开放云端计算环境中的任务执行代码安全机制

徐小龙[1,2] 耿卫建[1] 杨 庚[3] 王汝传[1]
(南京邮电大学计算机学院 南京 210003)[1]
(中国科学院软件研究所信息安全国家重点实验室 北京 100190)[2]
(南京邮电大学计算机技术研究所 南京 210003)[3]

摘 要 云端计算可以充分聚合Internet网络服务器端和边缘终端节点的计算资源来获得更大的效益。但将计算任务部署到用户终端上执行却带来了安全隐患。分属于不同用户的海量终端节点之行为显然不可靠,计算安全性也难以保障。特别是作为任务执行者的用户终端节点可能篡改任务中的程序代码或数据,返回的是虚假的结果,或是窥探有私密性要求的代码和数据。提出一种新的基于内嵌验证码的加密函数的代码保护机制,它可同时满足计算完整性和私密性,能够有效验证返回结果的正确性,并保障计算代码不被窥知。为了进一步提高任务执行的成功率和缩短作业周转时间,将任务代码优先分发给信誉良好且执行成功率高的节点来执行。还提出了一种评估任务执行节点可信性的方法。具体描述了任务执行代码保护机制的实现流程,并对机制的性能进行了详细的分析与验证。

关键词 云计算,信息安全,代码保护,可信评估
中图法分类号 TP309 **文献标识码** A

Code Execution Security Mechanism for Open Cloud & Client Computing

XU Xiao-long[1,2] GENG Wei-jian[1] YANG Geng[3] WANG Ru-chuan[1]
(College of Computer, Nanjing University of Posts & Telecommunications, Nanjing 210003, China)[1]
(State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[2]
(Institute of Computer Technology, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)[3]

Abstract The cloud & client computing can take full aggregation of network server-side and edge node computing resources of Internet to gain greater benefits. However, deploying tasks to terminal nodes would bring the corresponding security risks at the same time. The behaviors of terminal nodes belonging to different users are clearly not reliable, which means the computing security is difficult to guarantee. One of these security risks is that a terminal node working as the task executor may tamper with the program or data of the task, and return the fake result, or pry into the code and data with privacy requirement. This paper presented a new code protection mechanism based on encryption function with verification code meeting integrity and privacy both, which makes it possible to effectively verify the correctness of returned results and to guarantee the code not be spied. In order to improve the success rate of task implementation further and reduce job cycle time, tasks ought to be distributed to those nodes with good reputations and high success rate of task implementation to execute. This paper proposed the credibility evaluation of node, described the work procedure of the code protection mechanism and gave the analysis and verification of the security performance of the system in detail.

Keywords Cloud computing, Information security, Code protection, Credibility evaluation

1 引言

云计算(Cloud Computing)技术[1-5]是一种新型的网络计算技术,它基于充分利用网络化计算与存储资源,更好地整合了互联网和不同设备上的信息和应用,把所有的计算、存储资源连结在一起,实现最大范围的协作与资源分享,达成高效

到稿日期:2011-08-03 返修日期:2011-11-22 本文受国家自然科学基金项目(60873231),国家教育部高等学校博士学科点专项科研基金课题(20093223120001),中国博士后科学基金项目(2011M500095),江苏省科技支撑计划(BE2009158),江苏省自然科学基金(BK2011754,BK2009426),信息安全国家重点实验室开放课题(03-01-1),江苏省高校自然科学研究项目(09KJB520010)以及江苏高校优势学科建设工程项目(yx002001)资助。

徐小龙(1977—),男,博士,副教授,主要研究方向为计算机软件、分布式计算、信息安全、Agent技术等,E-mail:xuxl@njupt.edu.cn;耿卫建(1987—),男,硕士生,主要研究方向为基于网络的计算机软件和信息安全技术等;杨 庚(1961—),男,博士,教授,博士生导师,主要研究方向为计算机应用、网络计算技术、信息安全技术等;王汝传(1943—),男,教授,博士生导师,CCF会员,主要研究方向为计算机软件、网络计算和信息安全技术等。

率、低成本计算目标，按需求解各类复杂的用户问题。云计算达到了两个分布式计算的重要目标：可扩展性和高可用性。可扩展性表达了云计算能够无缝地扩展到大规模的集群之上，甚至包含数千个节点同时处理，使得处理超大规模数据的分布式计算成为现实。目前云计算已被广泛用于包括数据挖掘在内的大规模数据处理工作。

目前的云计算应用系统虽然也倾向于利用廉价计算和存储设备来提供各种服务，但是都只考虑服务器端的计算能力和存储资源[5]，而简单认为网络边缘终端节点仅仅是服务的消费者，对于终端节点所蕴含的各种可利用的潜在资源考虑不够。事实上，终端节点本身也拥有各种计算、存储甚至信息资源，且常常处于闲置状态，接入 Internet 的海量终端节点所拥有的计算和存储资源被浪费了。显然，通过充分考虑和挖掘终端节点所蕴含的各种可利用的潜在资源，将“云计算”模型扩展为“云端计算”(Cloud & Client Computing)模型，可以更大范围地聚集网络上服务器端和用户终端上的所有资源，获得更大的效益。

然而，如果将计算任务部署到用户终端上执行，一个值得关注的问题是如何保证计算的安全性。这里的安全性包含了两个方面的内容：对于任务的发起者而言，如何确保发送到其它用户终端上执行的任务不被该用户篡改，如果有计算私密性需求的计算甚至还有“让该用户执行但又不希望该用户了解该任务是什么”的特殊要求；对于作为任务的执行者的终端用户而言，如何相信发给自己执行的任务程序是安全的，不存在导致自身利益损失的安全隐患。针对分发到用户终端节点上的任务执行代码受到的安全危险，本文提出一种用于开放云端计算环境的任务执行代码保护机制，目的是有效保障任务执行代码的正确性，从而保证利用云端计算系统平台最大限度地聚集 Internet 上包含网络边缘节点的计算能力的可行性。

2 云端计算环境及安全性问题分析

2.1 云端计算环境

在基于 Internet 的分布式计算环境中，可聚合的各种资源(计算、存储、数据等)并不仅仅来自于服务器节点，用户终端节点在获取服务和资源的同时，也完全可以利用自身的计算存储等能力同时提供服务。也就是说，由于终端节点本身也拥有资源，因此终端节点加入云计算环境时，也有可能贡献自身闲置的资源和提供服务，这就将单纯的云计算环境拓展为云端计算[5]环境。

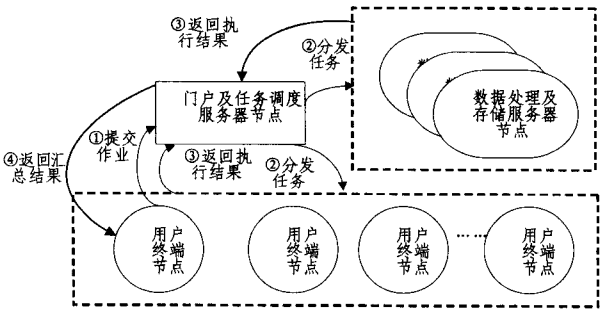

图 1 云端计算环境示意图

从图 1 可以看出，在云端计算环境中，作为服务提供或是任务执行者的节点并不仅仅是服务器端节点。当某一用户向系统提交一项作业时，用户需要自定义好自己需要的内容，经由客户端相关的代码，将作业及其相关内容和配置提交到作业服务器。门户及任务调度服务器节点将作业分解成一个个可相对独立执行的任务(即任务之间尽量是松散耦合)，并将任务进行封装，然后将任务调度到合适的节点上运行。数据处理及存储服务器节点和其它加入云端计算环境中的用户终端节点都将充当任务执行者的角色，其是实际任务的承担者。

2.2 云端计算安全性问题分析

在云端计算这样动态的、分布式的计算环境中，联合属于不同所有者的节点来完成某一次大规模计算任务，就需要重点考虑系统的安全及鲁棒性等问题。不同于系统可直接集中管理控制的集群服务器节点，海量的终端节点分属于不同的用户，行为显然是不可靠的，计算安全性也难以保障。总之，云端计算环境是一种不安全的计算环境。

(1)从任务提交者的角度

由于云端计算环境中的任务执行者既包括数据处理及存储服务器节点，也包括用户终端节点，因此被集中管理的数据处理及存储服务器节点常常被认为是安全的。原因是这些节点一般属于大型的机构(如 Google、Microsoft、IBM 等)，这些机构具有广泛、良好的信誉，用户可以信任由它们管理的服务器节点。但是同样作为任务执行者的用户终端节点，则具有很强的动态性，难以控制，甚至有些是恶意节点。因此，在开放的云端计算平台上，对于任务提交者而言，存在以下的安全性问题：

1)计算完整性。当作业中的一个计算任务调度到某用户终端节点上时，该节点如果是恶意节点，则有可能篡改任务中的程序代码或数据，返回的是虚假的结果。

2)计算私密性。有计算私密性需求的计算任务常常有“让该用户执行但又不希望该用户了解该任务是什么”的特殊要求，如商业中的调查、统计、分析等计算项目。但是该任务发送到任务执行节点时，让该终端节点愿意执行但又不能窥探代码和数据，是一个难点。

(2)从任务执行者的角度

同样，作为任务执行者的节点在接受到承载用户计算任务的程序时，同样面临的问题是：

1)计算安全性。承载用户计算任务的程序是否存在安全问题。例如在本地执行是否会窃取本地用户的私密信息，甚至会导致嵌入病毒、后门等恶意代码。

2)计算可控性。用户希望能够通过贡献自己空闲的计算资源来提供服务，但绝不愿意因此而影响到自身的正常工作，这就必须能够有效控制任务的执行。

上述问题都是实现云端计算系统必须解决的安全问题，本文重点从任务提交者的角度对执行代码的安全保障机制进行研究。

3 开放云端计算环境中的任务执行代码保护机制

3.1 基于内嵌验证码的加密函数的代码保护机制

为了同时满足计算的完整性和私密性，使其能有效地验

证返回结果的正确性，并保障计算代码[6,7]不被通过反编译等手段窥知，本文提出一种新的基于内嵌验证码的加密函数的代码保护机制。下面详细阐述该机制的实现流程。为了方便描述，我们将任务提交者和任务执行者抽象为节点 A 和 B，如图 2 所示。

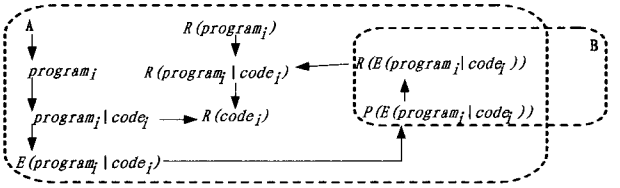

图 2　基于内嵌验证码的加密函数的代码保护机制示意图

①首先，任务提交者定义好本次提交的作业内容。作业 Job 为松散耦合的任务集合：

$$Job=\{task_1, task_2, \cdots, task_i, \cdots, task_m\} \tag{1}$$

②对于其中每一个任务（如 $task_i$），由任务提交者（也可由任务调度服务器节点）在任务 $task_i$ 的程序 $program_i$ 中嵌入验证性代码 $code_i$，即：

$$task_i ::= program_i | code_i \tag{2}$$

嵌入到正常的任务程序中的验证代码 $code_i$ 事实上也是一段程序代码，但该段代码的输出是任务提交者事先已经确定已知的。任务执行者收到任务 $task_i$ 并正常执行代码 $program_i$ 时，也必须顺带执行验证性代码 $code_i$。

③任务提交者对 Job 中的每一个任务的嵌入 $code$ 的 $program$ 利用加密函数 E 进行加密，使任务执行者或是其它恶意节点无法了解任务代码的内部逻辑。这种加密可以是对全部程序进行加密，或是对程序中的关键代码和验证码进行加密。设对 $task_i$ 的全部内容进行加密，则其加密后为 $E(program_i|code_i)$，然后构建专门引导执行加密后任务的程序 $P(E(program_i|code_i))$。

④$A \xrightarrow{P(E(program_i|code_i))} B$，节点 A 通过门户及任务调度服务器节点将加密后任务的程序 $P(E(program_i|code_i))$ 发给节点 B。

⑤$B \xrightarrow{R(E(program_i|code_i))} A$，节点 B 通过门户及任务调度服务器节点将结果（也是处于加密后）$R(E(program_i|code_i))$ 返回给节点 A。

⑥节点 A 对加密后结果 $R(E(program_i|code_i))$ 解密后，即得到 $R(program_i|code_i)$，并对该解密后结果进行分离，得到 $R(program_i)$ 和 $R(code_i)$。

由于验证性代码 $code_i$ 的结果对于节点 A 是已知的，如果得到的 $R(code_i)$ 与节点 A 已知的结果不相符，那么节点 A 将丢弃 $R(program_i)$，否则认为 $R(program_i)$ 是正常执行后得出的正确结果。

3.2　任务执行节点可信性评估机制

在上述的代码完整性和私密性保障机制的基础上，为了进一步提高任务执行的成功率和缩短作业周转时间，首先应该将任务代码优先分发给信誉良好且执行成功率高的节点来执行，这就涉及对任务执行节点的可信性进行科学评估。一种客观的思路[8,9]是根据节点的历史表现来进行评价，具体而言，以完成任务的成功次数 s、正常失误次数 w、提供虚假结果次数 f 为计算和评价依据。下面具体阐述。

如何判断任务的返回执行结果是正确的，即任务执行是成功的，这是首先要解决的问题。由于云端计算环境基于 Internet 且利用了海量的网络边缘节点作为任务的执行者，因此通过增加一定的冗余度，即选取多个终端节点来同时执行同一任务，或采用待定备份的方式，可以降低因为某一个任务的未实现而导致整体任务无法达成的概率。具体的步骤是：

Step 1　将同一任务 $task_i$ 发送至多个终端节点上执行。

Step 2　当侦听到第 1 个完成任务的节点提交的结果时，首先验证 $R(code_i)$。如果不能通过验证，则直接丢弃，重复 Step 2；如果通过验证，则继续等待第 2 个完成任务的节点提交的结果。

Step 3　当侦听到第 2 个完成任务的节点提交的结果时，将结果与第 1 个完成任务的节点提交的结果进行比对。

Step 4　如果相同，则采用该结果；如果不同，则继续等待第 3 个完成任务的节点提交的结果。

Step 5　当侦听到第 3 个完成任务的节点提交的结果时，将结果分别与第 1 个和第 2 个完成任务的节点提交的结果进行比对。

Step 6　采用与之相同的那个结果。如果不同，则转 Step 5 反复运行，直至找到相同的值为止。

在上述流程结束时，由任务调度服务器汇总任务执行节点的行为表现，并更新如表 1 所列的存储于服务器端的节点可信矩阵。

表 1　节点可信矩阵

节点	次数		
	s	w	f
$Node_1$	s_1	w_1	f_1
$Node_2$	s_2	w_2	f_2
…	…	…	…
$Node_{n-1}$	s_{n-1}	w_{n-1}	f_{n-1}
$Node_n$	s_n	w_n	f_n
合计	$\sum_{i=1}^{n} s_i$	$\sum_{i=1}^{n} w_i$	$\sum_{i=1}^{n} f_i$

在挑选承担当前任务的终端节点时，可以采取的策略为：

(1)首先选择 f 值最小的最可信节点集合；

(2)然后依据节点的成功次数和正常失误次数来综合得出节点可信度：

$$T_i=\frac{s_i-\mu w_i-\eta f_i}{\sum_{k=1}^{n} s_k} \tag{3}$$

以节点可信度来对可信节点集合中的节点进行排序，优选排在序列前面（即 T 值高的）的节点作为任务执行者。上式不单独考虑成功次数和正常失误次数，是因为成功次数多的节点可能失误次数也较多，但是失误少的节点可能是因为参加的执行任务次数也少。式中的 μ 作为对失误严重程度的评估调节因子，而 η 作为对提供虚假结果行为的评估调节因子。

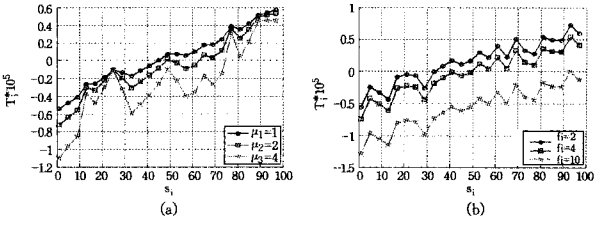

图 3　式(3)实验仿真结果

图 3(a)为在 f 值及 η 参数固定的情况下 w_i 取值为一个随机的失误发生概率时对式(3)进行实验仿真的结果。横轴为成功次数 s_i，为了方便观察，纵轴为 T_i 值乘 10^5 的结果。从图中可以看出，在参数 μ 取值不同的情况下，T_i 均随着 s_i 值增加；同时也显示即使 s_i 值较高，但如果 w_i 值也较高，T_i 值会有所下降；另外，参数 μ 的调节作用也显示明显。以上都显示式(3)与实际情况相符。

图 3(b)为在参数 μ 值固定的情况下 f 值不同时对式(3)进行实验仿真的结果。结果显示，提供虚假结果次数多的节点具有低的可信度，这也显示式(3)与实际相符。

4 系统安全性分析与验证

机制的关键是要为任意一段代码找到一个适合的加密方法，然而目前的研究还未能找到通用的加密方案，但是可以证明对多项式和有理函数的计算是完全可行的。一个典型的做法[6]是：A 希望 B 完成某一任务 X，但 A 不希望 B 了解他的代码的内部逻辑和实现的目标。因此 A 选取可逆矩阵 V，并计算出 $Y=V\cdot X$；A 将 Y 送给 B，B 利用本地数据 x 等计算资源计算 $Z=Y(x)$，并将 Z 返回给 A。A 通过计算 $V^{-1}\cdot Z$ 就得到任务的求解结果。通过这种方式可以使 A 的计算代码不至于暴露给 B。

4.1 分析与验证 1

场景：破坏计算完整性。当 A 提交的作业中的一个计算任务调度到用户终端节点 B 上时，节点 B 不愿提供自己的计算资源，而是简单伪造虚假的计算结果，或是篡改任务中的程序代码或数据后再执行计算任务，从而获得虚假结果并返回。

安全性分析：在本文提出的模型中，由于在程序里面引入了验证码，并且验证码和任务程序都是经过加密的，因此如果返回的是伪造虚假的计算结果，必然得不到正确的验证码执行结果，而该结果是节点 A 已知的。如果经过比对发现验证码结果错误，可以立刻判定结果难以相信，然后丢弃。退一步而言，假设验证码被获知，由于模型引入了冗余机制，即并非简单相信某一个边缘节点的计算结果，同样可以进一步保障最后确认的计算结果是正确的。另外，通过评估节点的可信度，可以随着系统的运行进一步推动云端计算环境的良性发展，逐步消除低可信度的恶意节点。

4.2 分析与验证 2

场景：破坏计算私密性。当作为任务执行者的节点 B 收到 A 提交的作业中的一个计算任务(如某电信运营商对其客户资源的分析)时，试图通过反编译等手段来获取计算任务中的代码和数据，从而了解任务提交者的目的，并将代码、数据等信息透露给其它的运营商。

安全性分析：在本文提出的模型中，节点 B 收到的计算任务的代码和数据已经是全部或是针对其中的关键部分加密后的内容，而执行任务获得的结果也是经过加密的。如上面的典型例子中，节点 B 不知道 V 和 V^{-1}，因此 B 既无法知道任务的代码、数据，又无法得到任务的执行结果。系统的计算私密性得到了良好的保证。

结束语 传统的云计算环境中由于所有的计算节点集中管理，因此对其中的安全问题大多数可以参照分布式计算系统的安全解决方案和技术来解决。但是如果将云计算拓展为云端计算，由于引入了“端”节点，带来的安全隐患必须重视并解决，否则利用云端计算最大程度地聚集服务器和边缘节点的资源来获得高效益的目标则不具有可行性。本文的研究成果解决了其中的一项问题，即保证分发到异地执行的代码的计算完整性和计算私密性。重点是提出一种新的基于内嵌验证码的加密函数的代码保护机制和节点可信度评估方法，以有效验证返回结果的正确性，并保障计算代码不被非法窥知。

下一步的研究工作包括：

(1)保障计算代码的安全性，即能够有效检测承载用户计算任务的程序中的安全问题，防止恶意代码窃取本地用户的私密信息，甚至嵌入病毒、后门等。

(2)保障计算代码可控性。任务执行节点应该有效控制任务代码的正常执行，而不会对自身正常的工作造成负面的影响，避免成为拒绝服务式的攻击目标。

(3)保障云端计算的可持续性，即能够长久刺激网络边缘节点愿意积极、诚实地提供资源和服务。

参 考 文 献

[1] 陈康，郑纬民. 云计算：系统实例与研究现状[J]. 软件学报，2009，20(5)：1337-1348

[2] Barroso L A, Dean J, Hölzle U. Web search for a planet: The Google cluster architecture[J]. IEEE Micro, 2003, 23(2): 22-28

[3] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data[C]// Proc. of the 7th USENIX Symp. on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2006: 205-218

[4] Dean J, Ghemawat S. Distributed programming with Mapreduce [M]// Oram A, Wilson G, eds. Beautiful Code. Sebastopol: O'Reilly Media, Inc., 2007: 371-384

[5] 徐小龙，程春玲，熊婧夷. 基于 Multi-Agent 的云端计算融合模型的研究[J]. 通信学报，2010，31(10)：203-211

[6] Fritz H. Time limited blackbox security: Protecting mobile agents from malicious hosts[C]// Vigna G, ed. Mobile Agents and Security, LNCS 1419. New York: Springer-Verlag, 1998: 92-113

[7] Sander T, Tschudin C F. Protecting Mobile Agents Against Malicious Hosts[C]// Vigna G, ed. Mobile Agents and Security, LNCS 1419. New York: Springer-Verlag, 1998: 44-60

[8] 窦文，王怀民，贾焰，等. 构造基于推荐的 Peer-to-Peer 环境下的 Trust 模型[J]. 软件学报，2004，15(04)：571-583

[9] Kamvar S D, Schlosser M T. EigenRep: Reputation management in P2P networks[C]// Proc. of the 12th International World Wide Web Conf. Budapest: ACM Press, 2003: 123-134