Journal of Computer Applications
计算机应用,2013,33(10):2792-2795
ISSN 1001-9081
CODEN JYIIDU
2013-10-01
http://www.joca.cn

文章编号:1001-9081(2013)10-2792-04
doi:10.11772/j.issn.1001-9081.2013.10.2792

Hadoop 视角下的 Nutch 爬行性能优化

周世龙, 陈兴蜀*, 罗永刚
(四川大学 计算机学院, 成都 610065)
(*通信作者电子邮箱 chenxs@scu.edu.cn)

摘　要:通过对 Nutch MapReduce job 配置参数调优而优化 Nutch 爬行性能。以 Hadoop 视角梳理 Nutch 爬行过程,并基于此详细分析 Nutch MapReduce job 的工作流特性;对 Nutch 爬行时 MapReduce job 进行持续监测,生成优化参数并代入下一轮相同类型的 job 运行中,从而达到优化目的;通过选取合适的间隔监测值平衡集群环境误差和监测负载以改进优化效果。经过实验测试,Nutch 的爬行性能提高了 5% ~14%,且当监测间隔值为 5 时有最好优化效果。

关键词:Nutch;Hadoop; MapReduce; 工作流; 性能优化
中图分类号:TP393.4　　**文献标志码**:A

Nutch crawling optimization from view of Hadoop

ZHOU Shilong, CHEN Xingshu*, LUO Yonggang
(*College of Computer Science, Sichuan University, Chengdu Sichuan* 610065, *China*)

Abstract: Nutch crawling performance was optimized by tunning Nutch MapReduce job configurations. In order to optimize Nutch performance, firstly Nutch crawling processes were studied from the view of Hadoop. And based on that, the characters of Nutch jobs' workflows were analyzed in detail. Then tunned job configurations were generated by profiling Nutch crawling process. The tunned configurations were set before the next job running of the same type. The appropriate profiling interval was selected to consider the balance between cluster environmental error and profiling load, which improved optimization result. The experimental results show that it is indeed more efficient than the original programs by 5% to 14%. The interval value of 5 makes the best optimization result.

Key words: Nutch; Hadoop; MapReduce; workflow; performance optimization

0　引言

Nutch[1]是基于 Lucene 的开放源码的 Web 搜索引擎,基于 Hadoop 实现分布式 Web 页面的爬行、索引以及检索工作。

詹恒飞等[2]对通过修改 Nutch 在抓取 Web 页面时的默认等待时间及抓取行为等方式,达到优化 Nutch 的目的。NutchWiki[3]分析了 Nutch 从互联网抓取 Web 页面时,11 种可能影响 Nutch 抓取速度的因素,并给出了相应的调优策略。文献[2-3]均从 Nutch 的爬行策略的角度对 Nutch 性能进行调优,但调优的角度局限在了 Nutch 爬行的业务逻辑本身。

Intel 公司的技术白皮书[4]以实验的方式描述了不同集群环境以及相关参数配置对 Hadoop job 性能的影响。Impetus 公司的技术白皮书[5]分析了 Hadoop 配置参数将直接影响 MapReduce job 的执行性能,并分析了相关的调优案例。Hansen 等[6]以实验的方式,对 Hadoop 中典型的 MapReduce job 进行参数调优分析。文献[4-6]表明 Hadoop 配置参数可以影响 Hadoop job 运行性能。但以上文献均定性分析配置参数对 job 的影响,并没有定量描述 job 性能和参数的具体对应关系。对 Hadoop 进行调优仍需要丰富的相关知识和较多的反复参数调优的相关知识。

Herodotou 等[7]提出基于代价的(Cost-based)Hadoop MapReduce job 运行参数调优工具 Starfish,对 job 进行监测并在 job 再次运行时通过生成针对当前 job 的优化参数,达到优化的目的。但 Starfish 仅作为工具针对一般 Hadoop job 的单次运行进行参数调优,应用受到局限。

本文立足于 Hadoop 视角,详细分析 Nutch 的 Hadoop 工作流特性,并基于 Starfish 对 Nutch 爬行时 MapReduce job 参数进行调优,从而提高 job 运行性能,达到优化 Nutch 爬行性能的效果。

1　Nutch 爬行优化方案

Hadoop 性能调优工具 Starfish 无法针对 Nutch 爬行过程中所产生的连续 job 进行应用。本文在分析 Nutch 的 MapReduce job 的工作流特征后,对 Starfish 模块进行拆解并与 Nutch 集成,从而最终达到对 Nutch 进行优化的目的。

1.1　MapReduce 性能优化

Starfish 是 Herodotou 等[8]开发的一个基于代价的用于 Hadoop MapReduce job 数据监测、job 运行模拟以及参数性能调优的开源工具。

对于一个 Hadoop MapReduce job,$j = \langle p,d,r,c\rangle$ 它的性能可以表示为

$$perf = F(p,d,r,c) \tag{1}$$

job 优化是指基于特定代价模型 F,对 MapReduce 程序 p,在指定的输入数据 d 和集群资源状况 r 的条件下,寻找 job 执行配置参数 c,使得整个 Map Reduce job 性能 $perf$ 达到一个近似最优值。

收稿日期:2013-03-25;**修回日期**:2013-05-22。　**基金项目**:科技部支撑项目(2012BAH18B05)。

作者简介:周世龙(1988-),男,山西平定人,硕士研究生,CCF 会员,主要研究方向:云计算安全、云计算中大规模数据处理; 陈兴蜀(1968-),女,四川成都人,教授,博士生导师,博士,主要研究方向:信息安全、计算机网络、云计算; 罗永刚(1980-),男,贵州黔南人,博士研究生,主要研究方向:信息安全、云计算安全。

Starfish 包含三个功能组件:

1)监测器(Profiler)。监测器基于 BTrace[9] 在 job 运行时监测和收集 job profile 数据。job profile 数据包含了一个 MapReduce job 各阶段的数据流信息及其相应的时间和资源耗费代价。

2)条件假设(What-if Engine)引擎。What-if 引擎在给定 $j = \langle p,d,r,c\rangle$ 的情况下,可以预测当 job 的输入数据大小变为 d',集群子节点数目变为 r' 以及 job 运行配置参数变为 c' 时,job $j' = \langle p,d',r',c'\rangle$ 的执行时间。

What-if 引擎预测 job 的执行时间基于 Hadoop Performance Model (HPM)[10]。HPM 以非常细的粒度对 MapReduce job 整个工作流的各个阶段进行数学建模。

当已知 $j = \langle p,d,r,c\rangle$ 的 job profile 数据以及 d'、r' 和 c' 时,What-if 引擎根据 HPM 中模型生成虚拟 job $j' = \langle p,d',r',c'\rangle$ 的 job profile 数据,随即将此数据代入调度器模拟器中运行,最后计算出虚拟 job $j' = \langle p,d',r',c'\rangle$ 的运行时间。

3)基于代价的优化器(Cost-Based Optimizer,CBO)。CBO 利用递归随机搜索(Recursive Random Search,RRS)算法[11] 以及 What-if 引擎完成对指定 job 配置参数的优化。

CBO 利用 RRS 算法对 job 的参数进行多组的随机枚举,随后将枚举的每一组参数分别代入 What-if 引擎,计算 job 在当前参数配置下的执行时间,最后选择使得 job 执行时间最少的一组配置参数作为 job 参数的优化结果。目前 CBO 可以支持对 11 个 MapReduce job 配置参数的优化。

1.2　Nutch 爬行的 MapReduce 工作流特征

Nutch 主要由爬行、索引和查询三个功能组件构成[12]。本文主要对 Nutch 爬行组件进行研究与性能优化。

Nutch 基于 Hadoop 实现分布式爬行,以 Hadoop 视角来看,Nutch 在爬行过程的各个阶段会向 Hadoop 提交一系列 MapReduce job,这些 job 具有明显的循环特性以及严格的时序关系,本文称以上特性为 Nutch 爬行的工作流特性。

Nutch 爬行各个阶段产生的 job 如下:

1) 产生并注入种子 URL 列表(Inject)。将种子 URL 格式化,过滤并消除非法 URL。设置 URL 状态为 unfetched 并注入爬行数据库(CrawlDB)。Inject 阶段产生 2 个 job,本文定义为 InjectA job 和 InjectB job。

2) 生成待爬行列表(Generate)。读取 CrawlDB 中数据,并生成需要从互联网上抓取的链接信息。Generate 阶段产生两个 2 个 job,本文定义为 GenerateA job 和 GenerateB job。

3) 抓取 Web 页面(Fetch)。根据待爬行列表从互联网上抓取链接相对应的 Web 源页面。Fetch 阶段产生 1 个 job,本文定义为 Fetch job。

4) 解析抓取内容(Parse)。对抓取的 Web 页面进行分析处理。Parse 阶段产生 1 个 job,本文定义为 Parse job。

5) 更新爬行数据库(UpdateDB)。根据 Fetch 阶段和 Parse 阶段产生的内容更新 CrawlDB 中的 URL 状态信息。此阶段产生 1 个 job,定义为 UpdateDB job。

6) 更新链接数据库(Update LinkDB)。Nutch 爬行阶段结束后,更新链接数据库并建立索引信息。这一阶段产生 2 个 job,定义为 LinkDBA 和 LinkDBB job。

Nutch 爬行时的工作流循环如图 1 所示。

循环特性　Nutch 在对互联网进行爬行时会进行多轮爬行,本文将爬行的轮次称作爬行深度。当 Nutch 爬行过程中没有到达指定爬行深度时,会不停循环执行 2) ~5) 四个阶段中的 5 个 job。每完成一次循环,则进入下一个爬行深度。当达到指定爬行深度后,Nutch 更新链接数据库,完成爬行工作。

时序性　Nutch 爬行中所产生的一系列 job 中,当前一个 job 结束后,后一个 job 才会开始。Nutch 爬行时产生的 job 具有严格的时序性。如果 Hadoop 集群只用来 Nutch 爬行时,那么在任意时刻,集群中有仅有唯一的 Nutch job 在运行。

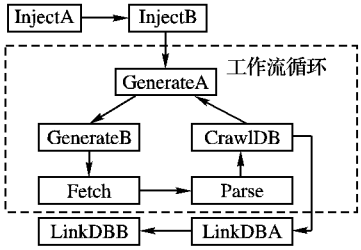

图 1　Nutch 工作流循环

由 Nutch 工作流特性可知:

1) Nutch 爬行过程会向 Hadoop 提交的一系列 job,由循环特性可知爬行 job 会重复运行,适用于 Starfish 的调优原理。

2) Nutch 工作流的时序性决定在爬行过程中的任意时刻,集群中仅运行唯一的 MapReduce job,因此对 job 进行监测不会受到来自其他 job 运行对集群监测环境的影响,使 Starfish 监测的数据不受干扰。

Nutch 工作流的两个特性符合 Starfish 进行参数优化的条件。本文利用 Starfish 针对 Nutch 爬行 job 进行参数调优,最终达到优化 Nutch 的目的。

1.3　持续优化模块的设计与实现

持续优化通过对 Nutch 当前爬行的 job_i 进行监测,搜集运行时数据(job profile 数据)进行暂存。然后使用暂存的 job profile 数据生成优化后参数,替换新一轮的相同类型的 job_{i+1} 的默认配置参数,从而使 job_{i+1} 以优化参数运行,完成 Nutch 的优化。

$$P_i = profile(job_i) \tag{2}$$

$$conf_{i+1} = opt(P_i, d_{i+1}, r, p) \tag{3}$$

$$job_{i+1} = run(conf_{i+1}, d_{i+1}, r, p) \tag{4}$$

式(2):对于特定类型的 Nutch job 的运行,对 job 运行时的数据进行监测并暂存,P_i 表示 job_i 的运行时捕获生成的 job profile 数据。

式(3)、(4):当此类 job 在下一轮运行前,根据本次 job profile 数据生成相应的优化参数替换原有默认参数并运行,完成 job 的优化。$conf_i$ 表示基于 job 的 profile 数据、job 下一次执行的输入数据、集群当前资源配置情况以及 MapReduce 程序生成的优化后的参数。表 1 对比了 parse job 优化前后部分配置参数的变化。

表 1　parse job 优化参数对比

参数名称	默认值	优化后
mapred. inmem. merge. threshold	1 000	151
io. sort. factor	10	82
io. sort. mb	100	124
mapred. job. shuffle. merge. percent	0.66	0.690 8

表 1 列举了 parse job 运行时部分配置参数的默认值和经过 Starfish 生成的优化后的配置参数值,可以看出前后参数的变化。不同阶段的 job 生成的配置参数优化值不同,同一个 job 每次优化时生成的 job 配置参数优化值不同。

Nutch 优化的模块设计如图 2。

当特定 job 运行时,数据监测模块监测并捕获 job 运行时数据,标记 job 类型并将生成的 job profile 数据保存,供以后

使用。

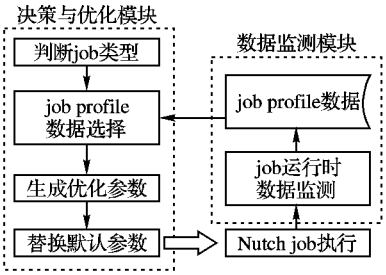

图 2 优化模块设计

当特定 job 运行前,决策与优化模块根据当前运行 job 的类型寻找相同类型的 job profile 数据,调用 CBO 优化器生成优化配置参数,最后替换 job 运行时的默认参数。

2 优化测量与数据分析

为了测量优化效果,本文通过对 Nutch 在默认配置下和优化以后爬行相同网页数量所消耗的时间进行对比。

2.1 实验环境

本文的实验平台为基于 Xen 的虚拟化平台。集群拥有 1 个主节点,3 个从节点。集群节点配置均为:Intel Xeon 2.13 GHz双核处理器,内存 4 GB,网络带宽 100 Mb/s,每台主机均有独立 IP 地址,并通过校园网接入教育科研网。

实验爬虫为 Nutch1.4 版本,实验对比原版以及优化后 Nutch 的爬行时间。爬行种子 URL 相同,爬行深度为 50 轮,每轮爬行一万个 Web 页面,总共爬行 50 万个页面。

2.2 持续监测优化实验及其分析

2.2.1 实验数据对比

实验对比 Nutch1.4 原版和优化版本的爬行时间,测试结果如图 3 所示。

图 3 对比了优化前(Def)Nutch 的爬行和持续监测优化(Opt)的效果。可以看出,当爬行到 50 万个页面时,优化后的 Nutch 比优化前节省了 4 h 左右的时间。因此,Hadoop 视角下针对 Nutch 生成优化参数的方式进行优化,取得了一定效果,Nutch 爬行时间缩短了 5%。

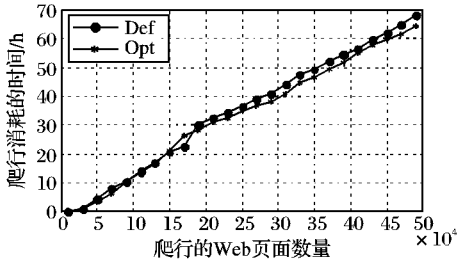

图 3 默认与优化版本 Nutch 爬行时间对比

2.2.2 监测代价分析

实验发现,优化虽然取得了一定的效果,但对 Nutch 爬行时间的减少并不是特别理想。

由于 Starfish 基于 BTrace 来完成对 job 的动态监测。BTrace 动态拦截 job 在运行时的 Java 字节码,并向其注入额外的用于完成监测功能的字节码。而动态地对 Java 字节码进行拦截和注入给 job 执行带来额外的负荷,影响优化的效果。

2.3 间隔监测优化

2.3.1 间隔监测设计

由于 BTrace 进行动态监测会导致爬行性能的下降,因此本文尝试降低监测 Nutch 的频度,在工作流循环中的后续相同类型的 job 均使用前几轮监测的数据结果进行优化参数的生成。

间隔监测持续优化算法伪代码如下:

```
m ← profiling interval;                    //m 为监测的间隔值
i ← 0;
WHILE i < crawl depth
  get current nutch job type;
  get job profile data for current nutch job;
  optConf ← performOptimize()
  replace default conf with optConf;
  IF i mod m = 0 THEN
    run job with profiling;
    save job profile data;
  ELSE
    run job without profiling;
  END IF
  i++;
END WHILE
```

Nutch 每完成一个爬行深度时便完成一个工作流循环。当针对本次 job 生成优化参数后,程序判断本次爬行的 job 是否应该进行监测,如果需要监测,则当 job 运行结束之后保存监测的 job profile 数据。

2.3.2 间隔监测实验对比

本文设置了不同的监测间隔值 m,并进行对比爬行实验,图 4 给出了在不同监测间隔值的条件下,Nutch 爬行 50 万个 Web 页面所消耗的时间。

图 4 分别对监测间隔值为 1(It 1)、3(It 3)、5(It 5)、10(It 10)、20(It 20)这几种情况进行了爬行对比,可以看出当间隔值为 5 时,优化具有最好的效果。

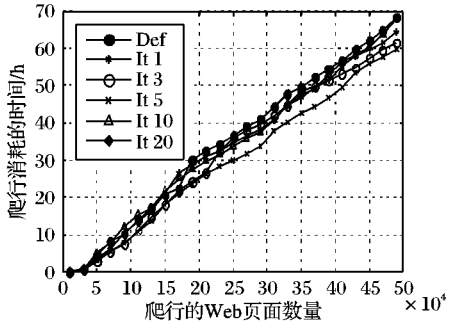

图 4 不同间隔监测值爬行时间对比

2.3.3 集群环境误差影响分析

由图 4 分析可知,在本文实验集群环境下,当间隔值 m 为 5 时优化效果最好而不是 20,这是因为过于稀疏的监测采样可能导致监测误差,影响优化参数生成。

分析 Nutch job 的运行过程中,由于 job 输入数据的大小和实际的存储位置、从节点间带宽的变动等因素以及 BTrace 本身对于数据监测的准确性等原因,使得监测 job 产生的 job profile 数据具有一定的误差。本文将以上引起误差的因素称之为集群环境误差,监测间隔值越大,误差可能越大。

为了观测运行集群环境迁移误差对 Profile 数据的影响,在 Nutch 连续爬行时分析 GenerateA job 的 profile 数据,对 HDFS 读入数据的速率和读取的数据总量进行了绘制,如图 5 所示。

图 5 中横坐标表示 GenerateA job 的执行次数,Nutch 每爬行一轮(完成 1 万个 Web 页面的爬行),该 job 执行一次。在最初的几轮爬行时,GenerateA job 的输入数据很小(刚开始爬行的时,CrawlDB 仅包含 Inject 阶段注入的种子 URL 信息),MapReduce 框架本身的运行负荷便会对读取一个字节时间的计算产生较大影响。而随着爬行的进行,CrawlDB 的增大,GenerateA job 的输入数据也逐步变大,虽然读取单个字节

记录所消耗的时间总体稳定,但是依旧上下起伏。

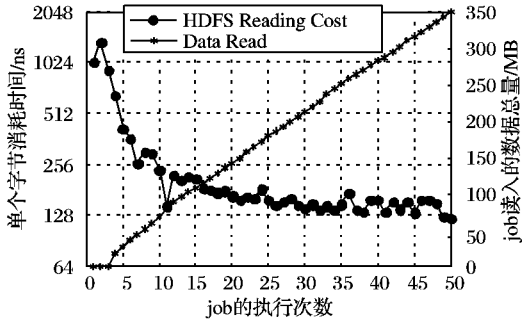

图 5　GenerateA job HDFS Read 代价

而由 1.1 节可知,Starfish 针对 job 产生优化数据依赖于 What-if 引擎,What-if 引擎使用 HPM 完成对 job 执行的各个阶段的时间代价的预测。因此, job profile 数据将最终影响 Starfish 针对 job 产生优化参数的值。由此可知,Nutch 的运行环境误差将会影响其后续 job 运行前优化参数的生成,最终影响优化效果。

2.4　实验总结

由实验可知,立足于 Hadoop 视角通过对 Nutch 运行时 Hadoop MapReduce job 优化参数进行生成的方式对 Nutch 进行优化,取得了一定效果。

同时优化效果受到 job 监测负荷和运行时环境误差的共同影响,监测间隔值的选择将很大程度上影响优化的效果。在本文实验集群环境下,当间隔值为 5 时,优化效果最好,Nutch 的爬行性能提高了 14%。

3　结语

本文分析了 Nutch 爬行时产生的 job 工作流特征,以 Hadoop 视角,基于 Starfish 针对 Nutch 运行时 job 生成优化参数,提高了 Nutch 的爬行性能。

但是依然存在较多问题,比如监测的间隔值需要进行不断尝试才能找到较优值,修改后的 Nutch 版本部署较为繁琐以及 Starfish 本身调优的 job 配置参数数量有限等。下一步工作将会针对目前优化中所存在的问题进行解决。

参考文献:

[1] The Apache Software Foundation. Apache Nutch™ [EB/OL]. [2012-12-11]. http://nutch.apache.org.

[2] 詹恒飞, 杨岳湘, 方宏. Nutch 分布式网络爬虫研究与优化[J]. 计算机科学与探索, 2011, 5(1): 68 - 74.

[3] DennisKubes. NutchWiki[EB/OL]. [2009-11-24]. https://wiki.apache.org/nutch/OptimizingCrawls.

[4] Intel. Optimizing Hadoop* deployments[EB/OL]. [2010-10-08]. http://communities.intel.com/servlet/JiveServlet/downloadBody/5645-102-1-8759/Optimizing Hadoop_201 0_final.pdf.

[5] Impetus Technologies Inc. Hadoop performance tuning[EB/OL]. [2010-11-16]. http://hadoop-toolkit.googlecode.com/files/White paper-HadoopPerformanceTuning.pdf.

[6] HANSEN C A. Optimizing Hadoop for the cluster[EB/OL]. [2010-04-17]. http://www.scratchmytail.com/papers/cha030-optimizinghadoop.pdf, 2012.

[7] HERODOTOU H, LIM H, LUO G, *et al*. Starfish: A self-tuning system for big data analytics[EB/OL]. [2013-01-08]. http://x86.cs.duke.edu/~gang/documents/CIDR11_Paper36.pdf.

[8] HERODOTOU H, BABU S. Profiling, what-if analysis, and cost-based optimization of MapReduce programs[J]. Proceedingso of the VLDB Endowment, 2011, 4(11): 1111 - 1122.

[9] Oracle Corporation. A dynamic instrumentation tool for Java[EB/OL]. [2013-02-08]. http://kenai.com/projects/btrace.

[10] HERODOTOU H. Hadoop performance models[EB/OL]. [2013-01-20]. http://www.cs.duke.edu/starfish/files/hadoop-models.pdf.

[11] YE T, KALYANARAMAN S. A recursive random search algorithm for large-scale network parameter configuration[J]. ACM SIGMETRICS Performance Evaluation Review, 2003, 31(1): 196 - 205.

[12] DMOREIRA J E, MICHAEL M M, DA SILVA D, *et al*. Scalability of the Nutch search engine[C]// Proceedings of the 21st Annual International Conference on Supercomputing. New York: ACM, 2007: 3 - 12.

(上接第 2782 页)

客户端的终端用户与云中心交互频繁,数据传输量大,对实时性要求高,网络速度严重限制瘦客户端的分布式虚拟维修仿真;3)云仿真平台的军事数据、信息安全必须加以高度重视。

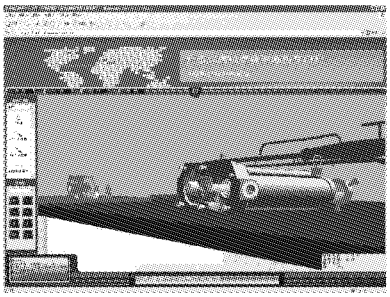

图 9　装备网络化仿真训练开发实例

参考文献:

[1] 郭齐胜,张伟,杨立功. 分布交互仿真及其军事应用[M]. 北京:国防工业出版社,2003:21 - 23.

[2] 陈全, 邓倩妮. 云计算及其关键技术[J]. 计算机应用,2009,29(9):2562 - 2567.

[3] 李伯虎,柴旭东,侯宝存,等. 一种基于云计算理念的网络化建模与仿真平台——"云仿真平台"[J]. 系统仿真学报,2009,21(17):5292 - 5299.

[4] 杜瑾. 云计算在军事仿真中的应用研究[J]. 电脑知识与技术,2010,6(25):6995 - 6997.

[5] 张雅彬, 李伯虎, 柴旭东,等. 基于虚拟化技术的云仿真资源迁移技术研究[J]. 系统仿真学报,2011,23(6):1268 - 1272.

[6] 张雅彬, 李伯虎, 柴旭东,等. 基于虚拟化技术的云仿真运行环境动态构建技术[J]. 系统工程与电子技术,2012,34(3):619 - 624.

[7] 华翔, 康凤举, 田学伟,等. 可视化仿真的私有云框架研究[J]. 系统仿真学报,2011,23(8):1652 - 1656.

[8] 杨晨, 李伯虎, 柴旭东,等. 面向云制造的云仿真支撑框架及应用过程模型[J]. 计算机集成制造系统,2012,18(7):1444 - 1451.

[9] 高武奇, 康凤举, 钟联炯,等. 一种基于 HLA Evovled 的云仿真技术研究[J]. 系统仿真学报,2011,23(8):1643 - 1646.

[10] JONES T. Anatomy of the libvirt virtulization library[EB/OL]. [2010-05-06]. http://www.ibm.com/developerworks/linux/library/l-libvirt/index.html.

[11] VMware. Vmware Workstation[EB/OL]. [2010-06-08]. http://www.vmware.com/cn/products/workstation/new.html.

[12] KATHERINE L M, DAVID L D, RYAN P Z B. Web enabling HLA compliant simulations to support network centric applications [EB/OL]. [2013-01-20]. http://www.dodccrp.org/events/2004_CCRTS/papers/172.pdf.

[13] 刘鹏. 云计算[M]. 北京: 电子工业出版社, 2012:251 - 252.