面向云计算的粒度自配置日志采集平台

冯禹洪[1] 吴晓峰[1] 白鉴聪[1] 陈国良[1] 明 仲[1] 张建华[2] 甘玉玺[3] 陈 沛[3]

[1](深圳大学计算机与软件学院 广东省普及型高性能计算机重点实验室 深圳 518060)
[2](华为技术有限公司云计算与数据中心 深圳 518129)
[3](中兴通讯股份有限公司云计算与政企业务产品部 深圳 518057)

摘 要 日志数据管理系统是最重要的云服务基础设施之一。重要日志数据缺失将造成相应日志分析与决策的片面性和不准确性。然而日志数据采集能力越强，日志采集的运行期开销就越大，海量日志数据的管理与分析就越耗时，对整个云服务环境的系统性能造成不可忽视的影响。针对如何采集必要的日志数据同时尽可能降低其运行期开销的问题，文章首先提出日志采集粒度的概念，然后设计并编程实现一个面向云计算的粒度自配置日志采集平台。其中，平台构成模块包括：日志采集工具、存储日志采集粒度规则和事实的知识库；基于规则动态增加或关闭相关日志数据采集模块的推理机；相应的图形界面，包括用于添加或修改知识库规则的管理界面和直观查看日志数据的用户界面。最后，初步的案例学习结果表明了平台的有效性。

关键词 日志采集；推理机；知识库；云计算；淘宝开源系统 Tsar
中图分类号 TP 319 **文献标志码** A

A Grain-Level Self-Configuring Log Data Capturing Platform for Cloud Computing

FENG Yuhong[1] WU Xiaofeng[1] BAI Jiancong[1] CHEN Guoliang[1] MING Zhong[1]
ZHANG Jianhua[2] GAN Yuxi[3] CHEN Pei[3]

[1](*School of Computer Science and Software Engineering, Guangdong Province Key Laboratory of Popular High Performance Computers, Shenzhen University, Shenzhen* 518060, *China*)
[2](*Cloud Computing & Data Centers, Huawei Technologies Co. Ltd., Shenzhen* 518129, *China*)
[3](*Government & Enterprise Solution Department, ZTE Corporation, Shenzhen* 518057, *China*)

Abstract The log data management system is one of the key infrastructures for cloud computing. Missing of important log data leads to inaccurate and one-sided data analysis and decision making. However, the stronger the log data capturing capability is, the higher the runtime overhead is. In order to capture necessary log data and reduce the runtime overhead as much as possible, this paper first put forward the log data capturing grain level concept was put forward firstly in this paper, and a grain-level self-configuring log data capturing platform was designed then for cloud computing. This platform is

收稿日期：2014-3-17
基金项目：国家自然科学基金（61103001、 61170077、 61272445 和 61202377），2012 年广东省大学生创新实验项目（0000175782），深圳市科技计划（JCYJ20120613102030248）。
作者简介：冯禹洪，博士，副教授，研究方向为工作流管理、云计算或物联网和中间件；吴晓峰，学士，研究方向为智能监控系统；白鉴聪，博士，讲师，研究方向为云计算和智能优化算法；陈国良，教授，博士生导师，中国科学院院士，研究方向为并行算法和高性能计算及其应用等；明仲（通讯作者），博士，教授，研究方向为面向对象软件工程和形式化方法、云计算或物联网和工作流，E-mail：mingz@szu.edu.cn；张建华, 博士，研究方向为云计算和虚拟化技术等；甘玉玺，研究方向为 IP 数据通信、轨道交通通信系统和云计算；陈沛，研究方向为云计算和 IDC。

consisted of a log data capturing tool, a knowledge base storing grain-level based log capturing rules and facts, a rule-based inference engine for adding and removing specific log data capturing modules, and graphical interfaces for managing the knowledge base and querying log data sets. Finally, our preliminary case study demonstrates the efficiency of our platform.

Keywords　log data capturing; inference engine; knowledge base; cloud computing; Tsar, the open source system of Taobao

1 引　言

云计算通过互联网以服务的方式提供动态可伸缩的虚拟化资源，使人们像用电一样享用信息的应用和服务，相应的信息系统由专业的服务商提供。云服务提供商如 Google、腾迅和 Facebook 等常采用在海量硬件上开发功能强大的软件来提供多种多样的云服务。复杂的云计算环境有着常态的故障事件，据 Google 在 2008 年的数据统计， 在一个拥有 1000 台机器的集群中，平均每天有 1 台坏掉[1]。

常态的故障是云服务的可靠性和可用性的巨大挑战。且由于云服务的自动化供给特性，即使是云环境中的一个小错误也可能产生扩大连锁反应，导致难以定位和恢复的系统故障事件。如 2011 年 4 月 21 日的 Amazon EC2 失效事件，就是因为 Amazon 美东地区服务区的一位维护人员不小心弄错了一项网路设定，导致该区 4 座资料中心的 EC2 服务严重大宕机，众多知名网站受影响甚至不可用，最终历时 38 小时才将该故障消除并恢复使用。最近，卡内基梅隆大学对 M45 超级计算集群连续跟踪 10 个月发现，大多数作业将在第一个终止任务出现后的 150 秒内终止。然而，人们观察到的最大错误延迟竟有 4.3 天[2]。这些错误的发现和定位延迟与云服务的高可用性和可靠性要求相矛盾，故云计算环境迫切需要更好的监控、诊断和恢复方法。

日志数据专门记录系统操作事件，包括基础设施、操作系统、应用和服务等不同层次的操作。日志数据被广泛用于诊断和解决系统问题，如通过学习错误和失败的日志，预测关键事件避免灾难性失效、确定触发系统故障的事件或确定可靠性瓶颈[3,4]。因此，日志数据管理也是最重要的云服务基础设施之一。通过分析日志、云服务业主和经营者可以了解每个基础设施组件的状态和监测业务流程；云软件开发人员可以检测和排除系统故障、监控系统性能，从而有效地提高和简化程序员的开发工作；运维和保安人员可以通过历史数据取证等。

目前云日志管理及其应用的重要性已得到广泛的关注[2]，并开展了大量相关工作，如大型集群系统日志事件关联性的分析[5]。同时也涌现出许多大规模分布式云日志记录和收集系统，如 Yahoo 的 Chukwa[6]、Facebook 的 Scribe[7]、LinkedIn 的 Kafka[8]、Cloudera 的 Flume[9,10]和淘宝的 Tsar[11]。这些日志系统为“分布式收集，统一处理”提供了一个可扩展的、高容错的方案。但我们调研发现，目前系统的日志采集能力往往是在启动前通过配置文件指定，一经启动就不再改变，这类一体式日志采集工具的性能在云计算环境下将受到新的挑战。

云计算一般提供 3 个服务模型：基础设施即服务（IaaS）、平台即服务（PaaS）以及软件即服务（SaaS）[12]，有着业务高多样化和环境高动态性等特点。相应地，各类服务需要采集的日志也不一样。部署提供 IaaS 和 SaaS 的系统操作有差异，相应的日志数据亦有差别。采用通用的日志采集平台，其日志采集需要覆盖所有的服务类型。日志数据采集能力是指其能采集的数据范围和更新

频率，能力越强，能采集到的服务日志也就越多。频繁的日志更新和海量的日志管理将带来不可忽略的运行期开销。而重要日志数据缺失，相应的日志分析与决策将是片面的、不够准确的。因此如何限制日志采集的运行期开销在一个可以接受的范围是一个有意义的课题。

本文首先提出日志采集粒度的概念，然后设计并编程实现一个面向云计算的粒度自配置日志采集平台。其中，平台构成模块包括：日志采集工具、存储日志采集粒度规则和事实的知识库；基于规则动态增加或关闭相关日志数据采集模块推理机；根据运行期系统状态自动调整采集粒度，从而有效地降低正常服务的运行期开销，并及时加细采集粒度捕捉导致故障服务的事件。文章最后通过初步的案例学习来验证平台的有效性。本文接下来的内容如下安排：第 2 节将详细介绍云计算系统体系结构及相应的日志数据，并进一步解释为什么云计算环境需要粒度自配置日志采集平台；第 3 节详细介绍我们的平台及其动态配置日志数据采集粒度的实现机制；第 4 节描述我们的原型系统实现及初步的案例学习，验证平台的可行性和有效性；第 5 节总结全文并展望未来。

2 云日志概述

图 1 为云计算的日志采集平台系统。如图 1 左边所示，云计算的系统体系结构一般可分为四个层次。自下向上，最底层为云机房，一般配有机柜、供电、制冷、消防和防雷等辅助基础设施，用于管理和存放云计算所需要的硬件设施。第二层为硬件基础设施层，包括服务器、存储、网络和安全设施如防火墙等。大型云服务的数据中心的硬件设备高达 6 位数，且绝大部分服务器采用普通的 PC 服务器。第三层为云系统软件和支撑软件，包括操作系统、集群管理、并行分布式处理和服务编程接口（API）等。这些软件为实现上层服务开发与可靠供给提供相应的系统支持。最上面一层为各云服务商提供的云服务，

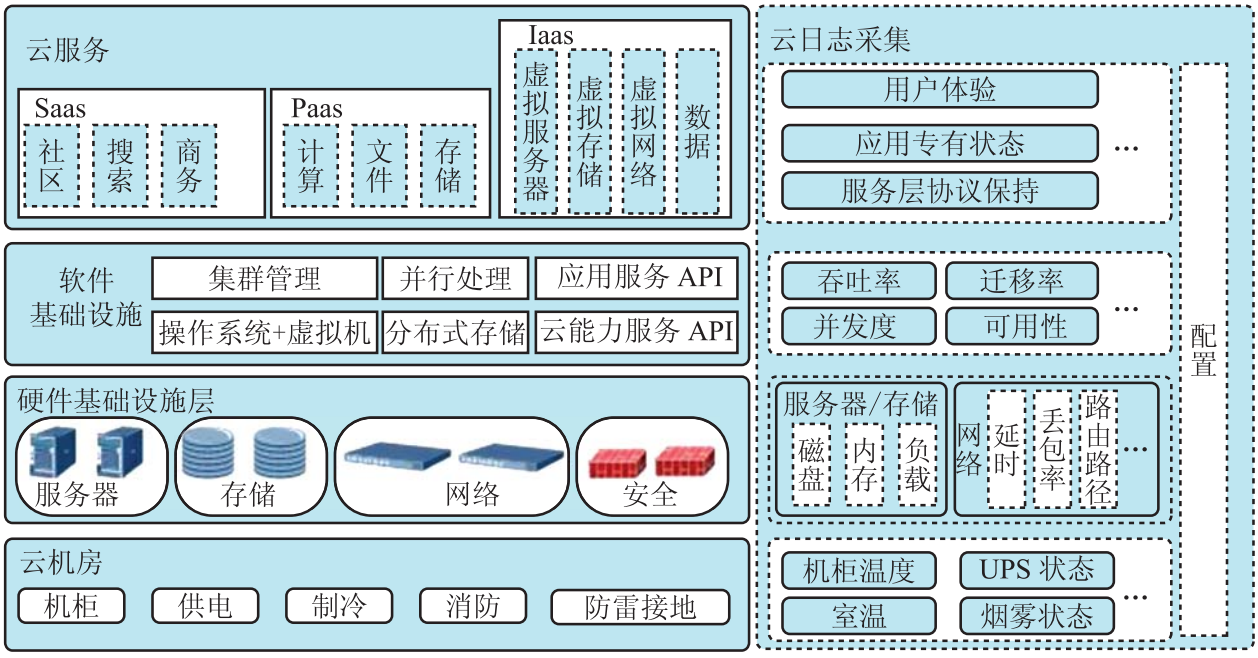

图 1 面向云计算的日志采集平台系统图

Fig. 1. The system model for capturing log data over the cloud

包括 SaaS、PaaS 和 IaaS。云服务多种多样，如 SaaS 包含社区、搜索和商务等。而社区服务又可以进一步细分为教育云、医疗云、娱乐云和生活云等。但云服务仍面临需求不断增加的挑战[13]。

基于上面的描述，我们可以看出，复杂的云计算环境有着复杂的软件、数据和硬件依赖关系，同时存在错综复杂的常态故障事件，包括硬件故障、网络故障和软件故障等。其中，硬件故障如前面所说的 Google 的一个 1000 台机器的集群中，平均每天坏掉 1 台；网络故障包括网络设备故障、不正当的网路配置造成的服务无法访问现象以及业务突发情况，如黄金周后大量照片上传所带来的服务器或网络带宽超载故障等；软件故障包括软件的 bug、不正当的参数配置和协作故障如死锁等所带来的服务无法访问现象。

一般来说，为有效预防和排除故障提供充足的事实依据，云日志采集平台需要对云体系结构的各个层次各组件的状态数据和操作事件等进行采集和记录。图 1 右边为相应于云计算体系结构的各个层次，云日志采集工具需要采集的日志数据。首先相对于最底层，日志采集系统需要监控机柜温度、室温、电压和烟雾状态，及时捕获温度过高或电压电流不足事件等，确保机房硬件设施的物理安全。硬件基础设施层需要采集各节点的内存、磁盘和负载，以及网络的性能等状态，及时发现硬件的出错和网络故障。软件基础设施层需要检测系统的吞吐率、迁移率、并发度和可用性等，及时发现软件故障。云服务层则需要关注应用的专有状态、是否能保持服务层协议以及用户的体验，了解服务的供给运维健康状态等。值得注意的是，自动化云服务的供给失败很有可能是管理不善的配置更改导致的。最后一次配置被修改的知识将成为失效根本原因分析的关键信息。因此，云日志采集系统应该捕捉各个层次配置信息的更新。例如，硬件基础设施层需要记录开机信息，因为这常伴有配置更新信息。

[1] Linux/Unix 操作系统将事件划分为 7 个级别。

捕获完备的日志事件将为做出准确的云决策提供充足的事实依据。然而，各层次众多组件频繁的日志更新和海量的日志管理也将带来不可忽略的运行期开销。为方便以上问题的讨论，我们首先定义日志事件的采集粒度为采集对象的精细程度。其中，采集粒度用字母 σ 表示。如果采集对象的精细程度可划分为 N 个层次，则 $1 \leqslant \sigma \leqslant N$。在 N 值给定的情况下，精细程度越高，粒度值 σ 越大。表 1 给出一个 $N=4$ 时粒度 σ 值的含义样例。在实际应用中，日志事件采集对象精细程度可由运维管理员根据具体计算环境设定。日志采集平台会根据设定的粒度采集特定对象的资源占用和性能状态，如 CPU 使用率、内存使用状态等。从表 1 可以看出，σ 越大，日志采集的粒度越精细。一个虚拟机可以承载多个服务，一个服务可以启动 1 个或成千上万个进程，而一个进程又可以启动多个线程。因此粒度越大，采集工作占用的 CPU 越高，取得的日志数据量越大，占用的内存和磁盘空间也就越大，后面数据的分析也越复杂。

表 1　采集粒度值与采集对象关系样例

Table 1.　An example relationship between the value of the capturing grain and the target objects

σ	采集对象
1	某个虚拟机/集群
2	具体部署在某个虚拟机/群上的云服务
3	提供某云服务的一个进程
4	提供某云服务的一个线程

更具体一些，我们结合一个应用场景进行讨论。根据日志事件对系统性能和正确性影响的严重程度，日志事件一般可以分为多个级别。为简单起见，我们初步划分为三个级别：信息、警告和错误[1]，具体描述见表 2。将虚拟机所有的 CPU 资源看作一个整体，我们可以定义 CPU 使用率

CPU%≥50% 为警告，CPU%＜50% 为信息。因为当 CPU%＜50%，CPU 资源不是紧缺资源，我们不必频繁地探测每个进程的 CPU 使用率，此时 $\sigma=1$ 即可。但当 CPU%≥50%，这说明该虚拟机 CPU 紧缺：它可能承载了计算密集型服务，也可能是某服务突然启动太多的进程满足用户激增的需求，也可能是有恶意或有 bug 的软件进程在运行等。此时，系统管理员需要关注系统相关层次和组件的状态改变和操作事件，包括过去的和正在进行的，确定具体的原因并采用相应的措施。不同的原因，处理方法也不一样。如果是计算密集或用户激增，系统管理员可经协商后增加该虚拟机或服务的 CPU 资源。如果是恶意应用或有 bug 程序，可能要停止该服务的运行。为确定原因，需要加大日志事件采集粒度，采集每个服务的状态及其相应运行进程的数目、每个进程的状态等，即此时 $\sigma=3$ 更好。图 2 描绘了几种常见的警告事件及相应细粒度的日志采集信息。

由上面的描述与分析可见，云日志采集器要尽量避免过度采集日志。构建一个动态粒度自配置采集行为的日志采集平台对构建健康的云计算环境是非常必要的。

3 粒度自配置日志采集平台

3.1 体系结构

图 3 为面向云计算的粒度自配置平台的体系结构，该平台主要包括两大模块：日志采集粒度决策器和日志采集器。其中日志采集粒度决策器包括存储日志采集粒度规则和事实的知识库、基于规则作出采集粒度选择和动态增加或关闭相关日志数据采集模块决策的推理机以及为方便系统管理员更新知识库规则的管理界面。其中，决策

表 2 日志事件级别

Table 2. The classification of the log events

级别	采集对象
信息	这类事件旨在记录基本操作信息，如某服务启动、网卡的打开或关闭等
警告	这类事件旨在捕获云服务提供商、系统管理员或用户的注意，但此类事件一般认为不会对系统的运作带来严重的影响
错误	这类事件表明系统状态的不正确改变，且该改变会导致系统实效，如非法内存访问

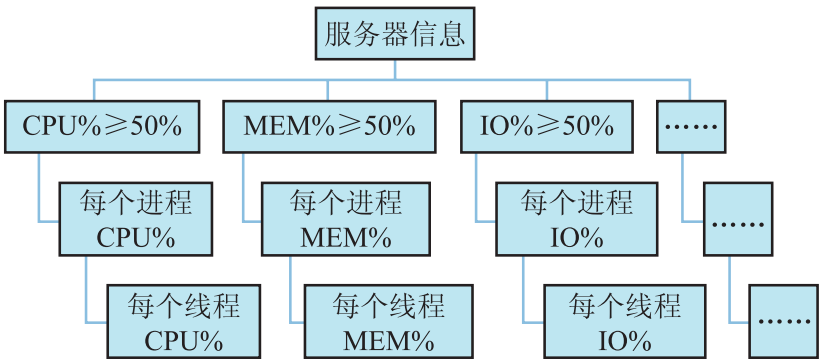

图 2 多粒度日志事件样例

Fig. 2. The multi-grain-level log event samples

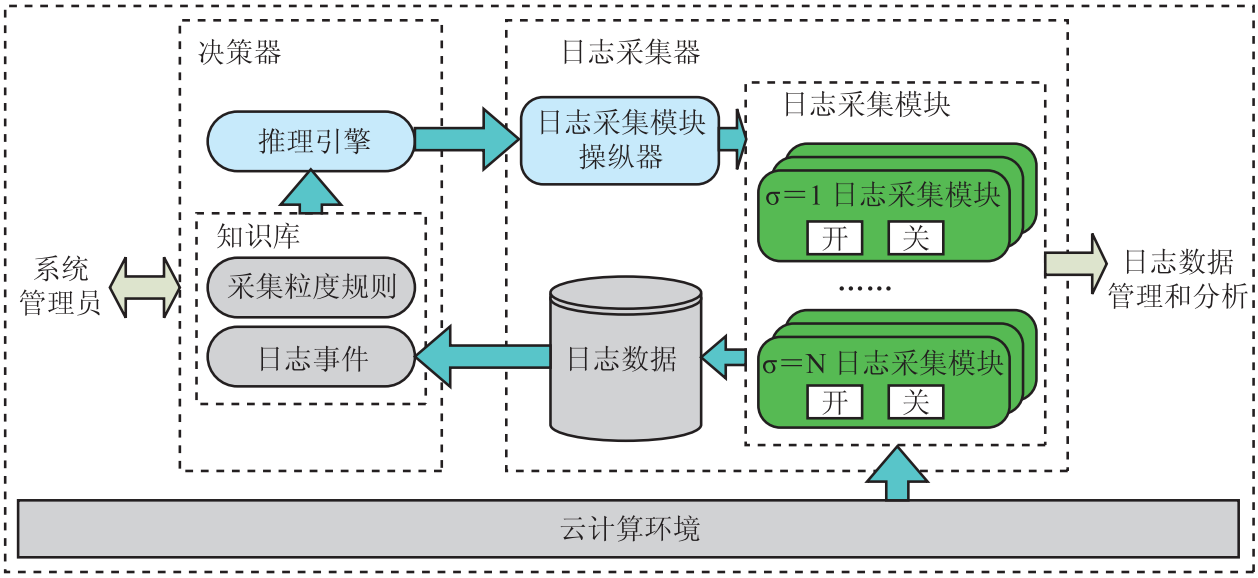

图 3　粒度自配置日志采集平台体系结构

Fig. 3.　The system architecture of the grain-level self-configuring log capturing platform

有三种模式：自动、推荐和不确定。当决策为自动模式时，意味着所发生的事件触发的规则在预定义的规则库中，系统将采用系统管理员根据经验总结出来的有效常用处理方式；当决策为推荐模式时，意味着该事件可触发多个规则，也就是有多项选择，相应的选择项将发给系统管理员作确认，然后系统将收到的确认选项发给日志采集器去执行；如果决策模式是不确定模式，系统管理员将挂起系统，深入调研历史日志信息后添加或修改相应的处理规则使平台得以继续运行。

相应的日志采集器包含三大主要构件：日志采集模块、日志采集模块操纵器和日志数据库。因为日志采集模块需要在运行期动态打开和关闭，所以我们的日志采集器包含众多便于添加和关闭的采集模块，每个模块采集一定的系统状态。而与现有的日志采集模块不同的是，每个模块都将添加一个属性：采集粒度值。日志采集操纵器是一个长期运行的服务程序，它根据决策器推理引擎推送来的决策决定启动哪个粒度的日志采集模块以及哪些模块。然后相应地打开的日志采集模块就会采集数据并存放到日志数据库中供日志数据管理和分析使用。

3.2　动态采集粒度选择规则

平台刚开始启动时，日志采集模块控制器将采用缺省的采集粒度，通常是 $\sigma=1$ 或系统管理员预先设定的采集粒度，对环境进行日志采集。决策器按照一定的周期扫描日志数据，然后针对知识库中预设的采集粒度规则进行扫描，如果前提满足，相应的规则将会激活。

针对 $\sigma=1$ 层的日志采集，采用大家熟悉的 CPU 使用率作为例子阐述动态采集粒度选择的规则。首先，云计算环境里一个很重要的概念是资源的池化。其资源分配一般是遵循以下流程：用户提出其资源的需求，一开始，系统分配给用户的资源一般低于其请求。当运行期发现用户对某资源的需求较大，目前的分配不能有效地提供服务时，系统将为其动态扩容，直至满足以下条件之一：满足该用户的请求、达到其请求的最大值、系统相应资源池备用资源低于预设的阈值。特别地，用大家熟知的 CPU 资源为例，为方便规则的表达，我们可以用 CPU_P 表示可用 CPU

资源总量，CPU_T 表示 CPU 资源阈值。假定当前采集粒度为 1，针对 CPU 使用率和当前 CPU 资源池的可用 CPU 数量，规则 1、2 和 3 给出三个粒度选择规则。当 CPU 利用率大于 50%，且可用 CPU 数量大于 CPU 资源池阈值时，表明系统有充足的 CPU 资源，可自动给该用户扩容。但因为警告事件出现，系统潜在可能的问题，采集粒度精细化为 2，相应规则见规则 1。此时，系统将进一步调研该虚拟机的进程数量、每个进程使用 CPU 的状况等以确定是否为突发事件发生还是恶意进程的存在等。扩容后，发现整个虚拟机的 CPU 使用率持续低于 50%，如经过连续 10 次日志采集所得的 CPU 使用率均低于 50%，也就是系统恢复正常，那么采集粒度粗化回 1，从而降低日志采集的运行期开销，相应规则见规则 2。当 CPU 利用率大于 50%，且可用 CPU 数量低于 CPU 资源阈值，表明系统此时可用 CPU 资源受限，系统向管理员人工推送以下三个建议选项同时等待管理员的决策：(1)暂停服务；(2)迁移服务到其他云环境；(3)扩大 CPU 总资源池的可用资源数量。同时日志采集粒度精细化，进一步调研更多日志信息以确定承载服务的状态，具体规则定义见规则 3。如果资源受限，日志采集粒度精细化到 3 并确定某进程的 CPU 利用率持续增加，则自动暂停该进程相应的服务并继续密集采集日志监控系统，相应规则见规则 4。

相应的粒度选择和行为决策将发给日志采集器，日志采集模块操纵器将根据决策打开或关闭相应的模块进行数据采集。通过以上例子可以看出，运行期自配置日志采集系统在服务正常运行期间将日志采集的运行期代价尽量降低，用尽量多的资源更好地满足用户的需求。当系统运行环境有异常时，启动必要的日志采集模块，采集相关信息从而确定问题。当问题解决后及时调整日志采集粒度，降低其运行期开销。

规则 1　自动 CPU 扩容并精细化采集粒度

Rule 1. Automatically add CPU and refine grain level

```
RULE_NAME＝AUTO_CPU_ADD_
GRAIN_INC
IF［σ＝1 CPU%≥50% && CPU_P ＞ CPU_
T］; THEN
    事件级别 ＝警告
    决策模式 ＝自动
    σ＝2
FI
```

规则 2　自动 CPU 扩容并粗化采集粒度

Rule 2. Automatically add CPU and coarsen grain level

```
RULE_NAME＝AUTO_CPU_ADD_
GRAIN_DEC
IF［σ＝2 CPU%<50% && Check_T＝4］;
THEN
    事件级别＝信息
    决策模式＝自动
    σ＝1
FI
```

规则 3　CPU 资源受限下推荐处理方案

Rule 3. Propose recommendations when CPU resources are limited

```
RULE_NAME＝RECOMMEND_CPU_
LIMITED
IF［σ＝1 CPU%≥50% && CPU_P＜ CPU_
T］; THEN
    事件级别＝警告
    决策模式＝推荐
    推送决策选项：
      1.增加CPU资源
      2.迁移服务
      3.暂停服务
    σ＝2
FI
```

规则 4　因 CPU 资源不足自动停止服务

Rule 4.　Automatically stop services since CPU resources are insufficient

```
RULE_NAME＝AUTO_CPU_ERROR
IF［σ＝3 某进程CPU%≥90% && CPU_P
＜CPU_T］; THEN
    事件级别＝错误
    决策模式＝自动
    暂停该进程相应的服务
    σ＝3
FI
```

4　平台实现和性能评估

作为概念验证，我们搭建了该平台的简单原型系统。同时，我们针对该平台进行一个初步的案例学习来评估其系统性能。

4.1　原型系统的实现

考虑到目前许多云服务提供商都将 Linux 作为他们的首选操作系统，故我们的平台采用 Linux 操作系统构建。其次，我们的日志采集器改写自淘宝开源数据采集工具 Tsar[11]。日志采集模块和日志数据库实现目前基本使用 Tsar 原来的实现，我们所作的修改主要是为每个日志采集模块添加采集粒度属性，便于实现采集粒度选择。Tsar 日志采集模块收集服务器系统和应用信息。其中，服务器信息包括 CPU、内存、TCP 和 IO 等使用信息， 应用信息包括响应时间等。最后，收集到的数据存储在服务器本地磁盘上，也可以发送到远程数据库保存。Tsar 提供界面供用户随时查询历史信息，也可以将信息发送到 Nagios 报警[14]。

我们采用 Tsar 的另一个重要的原因是 Tsar 增加模块比较方便，只需要新增模块的数据采集函数和展现函数按照 Tsar 的要求编写，就可以加入到 Tsar 相应的配置文件中。Tsar 数据采集模块的开关状态由配置文件指定，每次配置文件修改后，Tsar 会自动读取配置文件信息并启动或关闭相应的服务。在原来的 Tsar 系统中，配置文件更新由系统管理员操作。为实现运行期自动开关指定的日志采集模块，我们的日志采集模块操纵器实现为 Linux 操作系统上的一个守护进程，监听来自决策器的日志采集粒度和采集模块状态的更新请求。当收到相应的请求，日志采集模块控制器将自动修改相应的配置文件，然后 Tsar 会读取相应的变化并更新采集工作，不需要人工的参与。

决策器的构建基于开源规则引擎 CLIPS（C Language Integrated Production System）[15]。还有其他的开源规则引擎如 Drools[16]等，考虑到日志采集的次数频繁与数据大量，我们采用 C 语言编写开源软件以提高其运行效率。同样基于运行效率的考虑，所有上述模块都采用 C 语言实现。

4.2　性能评估

由于我们的平台改自一体式日志采集平台 Tsar，故将粒度自配置日志采集平台与 Tsar 的日志采集运行时间进行比较来初步验证我们平台的有效性。目前的粒度自配置日志采集平台的实现尚处于概念验证阶段，我们仅启动部分采集模块进行采集运行期开销比较，包括占用系统整体 CPU、内存、I/O 和负载等资源使用情况以及每个进程的 CPU 利用率，相应的模块名称、描述及其采集粒度值见表 3。

实验环境使用 1 台 DELL OPTIPLIEX760 计算机，其 CPU 设置为 Inter（R） Core（TM） Quad CPU Q8400 @ 2.66 GHz 2.67 GHz，内存为 4 G，硬盘为 500 G，操作系统采用 Ubuntu 12.04 LTS。云服务供给环境通常为多进程环境，为了评估不同进程数量环境下粒度自配置日志采集平台与 Tsar 的运行期开销，我们用多进程并发环境模拟云服务负荷环境，分别为 1000 个并发进程、5000 个并发进程和 9000 个进程的日志采集环境进行比较

表 3 性能评估实验中用到的日志采集模块

Table 3. The log data capturing modules used in the experimental performance study

模块	描述	σ
Mod_cpu	采集系统 CPU 利用率	1
Mod_mem	采集系统内存利用率	1
Mod_io	采集硬盘 I/O 读写数据	1
Mod_load	采集运行队列长度和平均负载均衡	1
Mod_Process	采集系统每个进程的 CPU 数据利用率	2

实验。同时，在运行过程中，为触发粒度自配置日志采集平台的粒度改变，我们模仿用户向服务器请求计算密集型云服务事件，启动一个计算密集型计算服务，实验采用计算斐波纳契数服务。该服务开始之后系统的 CPU 利用率会逐渐上升，当自配置日志采集平台采集的 CPU 利用率大于规则库里设定的阈值时，它会按照预先设定的规则库中的规则启动粒度为 2 的模块来进行细粒度监控，采集每个进程的 CPU 利用率并作记录。当该服务结束后，系统 CPU 利用率降低，自配置日志采集平台会按照预先设定的规则库中的规则关闭粒度为 2 的模块，重新恢复粒度为 1 的监控。

为公平起见，两个系统采用相同的日志采集频率：每隔一分钟收集一次资源利用率信息，并保存到相应的日志文件中。粒度自配置日志采集平台一开始的采集粒度设置为 1。实验结果发现，粒度自配置日志采集平台花在日志采集上面的时间为 700 多微秒，而 Tsar 的约为 1500 微秒。由此可以看到，粒度自配置日志采集平台的日志采集运行时间开销比较小。目前我们启动的采集模块比较少，真正的云计算环境采集模块会非常多，可以预见该系统可以节省更多的日志采集运行期开销。同时我们也注意到，由于规则库的引入，目前粒度自配置日志采集平台的虚拟内存消耗比 Tsar 大，在上述模块下，Tsar 约占 1000 KB，而粒度自配置日志采集平台约占 1844 KB。但当更多模块导入系统时，Tsar 的整体内存占用会超过粒度自配置日志采集平台。

5 总结与展望

复杂多样化的云计算环境需要日志采集系统提供必要的日志数据进行失效事件溯源、审计和取证，从而确保云服务持续可靠和有效的供給。

云计算需要充足的日志数据，然而日志采集能力越强的日志管理系统的运行期开销越高，这将对云服务系统整体性能带来损害。针对这一矛盾，本文提出日志采集粒度的概念，然后设计并编程实现一个面向云计算的粒度自配置日志采集平台，实现根据系统的运行情况和预先定义的规则动态增加或关闭相关日志数据采集模块，实现采集必要日志数据的同时避免过度采集。也即尽可能降低服务正常运行期日志采集的开销。在运行环境存在异常情况时，能尽可能详细地采集相关日志信息。

我们初步的案例学习结果表明了平台的有效性。但目前系统的设计、实现和性能评估都偏于简单的情景。如我们的规则是由管理员自定义的，一个自增强学习型规则库将进一步简化系统管理的工作。我们将进一步调研该平台的支撑算法和机制。同时，我们的初步试验结果也表明规则库本身比较占用系统内存，但当规则越来越多时，如果根据需要导入特定的规则集也将是我们下一步的研究目标。

致 谢

感谢深圳大学计算机与软件学院 2009 级计

算机科学与技术专业学生汪敏，2009 级软件工程专业学生肖东、赵慧敏，2010 级软件工程专业学生张志强、陈国焕，以及 2011 级软件工程专业硕士周鸿同学在这个项目的前期调研和实验工作。

参 考 文 献

[1] Miller R. Failure rates in Google data centers [OL]. https://www.datacenterknowledge.com/archives/2008/05/30/failure-rates-in-google-data-centers/.

[2] Kavulya S, Tan J, Gandhi R, et al. An analysis of traces from a production MapReduce cluster [C] // Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010: 94-103.

[3] Daniel E, Lal R, Choi G. Warnings and errors: a measurement study of a UNIX server [C] // IEEE the 29th International Symposium on Fault-Tolerant Computing, 1999.

[4] Sahoo PK, Oliner AJ, Rish I, et al. Critical event prediction for proactive management in large-scale computer clusters [C] // Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003: 426-435.

[5] Zhou W, Zhan JF, Meng D, et al. Online event correlations analysis in system logs of large-scale cluster systems [C] // Network and Parallel Computing, Lecture Notes in Computer Science, 2010: 262-276.

[6] Rabkin A, Katz RH. Chukwa: a system for reliable large-scale log collection [C] // Proceedings of the 24th International Conference on Large Installation System Administration, 2010: 1-15.

[7] Facebook scribe [EB/OL]. https://github.com/facebook/scribe.

[8] Kreps J, Narkhede N, Rao J. Kafka: a distributed messaging system for log processing [C] // The 6th International Workshop on Networking Meets Databases, 2011.

[9] Cloudera. Flume [EB/OL]. https://cwiki.apache.org/FLUME/.

[10] Caibinbupt. 分布式海量日志采集、聚合和传输系统: cloudera flume [EB/OL]. http://www.searchdatabase.com.cn/showcontent_45952.htm.

[11] Tsar: 淘宝服务器系统信息采集工具 [EB/OL]. http://tsar.taobao.org/.

[12] Mell P, Grance T. Effectively and Securely Using the Cloud Computing Paradigm [Z]. Technical Report, 2009.

[13] 陈军. 腾讯云平台与技术实践分享 [Z]. 第三届云计算大会分论坛二“云计算平台与应用实践”, 2011.

[14] Nagios. Nagios server and nagios plugins [EB/OL]. http://prdownloads.sourceforge.net/sourceforge/nagios/.

[15] CLIPS: http://clipsrules.sourceforge.net/index.html.

[16] Drools project [OL]. http://www.jboss.org/drools/.