doi:10.3969/j.issn.1000-436x.2014.01.005

基于动态域划分的 MapReduce 安全冗余调度策略

沈晴霓[1,2]，卿斯汉[1,2,3]，吴中海[1,2]，张力哲[1,2]，杨雅辉[1,2]

(1. 北京大学 软件与微电子学院，北京 102600；2. 北京大学 网络与软件安全保障教育部重点实验室，北京 100871；
3. 中国科学院 软件研究所，北京 100190)

摘 要：MapReduce 现有调度策略无法实现云环境中多租户作业的安全隔离。提出一种基于动态域划分的安全冗余调度策略：通过引入冲突关系、信任度、安全标签等概念，建立一种动态域划分模型，以将待调度节点划分为与不同租户作业关联的冲突域、可信域或调度域；结合冗余方式，将租户作业同时调度到其可信域节点和调度域节点(但不允许为其冲突域节点)，通过二者执行环境和部分计算结果的一致性验证决定是否重新调度。实验分析了其有效性和安全性。

关键词：云计算；MapReduce 框架；动态域划分；安全冗余调度

中图分类号：TP309 **文献标识码**：A **文章编号**：1000-436X(2014)01-0034-13

Securely redundant scheduling policy for MapReduce based on dynamic domains partition

SHEN Qing-ni[1,2], QING Si-han[1,2,3], WU Zhong-hai[1,2], ZHANG Li-zhe[1,2], YANG Ya-hui[1,2]

(1. School of Software and Microelectronics, Peking University, Beijing 102600, China;
2. MoE Key Lab of Network and Software Assurance, Peking University, Beijing 100871, China;
3. Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: MapReduce's current scheduling policies could not ensure the isolation between multi-tenant Tasks in the cloud. A securely redundant scheduling policy based on dynamic domains partition was proposed. First, a kind of dynamic domain partition model was introduced in this policy. Based on the node's current belief, security labels with the conflict relationship between tenants, a computing node was partitioned into the conflict domain, trusted domain or schedulable domain in this model. Second, through redundantly computing, two copies of each Task were assigned respectively to its trusted domain node and its schedulable domain node (but not allow for its conflict domain node) in this policy. And the integrity of the two nodes'execution environments and the consistence of their results on a small part of original input data were verified. Accordingly, it decided whether the schedulable domain node was trusted. Finally, the performance and security analysis in the prototype show its effectiveness.

Key words: cloud computing; MapReduce framework; dynamic domain partition; securely redundant scheduling

1 引言

MapReduce 是 Google 设计的并行数据处理模型[1](如图 1 所示)，它主要由一个中央节点(JobTracker)和若干计算节点(TaskTracker，也称 node，可以是物理机器，也可以是虚拟机)组成，用户提交的作业(Job)首先在中央节点被划分成多个任务(Task，分为 Map 和 Reduce 2 个阶段)，中央节点根据一定的调度策略将这些 Map/Reduce 任务分配给不同的计算节点来完成，这些任务并行地对各自的输入数据块、中间结果进行处理后产生最终输出结果(均为 Key-Value 形式)。近年来，许多公司使用 MapReduce

收稿日期：2013-08-20；**修回日期**：2013-11-10

基金项目：国家自然科学基金资助项目(61232005，61073156，61070237，61170282)；国家科技支撑计划基金资助项目(2008BAH33B02)

Foundation Items: The National Natural Science Foundation of China (61232005, 61073156, 61070237, 61170282); The National Science and Technology Supporting Program (2008BAH33B02)

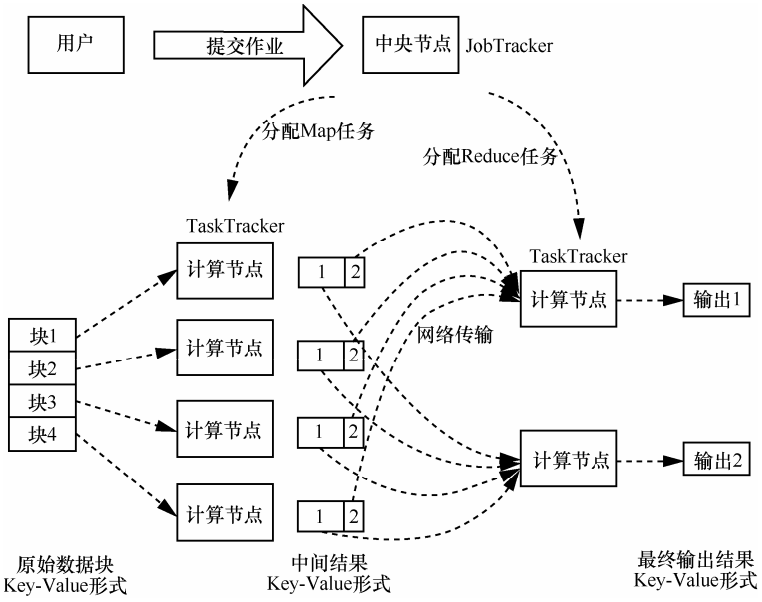

图 1　MapReduce 并行计算模型

模型处理自己的数据业务，包括网页搜索、高端计算、数据分析和机器学习等。随着目前云计算模式的推出和迅速发展，提供以 MapReduce 为基础的可扩展、低成本的海量数据处理开放式服务的需求成为了业界发展趋势[1~4]。

但是，使用 MapReduce 的云服务系统是开放式、多租户的基础架构（租户是指租赁云服务系统的计算/存储服务的实体，一个租户是一个组织/机构/个体，如银行、企业、高校或个人），除了传统的通信安全威胁，如数据窃听、重放攻击以及拒绝服务攻击等，还存在自身安全的复杂性和特殊性[2~4]，特别地，在 MapReduce 中央节点对大规模不同租户的作业进行调度的过程中，它可能面临如下新的安全问题。

1) 租户 *X* 和租户 *Y* 是竞争关系，但是 *X* 的任务 T*x* 与 *Y* 的任务 T*y* 可能在某个时刻被同时分配到同一个计算节点 *N*，从而导致一方(T*x* 或 T*y*)信息被另一方(T*y* 或 T*x*)恶意窃取或篡改。

2) 租户 *X* 的任务 T*x* 可能被分配到一个已被恶意控制的计算节点 *N* 上，导致节点 *N* 的控制者可以任意篡改 T*x* 的计算结果或者直接返回一个错误的结果，从而欺骗中央节点或者租户 *X*。

3) 租户 *X* 的任务 T*x* 可能被分配到一个计算节点 *N*，而节点 *N* 中正在运行某个租户 *Z* 的一个被注入了恶意代码的任务 T*z*，导致 *Z* 可以利用 T*z* 来破坏 T*x* 的计算环境和类似 2）的计算结果的完整性。

然而，在面向云服务的 MapReduce 计算框架中，目前采用的调度策略重点关注的仍然是作业的优先级、资源的利用率、资源分配的公平性、调度的实时性、异构环境的支持能力等问题[5~11]，对其中可能面临的上述安全性问题却考虑很少。

本文的创新之处在于以下 3 点。

1) 提出一种动态域划分模型。通过引入冲突关系、信任度和安全标签等概念，以及为不同租户作业定义 3 种域划分策略，将每个待调度节点划分为与租户作业关联的冲突域、调度域或可信域。

2) 提出一种基于动态域划分的安全冗余调度策略，包括：①根据动态域划分模型和冗余方式[8~10]，将每个任务的 2 个副本（副本 1 和副本 2）同时分配给与其作业关联的调度域节点和可信域节点(但不允许是冲突域节点)，从而保证冲突关系租户的计算任务不会被同时调度到同一个计算节点；②通过任务副本 1 在可信域节点上执行环境散列校验值，以及它对随机选取的小部分输入的计算结果散列校验值，对比任务副本 2 在调度域节点上的任务执行环境和计算结果散列校验值，如果不一致，则重新调度该任务，如果一致，则认为调度域可以继续完成当前任务。从而保证合法租户的计算任务不会调度到可能是恶意租户的节点或可能运行着恶意租户任务的节点。

3) 通过在 Hadoop MapReduce 调度器中的原型实现、性能测试和安全性分析论证了策略的有效性。

2 相关工作

目前，Hadoop MapReduce 的调度策略主要分为 3 种类型：基于优先级的调度策略(FIFO)[2]、基于资源利用率优化的调度策略(capacity scheduler)[5]和基于资源分配公平性的调度策略(fair scheduler)[6,7]。其中，基于优先级的调度策略按任务的优先级高低调度计算节点；基于资源利用率优化的调度策略通过设置多个作业队列并行调度计算节点，提高资源的利用率；基于资源分配公平性的调度策略通过设置每个作业可调度的资源池相等，实现公平的调度。此外，为了避免同构集群中少数任务落后导致整个作业进度变慢问题，Hadoop 调度器[2]提供一种冗余调度策略，它监控每个任务的进度，如果一个任务的进度明显落后于同类型任务进度(如落后 20%)，则把它当成落后任务，为它启动一个备份任务，二者同时执行，谁先完成和提交，则采取谁的结果。LATE 调度器[8]通过设置可同时执行的备份任务上限等，解决异构集群中同类型 Task 进度差异大时的性能优化问题。软、硬实时调度器[9~11]保证有时间限制要求的作业在规定时限内完成。但上述策略均很少考虑安全性问题。

Wei W 等[12]曾经为 MapReduce 提出一种基于冗余计算方式的安全调度策略，设计了非集中式的委托协议和验证协议，目标是保证数据处理过程的完整性。它在为一个计算任务同时分配 2 个计算节点的调度策略中重点考虑 2 个方面的安全问题：一是，如何保护任务分配消息的完整性和机密性，防止恶意的节点窃取好的节点的任务分配信息或任意伪造任务信息达到拒绝服务攻击(DoS)的目的；二是，如何通过时间戳和序列号自动生成的任务 ID 信息、签名保护机制防止对分配消息的重放攻击。尽管它给出的完整性验证协议可以在检测 2 个计算节点结果出现不一致的情况下，向中央节点报告它们可能存在潜在安全风险，但它采用的是完全冗余计算方式，且并没有给出如何将这种检测结果应用到安全调度策略中，以避免将后续的任务分配给这些恶意的节点。Roy I[13]等设计实现一个基于强制访问控制和差分隐私保护机制的安全 MapReduce 系统——Airavat，使得基于敏感数据上的 MapReduce 计算，不论其代码(Jobs)是否可信，都可以保证数据提供者所要求的隐私保护策略强制执行。但是 Airavat 重点考虑的是如何保证任务运行过程中不同租户数据的隐私性，而不是任务调度决策过程中的相互安全隔离性问题。Song S 等[14]针对异构网格计算环境中 2 种启发式作业调度算法的可信性需求，提出了 3 种安全调度模型：第 1 种是安全(secure)模型，总是分配任务给可信的计算节点(一个计算节点是可信的，当且仅当计算节点的安全级(SL)不低于被调度作业的安全需求(DS))；第 2 种风险(risky)模型，分配作业给任何可用的计算节点，无论会有什么样的风险；第 3 种是 f 风险模型，尝试将作业分配的冒险率限制在 f 以内(其中，f=0 为 secure 模型，f=1 为 risky 模型)；但是这 3 个模型针对的是来自网格环境中不同组织的计算节点本身可能是恶意的或不可信的，目的是防止其导致的作业调度失败或再分配等安全性问题。此外，针对 MapReduc 任务计算结果易被篡改和伪造问题，Ruan A 等[15]提出一种并行的远程证实机制来保证作业输出结果的完整性。HUANG C 等[16]提出一种水印植入和随机抽样相结合的方法来检测具有恶意/欺骗行为的节点。KHAN S M[17]等采用可信环境对不可信环境的断点检查方法形成计算结果的完整性证据，BENDAHMANE A 等[18]采用副本投票方法和信任管理系统来保证计算结果的完整性。但是这些工作的主要目标只是在计算任务的执行过程中保证计算结果的完整性。

1989 年，Brewer D 等[19]提出一种中国墙模型，旨在同时解决当时信息系统中的机密性和完整性保护问题。目前，该模型又被看成最适用云环境资源分配和管理的安全模型之一[20~23]。例如，已经有一些研究者采用中国墙模型的利益冲突类思想给出云存储中的强制访问控制策略[20,21]，实现数据集合在不同物理机器之间的强隔离性[22]。特别地，Chen Y C[23]等提出一种基于中国墙模型竞争关系的虚拟机安全部署策略，通过禁止竞争关系企业的虚拟机被部署到同一台物理机器上，防止云环境中的虚拟机之间产生信息泄漏等安全问题。但是，上述策略考虑的只是云计算环境部署过程中的虚拟机和被存储数据的安全性问题。

因此，以上提出的方法主要针对的是作业调度信息本身和调度之后任务执行过程中的信息安全性问题，或者依据安全级别保证作业被调度节点的可信性问题，或者云计算环境部署过程中的虚拟机和被存储数据的安全性问题。这些方法均没有在作业调度时考虑云环境中多租户的动态安全隔离问

题。本文策略重点解决这个问题，使存在冲突的多租户计算任务不会同时被分配给同一个计算节点，使合法租户的计算任务不会被分配给一个可能是恶意租户的计算节点或者可能运行着恶意租户任务的计算节点。

3　动态域划分模型

本节将提出一种动态域划分模型。通过引入冲突关系、信任度和安全标签等概念，以及为不同租户作业定义的 3 种域划分策略，将每个待调度节点划分为与租户作业关联的冲突域、调度域或可信域。

3.1　基本概念

定义 1　冲突(conflict)关系：是 2 个租户之间的一种关系，指的是 2 个租户 u_i 的 u_j 在请求各自的作业调度过程中，都不希望与对方的作业被同时分配到同一个计算节点上，即不希望双方的计算任务在同一个时刻运行在同一个计算节点上。同时，为了保证云环境计算资源具有较高的利用率，这 2 个租户并不反对在不同时刻将它们的计算任务被分配到同一个计算节点上运行。则将这 2 个租户之间的关系(u_i, u_j)称为冲突关系，且这种关系要求具有反自反性和对称性，但不要求可传递性。如 Alice 银行和 Bob 银行是同一个云计算环境的 2 个租户，但是它们在业务上明显存在利益竞争关系，那么 Alice 和 Bob 都不希望它们各自租用的计算节点在完成自己的业务（即作业）过程中有对方的任何业务在该节点上同时运行，因为他们会担心对方恶意窃取或篡改自己业务执行过程中用到私有数据，则(Alice 银行、Bob 银行)就属于这种冲突关系。如果(Alice 银行、Clark 公司)也属于冲突关系，则并不应推导出(Bob 银行、Clark 公司)是冲突关系。

为此，系统需要统一管理和维护一个冲突关系

表 1　模型元素和域划分策略逻辑表达式的构造

元素集	元素	说明
U	$\{U_1, U_2,\cdots,U_n\}$	租户
J	$\{J_1, J_2,\cdots,J_p\}$	作业
T	$\{T_1, T_2,\cdots,T_q\}$	任务(如 Map/Reduce 任务)
N	$\{N_1, N_2,\cdots,N_m\}$	计算节点(TaskTracker)
C	$C \subseteq U\times U$	租户之间的冲突关系
B	{1, 2, 3, 4}	租户对计算节点的信任度
S	$\{S_1, S_2,\cdots,S_n\}$	计算节点的安全级别(由系统配置为 $C1<C2<B1<B2<B3<A1$ 或低<中<高)
G	$\{G_1, G_2,\cdots,G_n\}$	计算节点的位置标识
GH	$GH \subseteq G\times G$	位置标识之间的层次化语义包含关系
L	$\{L_1, L_2,\cdots,L_k\}$,其中,$L_i \subseteq (S_i,G)$	计算节点的安全标签
D	$\{D_C, D_S, D_T\}$	计算节点可划分的域：冲突域 D_C、调度域 D_S 和可信域 D_T（对应特定作业）
jUser	*jUser*: $J\rightarrow U$	获取指定作业所属的租户
jTasks	*jTasks*: $J\rightarrow 2^T$	获取属于指定作业的计算任务
nJobs	*nJobs*: $N\rightarrow 2^J$	获取指定计算节点当前正在运行的作业
nUsers	*nUsers*: $N\rightarrow 2^U$	获取指定计算节点当前作业的租户信息
nLabel	*nLabel*: $N\rightarrow L$	获取指定计算节点当前标记的安全标签
nBelief	*nBelief*: $U\times N\rightarrow B$	获取指定租户对指定计算节点的当前信任度
tCurDom	*tCurDom*: $T\times N\rightarrow D$	获取指定节点当前被划分的特定域（对应特定任务）
nDoms	*nDoms*: $J\times N\rightarrow 2^D$	获取指定节点可被划分的若干域（对应特定作业）
tNode	*tNode*: $T\times D\rightarrow N$	获取指定任务被分配的具有特定域划分的计算节点
E	$E:=\varphi\|2^L\|2^B\|(E)\ \|E\&\&E\|E\|\|E\|!E$	策略表达式由空集、信任度、安全标签，以及带括号的逻辑表达式、逻辑表达式的与(&&)或(||)非(!)操作等构造
$P(U,J)$	$\{P(U,J)_C, P(U,J)_S, P(U,J)_T\}$,其中,$P(U,J)_x:=E$	对应不同租户不同作业的 3 个域划分策略：$P(U,J)_C$, $P(U,J)_S$, $P(U,J)_T$，每个域划分策略表示为一个逻辑表达式

集合 C(如表 1 所示)，并且在每个任务调度过程中检查待调度计算节点上当前运行任务所属租户，检查他们是否与待调度任务所属的租户之间存在冲突关系。

定义 2 信任度(belief)：是关于一个计算节点在过去一段时间内的行为是否满足租户的可信性期望的一种度量。其中，租户的可信性期望是指分配租户作业的计算节点总是能够同时满足相关的 2 个可信性属性，即任务执行环境的完整性和计算结果的正确性(见第 4 节)。作业调度器在每次决定待调度节点应分配的计算任务时，需要收集该节点的 2 个可信性属性状态，在完成了一个作业(多个任务)的分配时，综合评估和动态更新(增/减)相关计算节点的信任度。这里给出一种基于 PTM 信任模型[24,25]的信任度更新算法。系统需要设置每个节点针对不同租户的初始信任度，并在作业 j 调度完成时，将其租户 u 对被调度节点 p 的信任度 $b_{i-1}^{(u,p)}(\in[0,1])$ 更新为 $b_i^{(u,p)}$：

$$b_i^{(u,p)}=\begin{cases} b_{i-1}^{(u,p)}+\lambda\times\omega\times V_i\times(1-b_{i-1}^{(u,p)}),\ V_i>0 \\ b_{i-1}^{(u,p)}(1-\lambda\times\omega+\lambda\times\omega\times V_i),\ V_i\leqslant 0 \end{cases}$$

其中，$V_i=W_j\times\dfrac{m-r}{m+r+1}$，权值 $W_j\in[0,1]$ 表示的是作业 j 对租户的重要性；严格因子 $\omega\in[0.25,\ 0.75]$ 是一个可以由租户配置的参数，代表租户对计算节点期望的信任值；$\lambda\in[0.5,1]$ 表示对计算节点的可信性属性变化进行正确验证的概率，如执行环境的完整性验证依赖于散列函数(MD5/SHA-1)的安全性和对散列值保护机制的强度(通过 TPM 物理芯片与通过软件来保护的强度不同)，而计算结果的正确性验证依赖于输入数据的选取和作业中任务数量的选取(详见第 4 节)；$m(0\leqslant m)$表示分配给了节点 p 且在其上运行时满足了上述 2 个可信性属性的任务数量，$r(0\leqslant r)$则表示分配给了节点 p 但在其上运行时没有满足上述 2 个可信属性的任务数量。这里没有考虑与 p 承担过相同租户作业的其他节点的信任推荐，以防止恶意推荐。

为此，系统定义了租户对计算节点的信任度集合 B(如表 1 所示)，值域规约为：4($T\in(0.75,1)$)，3($T\in(0.5,0.75)$)，2($T\in(0.25,0.5)$)和 1($T\in[0,0.25]$))。

定义 3 安全标签(label)：借鉴多边安全思想[26]，这里的安全标签是指系统为计算节点(虚拟机或物理机)标记的安全属性，由 2 部分组成：安全级别(S)和位置集合(G)，其中，安全级别 S 主要是指计算节点的软件运行环境的安全级别（根据系统的安全评估等级来划分，可分为 $C1<C2<B1<B2<B3<A1$；根据系统的网络环境配置来划分，可分为“低”<“中”<“高”），通常在系统初始化时设置，或在运行过程时由云管理员设置；位置集合表示计算节点的服务区域，即一个计算节点可同时标记多个位置标识（如“北京”、“天津”等），但每个计算节点位置在云环境中存在大规模和动态的特点，需要做到：1）每个位置标识具有唯一的语义（如不会存在“China”、“中国”具有等价语义的 2 个标识）；2）明确不同位置标识之间存在的语义包含关系，即将它们按照层次化树型结构组织，树上每一个非根子节点能继承其所有祖先节点的值（如图 2 所示）。因此定义 2 个安全标签 $L_1=(S_1,G_1)$，$L_2=(S_2,G_2)$之间的包含关系如下。

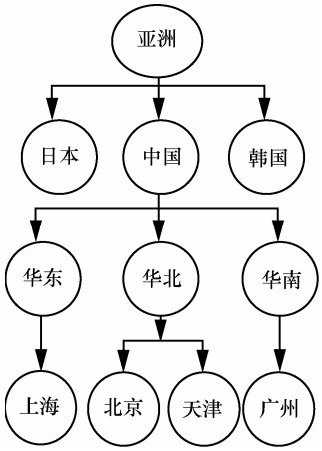

图 2 位置属性之间语义层次化树型关系

如果说 $L_1=(S_1,\ G_1)$包含 $L_2=(S_2,\ G_2)$，记作 $L_1\geqslant L_2$，当且仅当它们能够同时满足：$S_1\geqslant S_2$ 且 $G_2\subseteq G_1$；其中，G_1 和 G_2 是扩展了继承关系的位置集合，如：G_1={北京} => G_1={北京，华北，中国，亚洲}。

例如：若一个计算节点 X 标记为 $L_1=(B1,\{$北京$\})$，通过位置属性的扩展后，$L_1\backsimeq(B1,\ \{$北京，华北，中国，亚洲$\})$。当系统希望调度安全标签为 $L_2=(B1,\ \{$华北$\})(\backsimeq(B1,\ \{$华北，中国，亚洲$\})$的计算节点时，L_1和L_2满足包含关系定义中的2个条件，即 $L_1\geqslant L_2$。若系统希望调度安全标签为 $L_2=(B1,\ \{$中国，韩国$\})\ (\backsimeq(B1,\ \{$中国，韩国，亚洲$\})$的计算节点，则 L_1 和 L_2 之间不满足条件，记作 $L_1\leqslant\geqslant L_2$。3.3 节将在作业调度过程应用这种包含关系，以确

定待调度计算节点的安全标签是否满足租户作业的域划分策略。

因此，系统将通过可为计算节点标记的安全级别和位置集合等属性，定义一个安全标签集合 L（如表 1 所示），用于标记每个计算节点的安全属性。

定义 4　策略表达式（policy expression）：是指系统将为每个租户作业定义的一种策略逻辑表达式 E(如表 1 所示)，它由空集、信任度、安全标签，以及带括号的逻辑表达式、逻辑表达式的与(&&)或(||)非(!)操作等构造而成。由于云服务系统中存在大规模、多样化的计算资源，并且每个租户的作业都需要通过大量的计算资源来并行调度和执行，如果系统仅分配严格受限的少量计算资源来完成，势必影响租户使用云服务的体验。为此，系统应该提供一种比较灵活的策略来保证计算资源的可用性，因此对动态域划分使用的策略采用了具有灵活组合特点的逻辑表达式来构造。例如：假设系统通过与租户 Alice 协商之后，明确了 Alice 对其提交作业 Job 在调度过程中的安全需求，其中要求：1）节点的安全标签能够包含 L=(C1，{中国})，而且 Alice 对节点的信任度不低于 2；2）节点的安全标签能够包含 L=(C2，{韩国})或者包含 L=(C2，{日本})，而且 Alice 对节点的信任度不低于 3。为此，系统将对应的策略表示为：P=(‘2’&&(C1，{中国}))‖ (‘3’ &&((C2，{韩国})‖(C2，{日本}))。如果待调度计算节点 X 实际标记的 2 个属性为：Alice 对 X 的信任度为 2，计算节点当前标记的安全标签为(C2，{北京})，则根据信任度线性关系和安全标签的包含关系定义，该策略逻辑表达式的计算结果为真(true)，这表示节点 X 满足了 Alice 对 Job 调度的安全需求。

3.2　模型元素及其相互之间的约束关系

系统将通过策略表达式 E(如表 1 所示)来表达租户对作业待调度节点的安全需求。这也是本文动态域划分模型的重要基础（见 3.3 节和 4 节），域划分策略表达式的结果将决定一个计算节点是否被调度。因此，如表 1 所示，动态域划分模型相关的模型元素主要包括：租户集合(U)、作业集合(J)、任务集合(T)、计算节点集合(N)、租户之间的冲突关系(C)、信任度集合(B)、安全标签集合(L)、可划分域集合(D)、域划分策略集合 $P(U, J)$等(具体涵义和主要操作见表 1)。此外，这些模型元素之间的关系必须满足以下要求（如图 3 所示）。

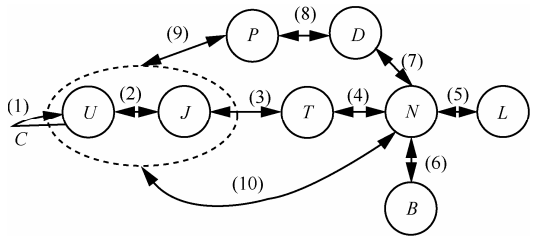

图 3　动态域划分模型元素及其关系

1) 每一个租户都可能和若干个其他租户之间存在冲突关系。

2) 每一个租户可以提交若干个作业，每一个作业必须属于一个租户。

3) 每一个作业可以划分为多个任务，每一个任务必须属于一个作业。

4) 每一个任务可以分配给若干个计算节点，每一个计算节点可以同时运行多个任务。

5) 每一个计算节点被赋予一个安全标签，每一个安全标签可以赋予若干个计算节点。

6) 针对每一个租户，一个计算节点有且只有一个信任度，针对不同租户，每个计算节点的信任度可以不同；针对同一个租户，具有相同信任度的计算节点可以有多个。

7) 针对每一个租户作业，一个计算节点能且只能划分为一个域，同一个域划分的计算节点可以有若干个。

8) 针对每一个租户作业，每一个域划分策略关联一个特定域，每一个域有一个对应的域划分策略。

9)每一个租户作业有 3 个域划分策略，每一个域划分策略属于一个租户作业。

10) 每一个租户能为其特定作业申请若干个计算节点，每一个计算节点能承担若干个租户的特定作业。

3.3　冲突域、调度域和可信域的划分策略和规则

根据 3.1 节和 3.2 节的描述，下面具体给出动态域划分模型的主要设计思想：首先，针对不同租户对不同作业的安全需求，系统将为每个作业定义如下 3 个策略。

冲突域划分策略 $P(U, J)_C$，除了冲突关系集合 C 之外，将通过专门定义的一种冲突域划分策略表达式（见 3.1 节）来进一步约束冲突计算节点的属性。例如，租户 Alice 与租户 Bob 为冲突关系，即(Alice, Bob)$\in C$，同时，若定义 $P(\text{Alice}, J_a)_C$= ‘1’&& (C1,ϕ)，则说明 Alice 不希望将它的作业 J_a 分配给任何正在运行 Bob 作业且具有这种属性的计算节

点，而不是正在运行 Bob 作业的所有计算节点。反之，如果 $P(\text{Alice}, J_a)_C=\phi$，则意味着 Alice 不希望将它的作业 J_a 分配给正在运行 Bob 作业的所有计算节点。这样冲突关系的约束就比较灵活。

调度域划分策略 $P(U,J)_S$，也是通过专门定义的一种调度域划分策略表达式来约束可调度节点的属性，它只是希望从原有调度策略已经筛选出来的待调度计算节点中进一步找出符合租户对作业安全需求的计算节点，通常地，$P(\text{Alice}, J_a)_S=\phi$，此时表示所有待调度节点都可以作为可调度节点，以保证计算资源的高利用率。当然，系统也可以根据一些特殊租户(如银行、证券)的要求，对可调度节点的要求做出进一步的约束，如 $P(\text{Alice}, J_a)_S$ = '2'&& ($C2$, {中国})，则表示 Alice 希望将它的作业 J_a 分配给位置在中国、安全级别为 $C2$，而且租户对它的历史信任度已经达到 2 的计算节点，从而提高租户作业的安全性。

可信域划分策略 $P(U,J)_T$，同样将通过专门定义的一种可信域划分策略表达式来约束可信节点(用于冗余计算，见第 4 节)的属性，其中会对表达式中的信任度进行最小限制，比如：要求它不低于调度域表达式中的信任度最大值，同时也会对安全标签进行最小限制，比如：要求它包含调度域表达式中最大安全标签。假设 $P(\text{Alice}, J_a)_S$='2'&& ($C2$, {中国})，则 $P(\text{Alice}, J_a)_T$='1'&& ($C1$, ϕ)将不符合要求，但 $P(\text{Alice}, J_a)_T$ ='3'&& ($C2$, {北京})则符合要求，因为'3'>'2'且($C2$, {北京})≥($C2$, {中国})，从而保证系统为 Alice 的作业 J_a 找到合适的可信节点。

再者，根据系统为租户作业定义的上述 3 个策略、计算节点的当前状态以及待调度的计算任务，将计算节点划分为与任务关联的一个域，即冲突域 D_C、可信域 D_T 或调度域 D_S。为了便于表达，将请求调度的作业表示为 $j\in J$，租户信息记为 $u=jUser(j)\in U$，当前待调度计算节点为 $x\in N$，节点 x 的当前状态为：u 对 x 当前信任度 $nBelief(u,x)\in B$，x 安全标签 $rLable(x)\in L$，及其上正在运行作业 $nJobs(x)\subseteq J$ 等。系统为租户 u 作业 j 定义的 3 个策略为 $P(u,j)_C$、$P(u,j)_S$、$P(u,j)_T$，假设节点属性满足的域划分策略至少有一个，即$|nDom(x,j)|\geq 1$，且目前待调度任务为 $t\in Tasks(j)$，则将 x 节点划分为与 t 关联的一个域的几个规则如下。

冲突域划分规则 节点 x 被划分为任务 t 的冲突域节点，即 $nCurDom(x, t)=D_C$，当且仅当对于节点 x 上正在运行的所有作业集合 $nJobs(x)$，至少存在一个作业 $j'\in nJobs(x)$，j'的租户 $u'=jUser(j')$与 u 属于冲突关系，即$(u, u')\in C$，且节点 x 当前安全属性 $nBelief(u, x)$, $rLable(x)$能够使租户 u 的作业 j 对应的冲突域划分策略 $P(u, j)_C$ 的值为真，即 $P(u, j)_C$ = true。

可信域划分规则 节点 x 被划分为任务 t 的可信域节点，即 $nCurDom(x, t)=D_T$，当且仅当节点 x 当前安全属性 $nBelief(u, x)$, $rLable(x)$能够使租户 u 的作业 j 对应的可信域划分策略 $P(u, j)_T$ 的值为真，即 $P(u, j)_T$=true，且待调度任务 t 没有分配给 j 对应的任何可信域节点，即 $tNode(t, D_T)=\phi$。

调度域划分规则 节点 x 被划分为任务 t 的调度域节点，即 $nCurDom(x, t)=D_S$，当且仅当节点 x 当前安全属性 $nBelief(u, x)$, $rLable(x)$能够使租户 u 的作业 j 对应的调度域划分策略 $P(u, j)_S$ 的值为真，即 $P(u, j)_S$=true，且任务 t 已经分配给了 j 对应的一个可信域节点，即 $tNode(t,D_T)\neq\phi$。

4 MapReduce 安全冗余调度策略

4.1 威胁模型和基本假设

目前，针对 MapReduce 的攻击手段可以分为以下 3 类。第 1 类是鲁莽攻击，即攻击者控制计算节点，总是返回错误的结果；第 2 类是普通攻击，即攻击者以一定的概率（而不是 100%）返回错误的结果，本类攻击比较常见，比第 1 类攻击有更高的成功率，也更难以识破；第 3 类是机智攻击，即攻击者假设 MapReduce 的 Job 调度策略中存在信任机制，在一段时间内它一直返回正确的结果，直到系统放松对其的监察，甚至完全信任攻击者，会无保留的接受攻击者提供的结果。此时，攻击者才会返回错误的结果。本文策略的安全目标包括：① 避免那些合法但互相冲突关系租户之间因为同平台运行而带来的潜在信息泄漏或篡改安全威胁（即第 1 节的第 1 类问题）；② 要在一定程度上避免或减少上述描述的几类安全威胁，包括恶意攻击者利用前面第 1 节提到的第 2 类和第 3 类安全问题形成非合谋方式下的鲁莽攻击、普通攻击和机智攻击。在给出 MapReduce 安全冗余调度策略之前，下面先给出该策略的一些基本假设。

1) 调度策略仍然遵循本地化优先原则，即优先选择本地保存了任务输入数据块副本的计算节点，否则按由近至远的方式选择距离任务输入数据副本

位置最近的计算节点，以优化任务的执行性能。

2) 系统可分配的计算资源总是足够的，即系统总是可以在原有调度策略基础之上，采用上述动态域划分模型，找到满足作业调度域、可信域划分属性且足够的计算节点。这样，在租户请求作业调度过程中，系统不会因找不到满足属性的节点而导致冗余调度失败。这种假设在具有大规模节点云环境下是可行的。

3) 系统的网络传输是安全的。尽管网络往往是分布式系统最薄弱的一环，入侵者多从网络入手，但此处的网络多特指系统内部节点之间进行数据传输的网络。网络是数据中心基础设施的重要部分，对于网络安全领域已有很多的研究，系统中可以通过多种手段保证网络安全（如加密），网络传输的安全可由计算框架之下的基础架构实现，并不属于本文讨论的范畴，因此做此假设是合理的。

4) 系统的中央节点是可信的。中央节点是系统的核心，负责整个系统任务的调度和分发，如果中央节点不可信，则基于中央节点的功能构建的整个系统就不可信，因此中央节点可信是必要条件。

4.2 基于动态域划分的安全冗余调度策略

为了实现上述安全目标，将基于上述动态域划分模型，并结合冗余调度方式，给出一种安全冗余调度策略，该策略的主要设计思想包括如下几方面。

首先，第 3 节的动态域划分模型说明：每个域划分策略实际上表示的是租户对其作业希望/不希望执行环境的基本安全需求（包括不期望的冲突关系以及期望的信任度、安全标签等信息，其中后者包括期望的安全级别和位置范围等），所以调度策略基于动态域划分模型，就可以将每一个待调度计算节点先进行冲突域、可信域和调度域的划分，即找出符合租户作业的基本安全需求的 3 类节点。

其次，为了防止 2 个冲突关系租户的计算任务同时分配给同一个计算节点可能带来的安全风险，如导致一方信息泄漏给另一方或被对方篡改，该策略将不允许一个任务分配给其冲突域节点。

再者，为了防止动态域划的调度域节点或者可信域节点仍然可能是恶意租户控制的节点或者存在恶意租户任务运行的节点（假设这 2 个计算节点同时是恶意节点且会同谋的概率很小），本文的策略将结合冗余方式，即对每个待调度计算任务产生 2 个副本，分别分配给任务对应的这 2 个域划分的计算节点。为了能够验证这 2 个计算环境中是否有一方是恶意的，将采用 2 种方法来一起验证。

1) 执行环境的完整性验证

首先，这里的执行环境是指计算节点上为任务的运行而提供的软件系统及其配置，如 Hadoop 中 JVM 目录的核心 JAR 包，租户提交 Task 的 JAR 包，HDFS 分布式缓存文件等。再者，对于执行环境的验证方法可以有 2 种：一种采用通用的软件计算方式，即通过软件实现的安全散列(MD5 或 SHA-1 等)函数来计算和保存二者执行环境的校验值，并进行简单的一致性对比验证；另一种采用安全性更强的硬件(如可信平台模块 TPM 或可信加密模块 TCM)支持方式，包括通过硬件来计算和保存二者执行环境的散列校验值，并通过硬件支持的远程证实[27]机制来进行一致性对比验证。这 2 种方法都使用了散列函数，在可信域或者调度域节点的执行环境被恶意控制和篡改的情况下（假设散列值已经通过软件或硬件方式做了加密存储和传输，不会被窃取或截获），它要试图伪造一个具有相同散列值的计算环境是困难的。

2) 随机选取小部分(1/a)原输入进行冗余计算和结果正确性验证

由于执行环境的完整性验证只能保证计算节点的静态运行环境是正确的，但并不能保证任务运行过程中出现的恶意行为（如恶意跳转等）。因此对输出结果进行验证将可以在一定程度上防止此类威胁。本策略将仅计算所分配任务原输入（如 Hadoop 每个任务的默认输入为一个数据块 64 MB）的 1/a 来完成验证，如为 Task 设置实际读取数据在原输入中的位置(如偏移量)和比例(如 1/a)。这样做有两点好处：一是可以减少完全冗余计算带来的计算资源开销大的问题，因为完全冗余会使系统中计算资源的利用率下降 50%，而针对 1/a 原输入的冗余计算，利用率仅下降 1/a(如 a=20 时，仅为 5%)，换句话说，若 1/a 足够小，则策略带来资源利用率的影响足够小；二是可以保证结果正确性验证的有效性，即系统能以足够大概率验证出计算结果是否被恶意篡改。例如，在大规模集群环境中，每一个作业 Job 通常会划分成大量的 Task 并行处理。假设 Job 被划分为 n(如 n=100)个 Task，每个 Task 只计算 1/a(如 a=20)原输入，则 Job 的输出结果被篡改但不被发现的概率为$(1-1/a)^n \approx 0.6\%$，即只要 n 足够大，则策略

验证出整个 Job 输出结果被篡改的概率也足够大。理论上，在指定的集群规模大小为 N（即最大可划分任务数）、资源利用率损耗要求不低于 U，和结果正确性验证失效率要求不高于 Q 的云环境中，a 的取值应该不小于 $1/U$ 和不大于 $1/(1-|N\sqrt{Q}|)$。

此外，由于上述 2 种验证结果能够体现调度域节点和可信域节点的可信性，因此本策略将要求在作业完成时对所有任务涉及的计算节点的信任度进行计算和更新。不选择每个任务完成时计算，主要是为了减小频繁更新给系统资源带来过大的开销。

经以上分析，下面具体给出本文安全冗余调度策略所包括的安全规则（如图 4 所示）。

安全规则 1 所有待调度计算节点都要求划分为与待调度计算任务关联的冲突域、可信域或调度域。

安全规则 2 一个租户作业 Job 的所有 Task 不允许被调度到属于其冲突域划分的计算节点 X 中运行。

安全规则 3 一个租户作业 Job 的所有 Task 都要求执行安全冗余调度(如图 4)，即 Job 的每个 Task 都要求产生 2 个副本(例如，图 4 Task A 的副本 1 和副本 2)，分别调度到 Job 关联的可信域节点 Y 和调度域节点 Z 中分别执行。

安全规则 3.1 Task 的第一个副本(简称副本 1)将被优先分配到 Job 的可信域节点 Y 中，并有如下要求。

1) 对副本 1 在节点 Y 中的当前执行环境计算 Hash 校验值 E_t，并提交中央节点。

2) 节点 Y 中的副本 1 仅需对随机选取的小部分(如 $1/a$)原输入执行冗余计算，对产生的输出结果计算 Hash 校验值 P_t，并提交中央节点。

安全规则 3.2 Task 的第 2 个副本(简称副本 2)将被分配到 Job 的调度域节点 Z 中，并有如下要求。

1) 在副本 2 执行之前，必须先对它在节点 Z 中的执行环境计算散列校验值 E_s，并提交中央节点，由中央节点完成 E_s 与可信域节点 Y 提交的 E_t 对比验证。

2) 若 E_s 和 E_t 相等，即判定节点 Z 可以满足执行环境的完整性要求（即执行环境没有被恶意篡改），则允许 Z 执行副本 2。一旦它对同样 $1/a$ 源输入数据产生了输出结果，则要求它计算该结果的散列校验值 O_s，并提交中央节点，由中央节点完成 O_s 与 O_t 对比验证。

3) 若 O_s 和 O_t 相等，即可判定节点 Z 可以满足结果正确性要求（即没有恶意租户的篡改或伪造），则判定 Task 已经成功分配给计算节点 Z，否则调度失败。此时可信域节点 Y 的副本 1 完成，Y 将供其他任务调度。

安全规则 4 一个租户作业 Job 的所有 Task 都已经成功调度时，对成功/失败分配了其中任何一个

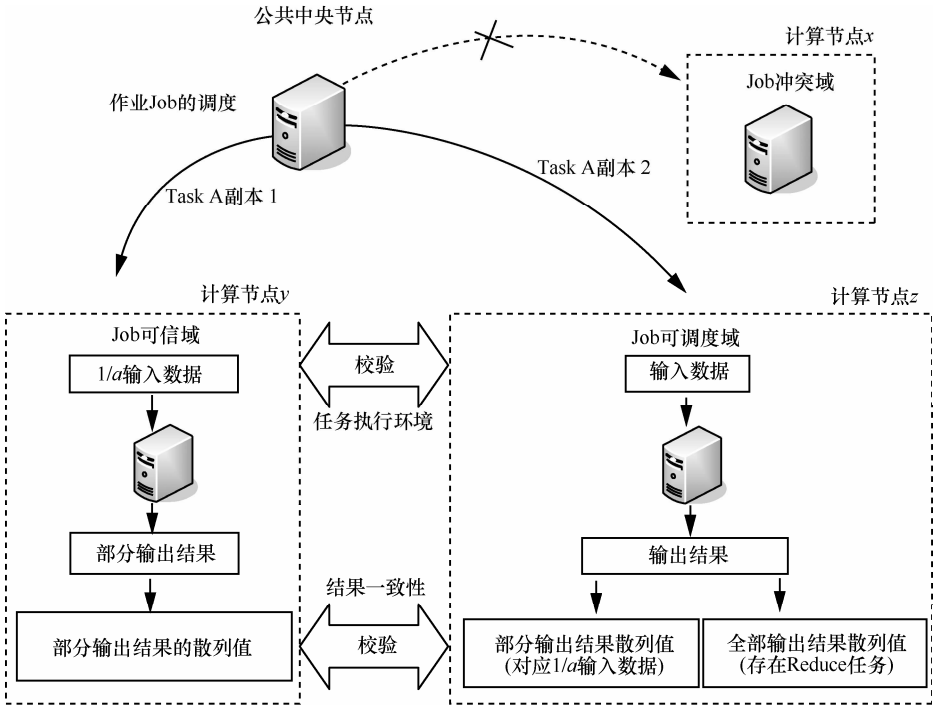

图 4 MapReduce 安全冗余调度策略示意

任务副本，而且属于 Job 调度域划分的计算节点，更新这些节点针对 Job 所属租户 U 的信任度。

5　实验和分析

5.1　基于 Hadoop MapReduce 原型实现

原型系统是基于 Hadoop 0.20.2 修改实现的，主要是在 Hadoop MapReduce 的 JobTracker 调度器和 TaskTracker 中增加了相关的安全机制，如图 5 所示。主要包括安全标签管理、节点信任度管理、冲突关系管理、域划分策略管理、散列值校验和安全冗余调度等 6 个新的安全模块。1）安全标签管理模块主要负责安全级别和位置属性的管理和维护，以及对计算节点进行安全标签的标记管理，并为域划分策略管理模块提供安全标签之间包含关系的计算功能等。2）节点信任度管理模块主要负责收集、验证和记录散列值校验模块提交的每个计算节点的 2 个可信属性情况，在一个作业完成时计算和更新对应的信任度。3）冲突关系管理模块主要负责系统中租户之间冲突关系集合的管理和维护，并提供给域划分策略管理模块使用。4）域划分策略管理模块主要负责对不同租户作业的域划分策略表达式进行管理和维护，以及根据安全标签管理、节点信任度管理和冲突关系管理等模块提供的待分配 TaskTracker 当前状态信息，计算出域划分策略表达式的值，并根据冲突域、可信域或调度域优先顺序决定 TaskTracker 的域划分。5）散列值校验模块主要负责对可信域节点和调度域节点的执行环境、部分结果的散列值进行计算和通过心跳机制提交给中央节点。6）安全冗余调度模块则主要是在租户提交作业 Job 的调度请求时，在现有调度策略基础之上，结合域划分判定策略管理模块的决策结果，从符合条件的调度域和可信域(但不选择冲突域)划分的 TaskTracker 中各选择一个，将 Job 的任务 Task 生成 2 个副本分配给他们，并通过节点信任度管理模块的 2 个可信性属性校验结果，决定该任务是否重新调度。

5.2　性能分析

Hadoop MapReduce 实验集群包括局域网中 4 个节点，其中在一台华硕 A8JR(1.5 GB 内存，1.8 GHz CPU) 机器上同时部署了 JobTracker(JT) 节点和 JobClient(JC)节点，它们是通过 VMWare Workstation 7.1 安装了 Ubuntu 8.1(512 MB)的 2 个虚拟机；在 3 台联想 M7100(3 GB 内存，3.5GHz CPU)机器上分别部署了 3 个节点：TaskTracker1 (TT1)、TaskTracker2 (TT2)、TaskTracker3 (TT3)，它们都是通过 VMWare Workstation 7.1 安装了 Ubuntu 8.1(1GB)的一个虚拟机。

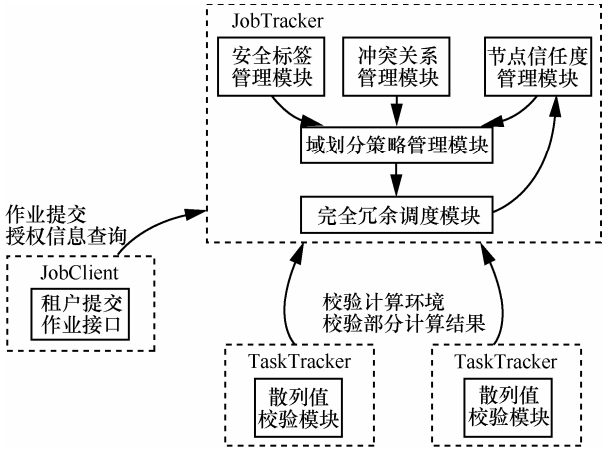

图 5　Hadoop MapReduce 安全冗余调度策略原型实现系统结构

5.2.1　策略对 Map/Reduce 任务调度性能的影响

在实验集群环境中，对计算节点进行动态的域划分，加上计算任务非本地直接读取输入的情况增加，都会带来时间开销；且每个作业的 Map/Reduce 任务都要求执行冗余计算，也会带来时间开销。如何测试这些开销呢？首先，测试 Map/Reduce 阶段的时间开销，以评估作业调度策略带来的时间开销，原因是 Hadoop 采取调度和处理并行的方式[5]。再者，以输入数据大小为基准来测试策略修改前后 Map/Reduce 阶段时间开销，原因是 Hadoop 对作业划分任务数与输入数据大小有关，如一个 Map 任务输入通常对应一个数据块(默认 64 MB)，在输入数据为 100 个数据块时，要调度的 Map 任务为 100 个。

图 6 给出了实验数据，其中：①时间开销测试数据是通过在系统中增加审计日志信息计算获得，并没有专门实现一种自动化的性能分析工具；②MapReduce 的日志时间可以精确到毫秒，但是由于实验集群数量有限，相对于工业环境使用的大规模集群，其总体运行性能更慢，所以测试得到的相关时间开销仅精确到秒。这些数据表明：策略修改后(a 选取 20)，Map 任务阶段调度时间开销平均增加了 20%~30%，如在策略修改前，待处理输入数据为 900 个数据块(每块 64 MB)时，Map 任务阶段时间开销为 320 s，在策略修改后，时间开销为 415 s，增加约 29%的开销；策略修改后，Reduce 任务阶段

时间开销增加比较小，基本在 10%以下，如在策略修改前，待处理输入数据为 100 个数据块时，Reduce 任务阶段的时间开销为 14 s，在策略修改后，时间开销为 15 s，仅增加 7%的开销。这是因为策略只要求 Reduce Task 对中间结果执行散列值校验，而散列值的计算本身不会带来过大的开销。但是随着输入数据的增加，Map 阶段产生的中间结果增多，Reduce 开销也略有增加。

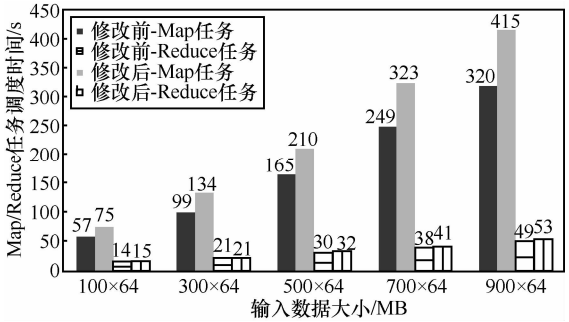

图 6 策略修改前后的 Map/Reduce 任务调度性能

可见，安全冗余调度策略会给 MapReduce 调度器带来一些时间开销。但是，如果可以选择合适的 MapReduce 集群规模，使可调度计算资源足够大，且每个作业的调度域节点和可信域节点总是能够获得情况下，可以适当减少由于计算资源不足而引起的动态域划分重复计算带来的开销。

5.2.2 策略对系统资源利用率的影响

现有 Hadoop MapReduce 将整个集群中的计算节点都看作是可以被调度的资源，但是在系统中采用了本文策略之后，整个集群中的计算节点都将动态划分为冲突域、调度域和可信域，而且只有调度域和可信域的计算资源对用户来说是可用的，这就可能造成可利用的计算资源减少的缺点。根据 4.2 节给出的 a 值选择方法，图 7 给出了策略修改前，策略修改后可保证资源利用率损耗不高于 5%的 2 个 a 值（20 和 50）条件下的 CPU 资源占用率测试数据。测试方法是，按顺序提交 10 个作业，在第一个作业提交之后直到所有作业全部完成的这段时间内，每隔 40 s 测试一次各个节点的 CPU 资源占用率，并计算其平均值(纵坐标)。实验数据表明：策略修改前，CPU 资源占用率为 23.5%，策略修改后，其占用率上升。例如：a 为 20 时，CPU 资源占用率为 27.9%；a 为 50 时，其占用率为 26.5%，但是可以发现，随着 a 值的增长，整个系统中资源占用率的上升比在减小，即策略带来的影响在减小。

此外，冲突域划分对资源利用率也有影响。如果冲突集合 C 足够小，即存在冲突关系的租户数量与系统总租户数量的比率足够小，则对系统中可用资源的影响将足够小。但是，如果出现冲突租户同时提交作业的概率很高的情况，则会导致系统可用的计算资源出现较明显减少的现象。

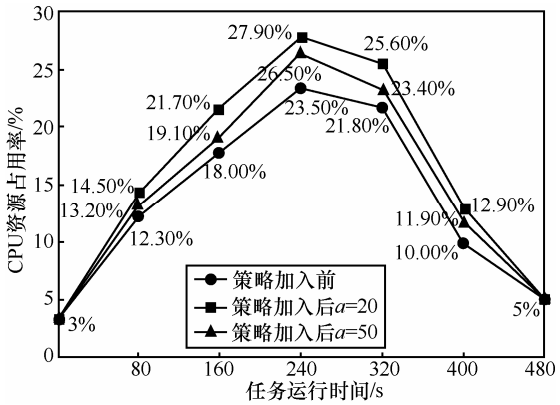

图 7 策略修改前后的 CPU 资源占用率

5.3 安全性分析

在实验集群环境中，模拟建立了 3 个租户(Alice 银行、Bob 银行、Clark 公司)，且设置(Alice 银行，Bob 银行)是冲突关系。如表 2 所示，为待调度节点 TT1~TT3 的当前状态，包括其当前运行的各租户的任务数、信任度、标签等，Alice 待提交作业 A 的域划分策略满足情况。实验的结果是：作业 A 的任务被分配到了其调度域节点 TT2 和可信域节点 TT3，但未被分配给其冲突域节点 TT1。这个结果与 3.3 节的动态域划分规则和 4.2 节的调度策略是一致的。

表 2 系统中各待调度节点的当前状态和待调度作业 A 的域划分策略满足情况

节点	租户名：任务数	租户名：信任度	安全级，位置集合	$P(\text{Alice},A)_S$=‘2’ && ($C2$, {中国})	$P(\text{Alice},A)_T$=‘3’&& ($C2$, {北京})	$P(\text{Alice},A)_C$=‘1’&& ($C1$, φ)
TT1	Bob:2; Clark:1	Alice: 1; Bob: 2; Clark: 3	$C1$, φ	×	×	√
TT2	Bob:1;	Alice: 2; Bob: 2; Clark: 3	$C2$,{华东}	√	×	×
TT3	Alice:1;Clark:1	Alice: 3; Bob: 2; Clark: 2	$B1$,{上海，北京}	√	√	×

注：“×”表示不满足，“√”表示满足。

可见，本文策略可以在一定程度上避免第 1 节提及的 3 类安全问题，具体分析如下。

1) 本文策略不允许将计算任务分配给其冲突域节点，使 2 个冲突关系租户的计算任务不会同时运行在同一个计算节点上(即防止出现第 1 节提到的第 1 类安全问题)，从而保证云环境中 2 个冲突关系租户计算任务之间的强安全隔离。

2) 本文策略在对计算任务进行调度时，是将任务的 2 个副本同时分配给可信域节点和调度域节点，并通过软件/硬件支持方式对二者执行环境散列值进行对比验证。这样做可以防止将任务分配给可能已经被恶意控制的可信域/调度域节点(即防止出现前面第 1 节提到的第 2 类安全问题)，从而保证云环境中合法租户计算任务与被恶意租户控制的节点上的任务之间安全隔离。

3) 本文策略在对计算任务进行调度时，要求可信域节点和调度域节点仅对随机选取的小部分(1/a)输入的计算结果的散列校验值进行对比验证，这样做可以防止 Task 分配到一个可能运行了可植入恶意代码的任务可信域/调度域节点(即防止出现前面第 1 节提到的第 3 类安全问题)，从而保证云环境中的合法租户计算任务与恶意的租户任务之间的安全隔离。

再者，本文策略可以在一定程度上避免第 2 类和第 3 类问题导致的非合谋方式下的攻击手段，具体分析如下。

① 对于鲁莽攻击手段，由于攻击者控制了计算节点，并总是(即概率 100%)返回错误的结果，而本文策略采取对每个任务都冗余调度的方式进行结果一致性验证，能够以极高的概率防范这种攻击手段。

② 对于普通攻击手段，由于攻击者仅以一定概率（小于 100%）返回错误的结果，但它所控制节点的执行环境通常会不同，本文策略对任务执行环境的完整性验证将使检测概率较高。但是，如果攻击者对任务的执行环境不做改变，并且仅以 x% 概率返回错误的结果，则本文策略由于只对 1/a 输入进行结果验证，无法检测出单个任务错误的概率会达到 1−x/a%（如 x=20，a=20 时为 99%），但对于大规模集群环境下整个作业的大量（n=100）任务来说，无法检测的概率仅为$(1-x/a\%)(1-1/a)^{n-1}$（约 0.617%，只比不正常情况下$(1-1/a)^n$高出约 0.025%）。即使攻击者控制了一个作业的 50%任务节点，无法检测的概率也只有$(1-1/a)^{n/2}(1-x/a\%)^{n/2}$(约 4.7%)。而且，一旦它返回错误结果被检测到，策略就会降低其信任度，使得其攻击目标租户的下一个作业无法分配到这个被控制的计算节点。

③ 对于机智攻击手段，即它在一段时间内一直返回正确的结果，甚至通过本文策略中的信任机制获取了大多租户对它较高的信任度(如将其划分为可信域)，之后才开始返回错误结果。但策略并不假设可信域是完全可信的，总会在调度过程中先验证二者计算结果是否一致(除非二者合谋)，一旦它认为将自己的信任度提高而开始发错误结果，就会被检测到，概率取决于它采取鲁莽/普通攻击手段。

6　结束语

本文针对目前 MapReduce 调度策略无法解决云计算环境中多租户计算任务的安全隔离问题，提出一种基于动态域划分的安全冗余调度策略。首先，它通过计算节点动态标记的信任度、安全标签等信息，以及给出与租户作业关联的 3 个域划分策略，对每一个作业的计算节点进行动态的冲突域、调度域和可信域的划分，保证冲突租户的计算任务安全隔离，即它们不允许被同时调度到同一个计算节点。其次，它通过部分冗余计算和验证，即基于可信域节点和调度域节点上相同任务副本执行环境的完整性及部分(1/a)输入数据的计算结果的一致性验证，使合法租户的计算任务在调度决策时与其不期望的恶意租户的计算任务和恶意租户控制的计算节点安全隔离。本文策略也适用于 Master/Slave 分布式处理架构的其他云计算系统。在未来工作中，将研究如何防止恶意节点按较小的概率来提交错误结果，甚至 2 个恶意节点合谋欺骗，导致验证结果误报等问题，并参考策略领域相关成果[28]研究如何有效地表达和运行策略。

参考文献：

[1] DEAN J, GHEMAWAT S. Mapreduce: simplified data processing on large clusters[A]. Proceedings of USENIX 6th Conference on Symposium on Operating Systems Design and Implementation (OSDI'04)[C]. Berkeley, CA, USA, USA: USENIX Press, 2004.137-150.

[2] Hadoop tutorial[EB/OL]. http://public.yahoo.com/gogate/ hadooptutorial/start-tutorial.html.

[3] Amazon elastic mapreduce[EB/OL]. http://docs.amazonwebservices.com/Elastic-MapReduce/latest/DeveloperGuide/index. html.

[4] MATHER T, KUMARASWAMY S, LATIF S. Cloud Security and Privacy: an Enterprise Perspective on Risks and Compliance[M]. USA, O'Reilly Media, 2009.

[5] Capacity scheduler[EB/OL]. http://hadoop.apache.org/common/docs/r0.20.0/capacity_scheduler.pdf.

[6] Fair scheduler[EB/OL]. http://hadoop.apache.org/common/docs/r0.20.2/fair_scheduler.html.

[7] ZAHARIA M, BORTHAKUR D, SARMA J S, *et al*. Job Scheduling for Multi-user Mapreduce Clusters[R]. EECS Department, University of California, Berkeley, USA, 2009.

[8] TIAN C, ZHOU H, HE Y, *et al*. A dynamic mapreduce scheduler for heterogeneous workloads[A]. Proceedings of 2009 8th International Conference on Grid and Cooperative Computing(GCC'09)[C]. Lanzhou, China, 2009.218-224.

[9] POLO D, CARRERA Y, BECERRA J, *et al*. Performance-driven Task co-scheduling for mapreduce environments[A]. Proceedings of 2010 IEEE/IFIP Network Operations and Management Symposium (NOMS)[C]. Osaka, Japan,2010.373-380.

[10] KC K, ANYANWU K. Scheduling hadoop Jobs to meet deadlines[A]. Proceedings of 2nd IEEE International Conference on Cloud Computing Technology and Science(CloudCom)[C]. Indianapolis, Indiana, USA, USA: IEEE CS Press, 2010.388-392.

[11] LI X, JIA Z, JU L, *et al*. Energy efficient scheduling and optimization for parallel Tasks on homogeneous clusters[J]. Chinese Journal of Computers, 2012:35(3):591-602.

[12] WEI W, DU J, YU T, *et al*. Securemr: a service integrity assurance framework for mapreduce[A]. Proceedings of the 2009 annual computer security applications conference(ACSAC'09)[C]. Washington, DC, USA, 2010.73-82.

[13] ROY I, SETTY S T V, KILZER A, *et al*. Airavat: security and privacy for mapreduce[A]. Proceedings of the 7th conference on networked systems design and implementation(NDSI'10)[C]. Berkeley, CA, USA, USA: USENIX Press, 2010.20-20.

[14] SONG S, KWOK Y, HWANG K. Security-driven heuristics and a fast genetic algorithm for trusted grid Job scheduling[A]. Proceedings of 19th IEEE international parallel and distributed processing symposium(IPDPS'05)[C]. Denver, Colorado USA, USA: IEEE CS Press, 2005.65-74.

[15] RUAN A, MARTIN A. TMR: towards a trusted mapreduce infrastructure[A]. Proceedings of the 2012 IEEE 8th World Congress on Services[C]. Hawaii, USA, 2012.141-148.

[16] HUANG C, ZHU S, WU D. Towards trusted services: result verification schemes for MapReduce[A]. Proceedings of 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)[C]. 2012.41-48.

[17] KHAN S M, HAMLEN K W. Computation certification as a service in the cloud[A]. Proceedings of 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)[C]. 2013. 434-441.

[18] BENDAHMANE A, ESSAAIDI M, EL MOUSSAOUI A, *et al*. Result verification mechanism for mapreduce computation integrity in cloud computing[A]. Proceedings of International Conference on Complex Systems (ICCS)[C]. Agadir, MAROCCO, IEEE CS, 2012.1-6.

[19] BREWER D, NASH M. The chinese wall security policy[A]. Proceedings of the Symposium on Security and Privacy[C]. Los Alamitos, Calif, 1989.215-228.

[20] WU R, AHN G J, HU H, *et al*. Information flow control in cloud computing[A]. Proceedings of IEEE 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)[C]. Chicago, Illinois, USA, 2012.1-7.

[21] SHEN Q, YANG X, YU X, *et al*. Towards data isolation and collaboration in storage cloud[A]. Proceedings of the 2011 IEEE asia-pacific services computing conference(APSCC2011)[C]. DJeju, Korea, 2011.139-146.

[22] KESARWANI A, GUPTA C, TRIPATHI M M, *et al*. Implementation of Chinese wall model in cloud computing for enhanced security[A]. Proceedings of IEEE International Conference on Emerging Trends in Networks and Computer Communications(ETNCC)[C]. 2011.411-413.

[23] CHEN Y, HUANG H, HUANG P, *et al*. A practical Chinese wall security model in cloud computing[A]. Proceedings of 2011 13th Asia-pacific Network Operations and Management Symposium (APNOMS)[C].2011.1-4.

[24] ALMENAREZ F, MARIN A, DIAZ D, *et al*. Developing a model for trust management in pervasive devices[A]. Proceedings of the 3rd IEEE Int'l Workshop on Pervasive Computing and Communication Security(PerSec 2006)[C]. Washington, USA, 2006.267-272.

[25] LI X, GUI X. Research on dynamic trust model for large scale distributed environment[J]. Journal of softwar, 2007, 18(6): 15510-1521

[26] ANDERSON R. Security Engineering: A Guide to Building Dependable Distributed System[M]. Indiana, USA: Wiley Publishing Inc., 2008.

[27] SAILER R, ZHANG X, JAEGER T, *et al*. Design and implementation of a TCG-based integrity measurement architecture[A]. Proceedings of the 13th USENIX Security Symposium[C]. San Diego, CA, USA,2004. 223-238.

[28] HAN W, LEI C. A survey on policy languages in network and security management[J]. Computer Networks(COMNET), 2012, 56(1):477-489

作者简介：

沈晴霓（1970-），女，江西宜春人，博士，北京大学副教授、硕士生导师，主要研究方向为云计算和大数据安全与隐私、操作系统与虚拟化安全、可信计算等。

卿斯汉（1939-），男，湖南隆回人，中国科学院软件研究所研究员，北京大学客座教授，主要研究方向为云安全、操作系统安全、应用密码学与安全协议等。

吴中海[通信作者]（1968-），男，浙江绍兴人，博士，北京大学教授、博士生导师，主要研究方向为情境感知服务、云安全与隐私保护、嵌入式智能、大数据与信息融合等。E-mail: wuzh@pku.edu.cn。

张力哲（1988-），男，黑龙江双鸭山人，硕士，主要研究方向为分布式系统、云存储系统安全等。

杨雅辉（1966-），女，河北唐山人，博士，北京大学教授，主要研究方向为云计算和大数据平台及安全、网络安全管理、网络虚拟化技术及应用、网络性能评价与优化等。