一种云环境下基于分级管理的自律计算模型

刘文洁 李战怀
(西北工业大学计算机学院 西安 710072)

摘 要 自律计算是实现复杂IT系统的自我修复、自我管理的有效手段。但是在云环境下,系统所管理的计算资源数量急剧增长,传统的资源调度算法和模型架构无法对资源进行合理分配,导致资源的利用率很低。为了提高云环境下的资源管理效率,提出了一种面向云环境的自律计算模型,它采用多自主管理器的分级管理模式来解决传统自律计算模型在处理大量请求时的瓶颈问题;并提出了混合策略管理模式来应对不同的故障修复请求。该模型能够降低云环境下的资源管理成本,满足云服务所需的SLA。

关键词 云环境,自律计算,分级管理,故障修复
中图法分类号 TP311 **文献标识码** A

Autonomic Computing Model Based on Hierarchical Management in Cloud Environment

LIU Wen-jie LI Zhan-huai
(School of Computer, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract Autonomic Computing is an effective method to make complex IT system to realize self-recovery and self-management. But in cloud environment, with the rapid growth of the computing resources, traditional resources algorithms and models can not allocate the resources effectively, therefore the resources usage rate is very low. To enhance the resource management efficiency, this paper proposed an autonomic computing model for cloud environment, which divides the autonomic manager into master and slave, uses hierarchical management to solve the bottleneck problem when dealing with massive requests, and uses the mixed policies method for different fault recovery requests. This model can lower the resources management cost in cloud environment and meet the SLA that cloud services need.

Keywords Cloud environment, Autonomic computing, Hierarchical management, Fault recovery

1 引言

云计算使得网络环境下的可用资源可以整合在一个平台下运转,并以提供高可用不间断的服务为目标。但是,云环境下的计算资源数量巨大,管理极为复杂,云端需要部署不同的软件来满足不同用户的服务请求,而且,云端的软件存在系统与版本的差异,对于软件的部署、维护和更新必须协调进行,这些导致的云计算的系统管理复杂性是前所未有的。自律计算是IBM提出的解决系统管理复杂性问题的一个关键技术,通过实现系统的自我管理来使复杂的IT系统具有"自我监控、自我配置、自我优化和自我恢复"的能力,从而降低人力管理成本[1]。

但是,现有的自律计算领域所提出的基础模型所管理的资源数量有限,针对云环境下的资源数量的急剧扩展,自主管理器将遇到无法及时应答用户请求、响应延迟等问题。此外,基础模型所管理的资源以异构的物理资源为主,针对云环境下主要采用虚拟化技术的资源特点,还需要对资源类型进行分类扩充,并针对资源的高可扩展性进行有效处理。再者,基础模型没有考虑在云环境下需要保证服务的SLA和高可用问题,这些都需要进一步改进。

本文在对云环境下的资源特点以及自律计算模型研究的基础上,提出了一个分级管理的自律计算模型。将传统模型中的单自主管理器划分为主从两级管理模式,以缓解云环境下自主管理器处理大量请求的延迟问题,此外,在模型中添加了对服务SLA的匹配和监控处理,满足了云环境下系统的管理需求。该模型能够减少云计算环境下的系统管理成本,提高故障发现和修复率,实现云服务的高可用性。

2 相关研究

用自律计算来解决系统管理的复杂性问题由来已久,但是早期的自律系统大多数考虑的是分布式异构环境下的物理资源管理,并没有考虑云环境下的各种制约因素,如大规模的虚拟化应用、服务等级协议等。虽然研究人员也提供了改进的模型和管理框架[2,3],但其并不能直接用于云环境资源管理。

针对现有的自律模型并不能直接应用于Web服务的问

到稿日期:2013-05-23 返修日期:2013-07-19 本文受国家高技术发展计划(863计划:2012AA011004),国家自然科学基金(61303037,61033007),西北工业大学基础研究基金(JC201261)资助。

刘文洁(1976—),女,博士,讲师,主要研究方向为软件理论、自律计算、高可用性系统,E-mail:liuwenjie@nwpu.edu.cn;李战怀(1961—),男,教授,博士生导师,主要研究方向为数据库与知识库系统、存储区域网络、软件工程。

题，很多学者提出了改进的自律计算基础模型，使其适应复杂的网络资源管理。芝加哥大学的 YU Cheng 提出了一种面向服务的自律管理框架[4]。该架构中，具有点对点的 QoS 请求的业务和管理应用可以由分布式环境中的标准组件来构成，并且能够根据服务等级协议来管理。该体系的目的是解决网络管理的复杂度问题，具有一定的可行性，但该框架并没有考虑到云环境下的资源特点、高扩展性等要求，因此还无法在云环境下应用。文献[5]提出了一种自律资源管理框架来实现虚拟服务中心的自我管理行为。通过采用虚拟服务器，细化资源共享方式来改善服务中心资源分配方式的灵活性，提高工作效率。文章提出了采用可能性分析模型的非线性优化技术来实现框架，以实验的方式证明了框架对于自律资源管理在性能上的提高，但该框架同样尚未应用于实际的自律资源管理中。

针对自律计算基础模型在管理云资源上的缺陷，维也纳科技大学的 Michael Maurer 指出，在云环境下需要重新设计自律系统的 MAPE 循环评测方法来适应新的需求[6]。原有的自律计算框架不能直接在云环境下应用，例如在虚拟层，监视工具通常需要按需配置，以区分所监视的应用和资源。因此，传统的 MAPE 循环也需要重新设计以适应云环境。

云计算中的资源管理除了要考虑资源分配，还要考虑满足服务等级协议 SLA。有研究人员指出，在云环境下构建自律计算系统的一个挑战就是我们如何评估一个自律计算系统在不断增长的性能或能源下的执行能力，其能力能够满足指定的 SLA[7]。美国韦恩州立大学的 Cheng-Zhong Xu 提出了一种面向自律云管理的加强学习方法[8]，该方法应用在实时自动配置的云中，能够根据云的动态性和变化的工作负载自适应虚拟机资源预算和装置的参数设定，从而确保服务质量，在系统级应用目标和指定装置的 SLA 优化目标间进行很好的平衡。但是该方法只适用于相同物理机上的虚拟机配置和管理，对于在云环境下跨物理机集群的虚拟机配置，并不能保证系统满足指定的 SLA。

文献[9]提出了一种检测 SLA 违例的基础设施(DeSVi)体系结构，其通过有效的资源监控检测到 SLA 违例。但由于并未考虑到云资源管理的需求，而且所监控的节点数量有限，其并不能用于实际的云资源管理。Champrasert 提出了在自律的云应用中如何建立具有自优化和自稳定属性的方法[10]，并设计了基于生物学原则如分散、演化、互利的系统：SymbioticSphere，在系统中，每个应用都包含了应用服务和中间件平台。每个服务和平台都作为生物实体来设计，实现了生物学行为，如能量交换、迁移、再生和死亡。每个服务和平台拥有行为策略，如基因，每种策略管理何时怎样激活一个特殊行为。该方法对于建立具有自管理系统的自律云系统具有指导意义。针对云环境下的资源管理，文献[11]中提出了基于拓扑意识的热点移除方法，设计了一个分级近似技术用于节点选择，以有效划分和控制计算，从而减少再配置成本。该方法的目的是减少云系统的能耗，达到系统内部的负载均衡，并未涉及到系统自身的自动化管理，但其分级管理的思想对于云环境下自律系统的设计具有参考价值。

因此，现有的资源管理方法和自律计算模型在资源管理数量、自律特征的实现等方面还有所欠缺，无法适应云环境下系统的复杂性要求，必须产生新的体系结构来管理资源，即需要设计自适应的解决方案来应对不可预测的工作负载。

3 基于分级管理的自律计算模型

3.1 模型体系结构

为了解决云资源管理问题，我们设计了一个分级管理的自律计算模型，将基础模型中的单自主管理器划分为主从两级自主管理器，并在虚拟机上建立多 Agent 协作体系，保证系统内部间的交互协作和虚拟机在系统中的快速扩展。模型体系结构如图 1 所示。

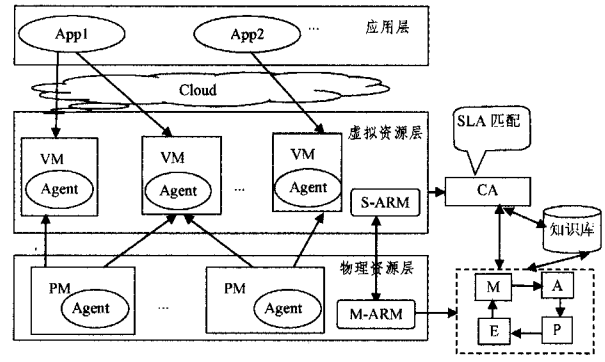

图 1 基于分级管理的自律模型体系结构

模型共分为 3 层：应用层、虚拟资源层和物理资源层。应用层是面向用户提供云上的各种应用或计算资源，根据用户指定的 SLA 来提供相应的服务和资源，SLA 的需求将会传递给虚拟资源层。虚拟资源层是提供所有的可用资源，包括应用与服务，该层上的资源会根据用户的请求数以及物理资源数进行资源的添加和删除。为了实现资源的自动化管理，每个虚拟资源在配置完成后将安装 Agent 来监控当前的资源状态和服务实施过程，并定期向本层的 S-ARM 汇报。S-ARM 表示从属的自律资源管理器(Slave-Autonomic Resource Manager)，其目的是完成虚拟资源的自适应过程，即 CA 阶段(Cloud Adaption)，根据 SLA 请求从知识库中挑选合适的模板作为服务层的服务目标，并根据请求自动选取可用资源，安装相应的应用并提供给用户。物理资源层是实际的物理机器管理层，其管理方法和基础自律模型相同，所有的物理机器安装监视 Agent，每个 Agent 定期地向主自律管理器 M-ARM 汇报自身健康状况并实施对应的动作。M-ARM 表示 Master-Autonomic Resource Manager，完成整个系统的自律管理过程，包括系统监控(Monitor)、分析(Analyze)、计划(Plan)和执行(Execution)过程，即 M-A-P-E 循环。

3.2 自适应配置

根据上述体系结构，假设在云上的某个数据中心可以为客户提供各种应用和服务，客户通过服务等级协议 SLA(说明期望的性能、安全性和安全等级等)征订服务，数据中心的自主管理器根据 SLA 的等级自动从知识库挑选合适的模板作为服务目标，并根据可用资源的数量和 IT 管理策略，自动地为客户选择和配置相关的资源，如服务器、数据库、存储器和网路资源，并在资源上安装和配置各种应用模块，以满足 SLA 规定的服务需求，从而实现系统的自配置过程。在服务开始工作之后，整个服务过程将受到监视服务的监控(由 VM 上的 Agent 和 PM 上的 Agent 来完成)，自主管理器将根据各个 Agent 传来的监控信息觉察服务运行过程中发现的错误，当异常发生时，根据 Agent 所报告的资源编号来定位具体的

错误位置，如出错的服务器、路由器、存储器、虚拟机等，根据问题的特征从知识库选取必要的策略来修复故障，如重启失败的组件或升级一个应用、启动配用服务器等，从而实现系统的自修复过程。此外，由于监控组件不断地检查系统的健康状况并采取必要的措施来防止危险的入侵，因此实现了系统的自保护过程。在自主管理器上的配置优化器能够根据系统掌控的各种参数来优化系统性能，当目标改变、服务添加或删除时，它会采用各种策略来调整自身状态，达到系统的自优化过程。对于上层管理者而言，只需定义目标和策略，如对 A 类用户的反应时间小于 100 毫秒、当系统负载超过 80%时启用备用服务器、A 组件异常时用 B 组件替换等。

3.3 知识库策略

知识库的策略包含 3 种：动作策略、目标策略和效用函数。动作策略是预先规划好的策略，定义了在某些条件到达时采取什么样的动作，如 CPU 超过 80%时启动备用服务器。目标策略只规定目标不指定动作，具体动作由系统根据当前资源状况自动决定，如系统可用存储容量不低于 10%，当低于 10%的时候可以添加存储服务器、添加磁盘阵列或清理磁盘等，使得系统保持在指定的目标范围内，但具体动作不限定。这种策略更具有灵活性，但要求知识库中必须包含可选的备用动作。效用函数策略是最灵活的一种策略，也是最体现系统自主能力的一种策略，它不指定具体目标也不指定具体动作，只指定一个目标函数，函数用于表达每个可能的状态的标量值。系统根据效用函数计算最大的效用值来确定具体条件下的期望状态。这种策略使得系统不受限于具体动作，完全根据系统状态对系统进行调优配置，使得无论何时系统都可以获得当前状态的最优目标。

4 自律管理能力分析

4.1 自适应的 MAPE 循环

相对于传统的自律系统，基于云的自律系统的 MAPE 循环中需要添加对云的自适应过程，即 CA-MAPE 循环。其中，CA 阶段的主要功能是对于请求的 SLA 等级的分析、选择和匹配，并在整个服务运行过程中对服务是否满足用户要求的 SLA 的监控和对于违例状况的处理。CA 过程中的 SLA 匹配流程如图 2 所示。

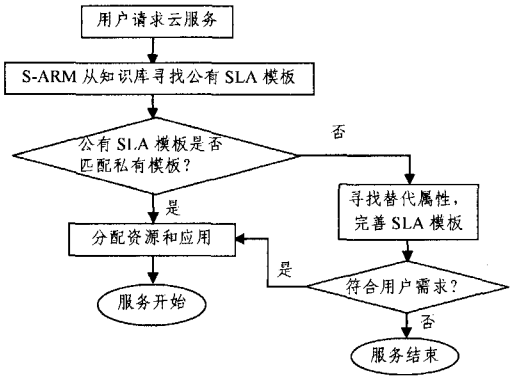

图 2 CA 过程的 SLA 匹配流程

自适应过程在系统建立时完成，包含在配置成功和启动应用前必需的所有步骤，其中包括 SLA 协议的建立和监控系统对于特殊应用的定制。在这个阶段，S-ARM 判断系统的公有 SLA 模板和客户的私有 SLA 模板是否匹配，如果匹配，则按照 SLA 要求和用户需求分配相应的资源和应用，即不进行 CA 过程。如果不匹配，例如，在私有模板中要求的某些属性在公有模板中缺失（如 SLO 时钟速度），此时则进行 CA 过程，根据知识库预先定义的规则寻找替代属性，找到则加入公有模板，否则与用户协商，根据协商结果终止本次服务或采用替代属性继续提供服务。公有模板中预先定义了常用的 SLA 参数以及可以相互替代的属性列表，其中包含的一部分的 SLA 参数和目标值如表 1 所列。

表 1 部分 SLA 参数目标

SLA 参数	值
Incoming Bandwidth(IB)	>10Mbit/s
Outgoing Bandwidth(OB)	>10Mbit/s
Storage(St)	>1024GB
Availability(Av)	≥99%

我们假设典型的云应用中，云用户以 PaaS 的形式来配置应用。服务提供商将资源注册到了包含公有 SLA 模板的知识库中，随后，用户寻找能够配置应用的云服务，云用户使用自己的私有 SLA 模板进行商务应用，假设私有模板不变，如找到匹配的公有模板，协议就达成，应用开始配置并执行。一旦应用配置，执行将会自动完成，减少用户与系统的交互，优化能源消耗，防止 SLA 违例。资源管理器根据监控模块来判断 SLA 是否违例，知识库根据 SLA 违例来决定应用是否迁移，虚拟机和物理机是否要再配置、迁移或开关机。

4.2 基于混合策略的决策

云环境下，资源的添加和删除非常频繁，系统的状态变化也不断发生，传统的基于动作的策略由于必须预先设定条件与动作，当且仅当该条件发生时才能激活所规定的动作。云环境下，对于常规故障可以采用动作策略来完成，如 A 组件异常用 B 组件替换等，但是，为了保证系统的高可用性并且在一些未知条件下能够采取适当的动作来处理故障、维护系统的稳定性，必须采用基于效用函数的策略来管理系统。我们在系统中会采用 3 种策略混合的方式，普通故障采用动作策略和目标策略，未知故障采用效用函数处理，以适应云环境的多样化资源状态。动作策略用于实现系统的自修复，目标策略用于实现系统的自配置，效用函数策略用于实现系统的自优化。效用函数定义如下：

M-ARM 通过计算系统的最大效用值来确定最优的反应方案 δ，设 $C(s_i, s_c)$ 是当前状态下的约束条件（如可用资源集），$U_i(x_i)$ 是第 i 个被管资源的局部效用函数，则对于一个管理 n 个被管理资源的 ARM，基于效用函数的决策表示为：

$$\delta = \arg\ \mathrm{Max} \sum_{i=1}^{n} U_i(x_i)，\text{满足约束条件 } C(s_i, s_c)$$

满足约束条件效用函数把实体的每个可能的状态（系统性能）都映射为一个实数值，用于指示与系统性能（反应时间、延迟、吞吐量）对应的价值，ARM 根据系统的性能模型通过效用的最大化来得到期望的系统状态和可调参数的取值情况，最后调整系统参数，使得系统达到期望的状态。

4.3 主从 ARM 的动态协作

传统的自律系统中只有一个自律管理器 AM，完成整个系统的监控、分析、计划和执行过程。在云环境下，资源以虚拟化的形式对外提供，虚拟资源的变更频繁，必须添加对于虚拟资源管理的自律管理器才能适应资源的变动。因此，我们将原有的一层 AM 变为主自律管理器 M-ARM 和从自律管

理器 S-ARM,M-ARM 完成传统的自律循环过程,负责系统整体的状态收集、监控、策略选择和实施,并接受 S-ARM 的请求,分析故障并将解决方案传递给 S-ARM,S-ARM 将方案进一步分配给出现故障的虚拟资源,通过虚拟资源上的 Agent 来实施策略,修复故障。此外,CA 过程主要由 S-ARM 来完成,根据知识库的模板来完成 SLA 的匹配。对于 SLA 的违例则由 VM 上的 Agent 来监控,并定期地向 S-ARM 汇报,S-ARM 与 M-ARM 协商后对违例进行记录与处理。协作过程如图 3 所示。

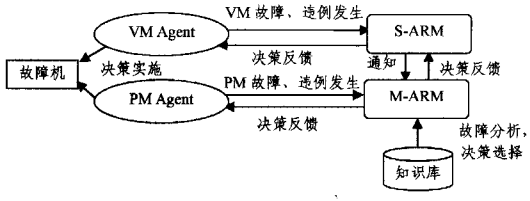

图 3 S-ARM 和 M-ARM 的协作过程

结束语 为了提高云环境下的资源管理效率,解决传统的自律计算模型不能直接应用云环境的问题,本文提出了一个基于分级管理的自律计算模型。针对云环境下的资源虚拟化特征以及需要保证服务的 SLA 等特点,在传统的自律模型中设计了 SLA 匹配的 CA 过程,建立了分级管理的多 Agent 协作体系,使得云环境下的资源管理可以纳入自律循环体系中,从而实现资源管理的高效率。该模型已经应用于某国际合作项目的云系统设计中,实践表明该模型能有效改善云系统的资源管理能力,提高故障修复率并保证云服务的 SLA。

参 考 文 献

[1] Kephart J, Chess D. The Vision of Autonomic Computing [J]. IEEE Computer Society, 2003, 36(1): 41-50

[2] Shi C Y, Zhang W. Agent-Based Computing [M]. Beijing: Tsinghua University Press, 2007: 275-320

[3] Bogdan S, Dan I, Marin L, et al. Towards a real-time reference architecture for autonomic systems[C]//Proceedings-ICSE 2007 Workshops: International Workshop on Software Engineering for Adaptive and Self-Managing Systems. 2007: 1-10

[4] Yu Cheng, Alberto L-G, Source F I. Toward An Autonomic Service Management Framework: A Holistic Vision of SOA, AON, and Autonomic Computing [J]. IEEE Communications Magazine, 2008, 46(5): 138-146

[5] Wang Xiao-ying, Lan Dong-jun, Fang Xing, et al. A resource management framework for multi-tier service delivery in autonomic virtualized environments [C]// Network Operations and Management Symposium, 2008, NOMS 2008. IEEE, 2008: 310-316

[6] Michael M, Ivan B, Vincent C E, et al. Revealing the MAPE loop for the autonomic management of cloud infrastructures [C]// Proceedings-IEEE Symposium on Computers and Communications, ISCC'11. 2011: 147-152

[7] Huebscher, Markus C. A survey of Autonomic Computing-Degrees, models, and applications [J]. ACM Computing Surveys, 2008, 40(3): 7-28

[8] Xu Cheng-zhong. URL: A unified reinforcement learning approach for autonomic cloud management [J]. Journal of Parallel and Distributed Computing, 2012, 72(2): 95-105

[9] Emeakaroha, Vincent C. Towards autonomic detection of SLA violations in Cloud infrastructures [J]. Future Generation Computer Systems, 2012, 28(7): 1017-1029

[10] Champrasert, Paskorn. Exploring self-optimization and self-stabilization properties in bio-inspired autonomic cloud applications [J]. Concurrency Computation Practice and Experience, 2012, 24(9): 1015-1034

[11] You G-W, Hwang S-W, Jain N. Ursa: Scalable load and power management in cloud storage systems [J]. ACM Transactions on Storage, 2013, 9(1): 1-29

(上接第 188 页)

[3] Wang C, Lai J, Suen C, et al. Multi-Exemplar Affinity Propagation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(9): 2223-2237

[4] Sakellariou A, Sanoudou D, Spyrou G. Combining multiple hypothesis testing and affinity propagation clustering leads to accurate, robust and sample size independent classification on gene expression data[J]. BMC bioinformatics, 2012, 13(1): 270

[5] Wang L, Zhang L. Color Image Segmentation Algorithm Based on Affinity Propagation Clustering[J]. Foundations of Intelligent Systems. Springer Berlin Heidelberg, 2012, 122: 731-739

[6] 王开军,李健,张军英,等. 半监督的仿射传播聚类[J]. 计算机工程,2007,33(23):197-201

[7] He Yan-cheng, Chen Qing-cai, Xiao-long, et al. An Adaptive Affinity Propagation Document Clustering[C]//Proceedings of the 7th International Conference on Informatics and Systems. Shenzhen, China, 2010: 1-7

[8] Zhong Y, Zheng M, Wu J, et al. Search the Optimal Preference of Affinity Propagation Algorithm[C]//2012 Fifth International Conference on Intelligent Computation Technology and Automation (ICICTA). IEEE, 2012: 304-307

[9] Shang F, Jiao L C, Shi J, et al. Fast affinity propagation clustering: A multilevel approach [J]. Pattern recognition, 2012, 45(1): 474-486

[10] 张震,汪斌强,伊鹏,等. 一种分层组合的半监督近邻传播聚类算法[J]. 电子与信息学报,2013,35(3)

[11] Frey B J. Affinity propagation FAQ [EB/OL]. http://www.psi.toronto.edu/affinitypropagation/faq.html, 2012-01-05/2012-12-01

[12] Ding Fan, Luo Zhi-gang, Shi Jin-long, et al. Overlapping community detection by kernel-based fuzzy affinity propagation[C]// Proceedings of International Workshop on Intelligent Systems and Applications. Changsha, China, 2010: 1-4

[13] Dudoit S, Fridlyand J. A prediction-based resampling method for estimating the number of clusters in a dataset [J]. Genome Biology, 2002, 3(7): 1-21

[14] Rousseuw P J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis [J]. Journal of Computational and Applied Mathematics, 1987, 20: 53-65

[15] Blake C L, Merz C J. UCI repository of machine learning databases[EB/OL]. http://archive.ics.uci.edu/ml/, 2012-05-01/2012-12-01

[16] Fred A L N, Jain A K. Robust Data Clustering[C]//Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Wisconsin, USA, 2003