因为网络资源竞争而成为瓶颈. 仔细分析实验数据, 单个 WordCount 的 Map 阶段吞吐率约为 50MB/s, Sort 约为 20MB/s. 在 SAL 算法下, 由于 Hadoop 应用每个实例的资源配置为 2CPU 和 2GB 内存, 即每台容器服务器可部署 7 个 WordCount 实例和 Sort 实例, 故导致资源瓶颈.

4 结语

资源调度作为容器管理的关键技术之一, 已有研究工作难以同时适用大数据处理应用和事务型应用场景. 本文提出应用感知的容器资源调度方法, 其核心思想是采用多队列模型, 兼顾两种场景. 实验结果显示本方法具有有效性. 本文下一步工作拟开展基于学习的应用类型自动识别技术, 进一步提高本方法的实用性.

参考文献

1 Abraham L, Allen J, Barykin O, et al. Scuba: Diving into data at facebook. Proc. of the Vldb Endowment, 2013, 6(11): 1057–1067.

2 Ananthanarayanan G, Agarwal S, Kandula S, et al. Scarlett: Coping with skewed content popularity in MapReduce clusters. In EuroSys, 2011: 287–300.

3 Akidau T, Balikov A, Bekiro, et al. MillWheel: Fault-tolerant stream processing at internet scale. Proc. of the Vldb Endowment, 2013, 6(11): 1033–1044.

4 Andersen DG, Franklin J, Kaminsky M, et al. FAWN: A fast array of wimpy nodes. Communications of the ACM, 2011, 54(7): 101–109.

5 Petrucci V, Laurenzano M, Doherty J, et al. Octopus-man: QoS-driven task management for heterogeneous multicores in warehouse-scale computers. 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). IEEE Computer Society. 2015. 246–258.

6 Aksanli B, Venkatesh J, Zhang L, et al. Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. Association for Computing Machinery, 2011: 53–57.

7 Marchukov M, Song YJ, Bronson N, et al. TAO: Facebook's distributed data store for the social graph. Proc. of the 2013 USENIX Conference on Annual Technical Conference. USENIX Association. 2013. 49–60.

8 Baker J, Bond C, Corbett J, et al. Megastore: Providing scalable, highly available storage for interactive services. 5th Biennial Conference on Innovative Data Systems Research (CIDR '11). Asilomar, California, USA. 2011. 223–234.

9 Baumann A, Barham P, Dagand PE, et al. The multikernel: A new OS architecture for scalable multicore systems. ACM Sigops, Symposium on Operating Systems Principles. ACM. 2009. 29–44.

10 Belay A, Bittau A, Mashtizadeh A, et al. Dune: Safe user-level access to privileged CPU features. Symposium on Operating Systems Design & Implementation (OSDI). 2012. 335–348.

11 Schwarzkopf M, Konwinski A, Abd-El-Malek M, et al. Omega: Flexible, scalable schedulers for large compute clusters. ACM European Conference on Computer Systems. ACM. 2013. 351–364.

12 Tune E. Large-scale cluster management at Google with Borg. Proc. of the 10th European Conference on Computer Systems. ACM. 2015. 18.