关于NoSQL数据库上的虚拟专用数据存储

摘要：许多现代应用使用上下文相关信息来提供高度个性化服务，基于NoSQL数据库的高性能和支持大容量存储数据，使用NoSQL数据库进行数据管理。 但是，NoSQL数据库集成了最基本的粗粒度访问控制使其数据保护特性较差，而且它不支持上下文感知策略。因此，我们相信需要一个用于增强NoSQL数据库的细粒度上下文感知访问控制的通用方法。在这篇文章中，我们开始填补这一空白，将目标锁定在一个非常受欢迎的数据库：MongDB.这篇文章的贡献是双重的。我们增强了带有高级特性的MongDB的访问控制模型并且我们为提出的增强模型定义了一个执行监控器，它能直接在任何MongDB中部署。由于技术的限制，MongDB不能够在所有的查询类型中实现相同的执行机制。因此，实验结果先是聚合查询的执行开销是值得注意的，find查询和map\_reduce查询都比它具有更低的开销。

I 介绍

上下文感知（Context awareness）正变成现代应用程序的一个重要方面，它为用户、服务提供商和供应商提供了许多潜在的好处。通常，上下文感知应用程序提供高度个性化有利于更好的用户体验。相比于更加传统的解决方案，用户更喜欢这种方式。进而增加了它们的快速扩散。例如，许多电子设备和运动服装连锁店现在提供上下文感知的电子商店，地理空间数据被用来为潜在的客户提供有用的信息，例如客户当前位置的店里能够获得的商品。

尽管开发上下文感知服务的最直接方法是在应用程序层面上编码嵌入上下文感知，但是这种方法存在通用性、可移植性和效率问题。事实上，更加通用的解决方案的需求使得许多数据库服务提供商开始在他们的产品中集成上下文感知功能。这种趋势已经影响了传统的关系型数据库管理系统（RDBMSs），同样也影响了NoSQL数据库（e.g MongDB支持地理空间查询）。除了定制服务之外，通过集成基于上下文的条件集成到访问控制协议中，上下文相关信息也被用户获得更好级别的数据保护。而且结合细粒度访问控制会更有效率，细粒度访问控制允许调节进入到数据模型的原子组件（e.g. 元组，文档，键值对根据所考虑的数据管理系统），结合允许调节访问数据值的基础的基于内容的策略。这种集成解决方式在RDBMSs上已经成功提出。一个最好的例子是Oracle虚拟专用数据库（Oracle VPD），最初的在Orancle DBMS上整合细粒度基于上下文和基于内容的访问控制的商业框架。相反，据我们所知，目前为止，在NoSQL数据库上还没有类似的提议存在。因此在这篇文章中，我们力图提出一个框架将NoSQL数据库转变为虚拟专用数据库来填补这一空白。我们相信这一挑战目标具有战略重要性，因为NoSQL数据库在上下文感知的后端经常被使用到。到目前为止，绝大多数的NoSQL数据库集成了非常基本的粗粒度访问控制模型。只有很少的最近定义的数据库，如Accumulo提出了支持FGAC，但是它仍然不支持上下文感知和基于内容的策略。

为虚拟专用NoSQL数据库定义一个通用的访问控制框架的目标是很宏伟的。一方面，在RBDMSs中开发的执行机制不能够直接应用到NoSQL数据库中，因为NoSQL数据库具有远比RDBMSs复杂的场景。NoSQL使用的多样的数据模型和查询语言，而且这些系统里存储的数据是无模式的，使得在RDBMSs里的方法在NoSQL里不适用。另一方面，这些高级访问控制特性的开销无法和NoSLQ数据库的效率妥协。最小化开销的需求使得需要对每一种数据库设计执行机制。因此，我们相信，这样一个通用框架只能通过逐步的方法实现，第一步先在单一的NoSQL平台上实现，之后再将这种开发机制用到其他的数据库中。基于这些考虑，在这篇文章中，我们开始在MongDB，目前的排名调查中最受欢迎的NoSQL数据库中开发。

更准确地说，这篇文章中，我们提出一个方法增强MongDB，带有更高级的访问控制形式。它允许执行类似于Oracle VPD的基于内容和基于上下文的策略，甚至在更细粒度的级别，即文档域（document field）级别。与其从头开始开发一个访问控制模型，我们在MongDB本地的基于角色的访问控制模型的基础上增加细粒度的基于角色上下文和内容的访问策略实现增强模型。增强模型已经被一个称为ConfinedMem(CONtext aware FIne graiNED MongoDB Enforcement Monitor)执行监控器实现，它已经在所有的MongDB的客户端和服务器版本中明确定义。从技术层面上说，ConfinedMem最主要创新的地方在于在通信协议层面上支持执行机制操作以及基于代理的监视器体系结构，确保了在驱动程序和编程语言上的独立性。它使得我们可以直接在现有的部署中通过基本的配置操作整合ConfinedMem而不需要任何编程活动。策略执行分为两个步骤：1) 资源重写活动，获取查询访问文档的授权视图。2) 查询重写活动，重写查询使得只能查询访问授权的视图。为了最小化执行开销，ConfinedMem对于不同的查询类型采用不同的实现方式和相关活动。

执行机制的工程代表了目前工作中最具挑战性的研究方面。目前MongoDB技术的局限性有一个预期的后果，就是它不允许重写聚合查询直接实现上述的资源修改任务。

当前的作品借鉴了我们之前的一篇文章的想法，这篇文章中我们提出带有基于目的的访问控制的增强版的MongDB.然而，本篇文章通过提供到文档域级别的细粒度访问控制和允许在访问控制过程中使用一般的上下文信息显著扩展了之前那篇文章。无论是在支持的访问控制模型的表现力上还是在执行机制的复杂性上都是一个显著的扩展。实际上，在之前的文章中提供一种在文档级别上的单一目的合规性谓词，允许通过重写查询机制增强策略。

这篇文章的剩下部分组织如下。第II部分介绍MongDB的背景知识和值文章其余部分使用的一个运行的例子。第III 部分描述了增强的访问控制模型。第IV部分介绍ConfinedMem,第V部分介绍实验结果。第VI部分介绍相关的工作。最后，第VII部分总结全文。

II 背景和运行的例子

MongDB是一个客户端-服务器数据管理应用，客户端部分向服务器前端提出查询请求，服务器前端处理策划若干个服务器节点之间的查询执行。客户端-服务器的交互是通过MongDB线协议实现的。线消息允许对任何类型的客户端请求和服务器响应进行编码。参与confinedmem执法活动的唯一信息的类型如下：1) OP\_QUERY, 携带查询和管理命令；2) OP\_REPLY ，编码服务器响应。

MongoDB采用面向文档的数据模型，文档是一系列的键值对（<key,value>）, 么个键值对表示文档的属性。在这篇文章的剩余部分，文档的属性将被成为域（field）。MongDB允许定义原始类型、数组和对象为域。可能具有异构格式（不同字段集组成）的文档被分组为集合（collection），最后，一组集合构成一个数据库.

为了说明上述概念我们在此介绍一种应用场景，它将作为整篇文章的运行实例。Enron语料库是一个email信息数据集，里面包括了Enron公司大约150名员工之间交流的50万（大约1.5GB的数据）电子邮件。我们使用MongDB来转储Enron数据集。它的组成是：一个数据库名为enron,它由存着用户之间交换的邮件称为message的集组成。图1展示了一个电子邮件文档的例子。如图1所示，一个文档有邮件内容、邮件标题、可能的附件和电子邮件所属的邮箱这些域组成。它还包含一个代表电子邮件头的字段，即一个对象，它和其他信息一起指定每个消息的发送者和接收者。

MongoDB支持一组丰富的查询类型，从简单的选择，对单个集合的聚合形式复杂。在这项工作中，我们主要关注find查询、聚合查询和MapReduce查询这三种我们认为最有代表性的查询类型。接下来，我们简单介绍我们所考虑的查询以及我们稍后将使用用来说明ConfinedMem的语法的细节和抽象的执法机制的一些简单的符号。

Find查询是MongoDB中最基本的分析操作，它允许进行文本的筛选。

定义1.Find查询：一个find查询fnq是一个<cl,sc>键值对，cl表示目标集，sc表示对cl中的文档的选择标准。

相反，聚合查询和Map-reduce查询允许执行复杂的聚合。聚合查询通过数据处理管道执行聚合，其中目标集合的选定文档通过增量将它们转化为聚合结果的操作序列。

定义2.聚合查询： 一个聚合查询agq是由三元组<cl,sc,ap>组成，其中cl代表目标集合，sc表示对cl中的文档的选择标准，ap是聚合管道，也就是允许处理cl中的文档的一系列操作。

最后，map-reduce查询通过map-reduce范式实现聚集。map阶段应用于每个选定的文档，并返回一个键值对。对于那些有多个值的键，reduce阶段中得出聚合数据。

定义3.Map-reduce查询：一个map-reduce查询mrq是由元组<cl,map,reduce,sc,clout>,cl表示目标集，map和reduce是函数，分别表示在map-reduce工作中map阶段和reduce阶段的操作，sc标识对cl中的文档的选择标准，clout标识mrq的结果集中可选的集合。

例1.参照我们的运行实例，find查询q1检索所有在时间间隔[t1,t2]内发送的邮件可以被建模为<messages, ”headers.Date” :{$lt : t2, $gt : t1}>,为了简单起见，选择标准使用MongDB的语法。聚合查询q2计算被发送到接收者rl的接收列表中的所有电子邮件，可以被表示为<messages,{”headers.To”:{”$in”:rl}},[{$group:{” id”:”sum”, ”num”:{$sum: 1}}}]>,管道使用MongDB的语法表示，执行满足选择标准的文档的计数。最后，map-reduce查询q3执行q2相同的任务可以表示为：<messages, {function(){var toF=this[”headers”][”To”];for(i

in toF){emit(toF[i], 1);}}, function(key, values){return Array.sum(values);}, {”headers.To”: f”$in”:rl}}, Φ>.map阶段生成对于每一个电子邮件的每一个接收者键值对<receiver,l> 。而reduce阶段计数所有由一个给定的帐户接收的消息。

III 访问控制策略

我们的模型在MongDB RBAC的基础上增加在集级别、文档级别和域级别基于内容和上下文的策略。这个策略根据以下策略和MongDB的RBAC结合（政策的组成和相关的执法机制的更多细节将在第Ⅳ-C提出）。假设cl是目标集，d是cl中的文档，然后一个用户u提交了一个在cl上的查询q。工作在集级别的MongDB的RBAC策略，首先进行评估。如果访问被授权，集级别的基于上下文的策略开始进行评估（如果存在的话）。这样的策略在上下文信息的基础上表达了对任何在cl上的所有文档访问的前提条件。因此，q查询只有在没有在cl上的基于上下文的策略或者上下文策略cxpc已经被制定，并且由u发出的q满足cxpc的上下文的情况下才会被执行。如果这些检查都成功了，在cl上的基于内容的策略开始评估，它可能会限制u提出的对cl中的文档的访问。在接下去的域/文档级别基于上下文的评估下，结果集会进一步别削减。与前面的情况相似，如果没有在文档d中域f上的策略，q可以访问f.相反，如果有在f上的基于上下文的访问策略cxpf，只有满足了该访问策略，才能访问f。

接下来，我们正式介绍ConfinedMem支持的所有类型的策略。为了介绍基于上下文的访问策略，我们应该首先定义表征其定义的基本概念元素，即上下文类型、上下文对象、目标属性类型、目标属性对象和基于上下文的谓词。

与Oracle VPD相似，我们不强制使用预定义的上下文模型，而是允许系统管理员定义针对他们的目标应用场景的要求和特性的上下文类型。上下文类型cxt指定用于描述应用程序场景的上下文的结构属性。有用户发出的每一个查询执行的请求被当作一个上下文对象co,co是上下文类型cxt的一个实例，它指定在查询调用时间中与用户和/或执行环境有关的信息，然后用于决定访问控制请求的结果。为了与MongDB一致，我们用JSON对象来表示co对象。因此，cxt被指定为一个约束co对象的特征的JSON模式。

例2. 让我们定义上下文类型CXT建模以下上下文信息：已发出请求的查询执行请求、时间和位置的标识符，用于发送请求的设备的地址。JSON的方案建模这样的上下文类型的一部分如表1所示。每次用户发送一个查询执行的请求，一个上下文对象co,cxt的一个实例，生成并于请求相关联。例如，co可以被定义成：{“id”:“0001”, “time”:“2015-04-17T09:38:12.152Z”, “location”:“15”, “mac”: “12-34-56-78-9A-BC”}

基于上下文的谓词指定用于授予访问必须满足的条件。

定义4 基于上下文的谓词：上下文谓词cbp(co:txt,tp:tpt)是由模式cp-op-tp的组合所构成的参数布尔表达式。cp(上下文属性)是一个类型cxt的上下文对象co的属性,tp(目标属性)是是相比于cp的目标上下文属性，而op(运算符)是一个布尔运算符/函数对比tp和cp.目标属性类型tpt定义目标属性tp的结构特征。

例3 参照例2,上下文谓词cbp可以指定如下要求： 1）发起查询的设备的MAC地址要属于授权地址的列表（e.g 12-34-56-78-9A-BC和34-56-78-9A-BC-12），2）查询发起在指定时间段（e.g 8:00-18:00）.根据定义4，谓词可以被定义为：cbp(c:cxt,t:tpt) = (c.mac 2 t.macList(t.start ≤c.time ≤ t.end).目标属性类型tpt建模一个数组maclist为授权MAC地址的列表，两个属性代表开始和结束的时间间隔。

基于上下文的策略将选定的基于上下文的谓词绑定到数据资源。

定义5（基于上下文的策略）：基于上下文的策略cxp是一个元组<crs,cbp,tpo>,其中1）crs表示约束的资源，它可以是文档的集，单个文档或者是文档的域。2）cbp表示基于上下文的谓词，tpo指定cbp会进行评估的目标属性对象。

给定一个基于上下文策略cxp保护资源rs的访问 (i.e., rs 2 cxp.crs)，访问rs要求的条件是基于上下文的谓词cbp满足cxp.cbp:1)上下文对象co,这无疑是与查询请求相关的，2）目标属性对象tpo在cxp.tpo中。换句话说，当*cbp(co,tpo)=true* 时条件满足。

例4 让我们思考一个策略cxp,使得在办公室的工作时间由行政部门中任何设置发出访问enron邮件的请求允许访问.这样的话，crs，cxp的约束资源的组成部分，将enron.messages作为唯一的受保护的资源。我们假定基于上下文的谓词，目标属性类型tpt和上下文类型txt是例3中的。

目标属性对象tpo被cxp指定为：

1）该部门中所有注册设备的MAC地址列表，

2）授权访问的时间段。

另一种支持的策略是基于内容的策略，允许定义每个用户/角色授权的视图

目标文档集。授权视图定义在文档内容上指定的选择标准。

定义6（基于内容的策略）：基于内容的策略cnp是一个元组<sid,cln,cnc>.其中sid表示主体的标识符，即策略指定的用户/角色的标识符，cln指保护的数据库集的名字，cnc指定对cln中的文档的约束。

cnc定义一个推导可访问cln文档的筛选视图的选择标准，他可以是MongDB查询的任何有效的表达式（e.g.,$eq,$gt,$or）以及cln文档域的标识符。

例5 让我们假设我们想指定一个基于内容的策略，允许角色analyst访问由u1和u2发送的enron邮件.于是sid就被设置为角色analyst,cln就是enron.messages,cnc被定义为{$or:[“headers.from”:“U1”, “headers.from”:“U2”]}.

IV ConfinedMem

这个部分，在这一部分中，我们讨论在第三节中增强访问控制模型的执行监控器confinedmem的结构。

A.架构层

confinedmem被设计为允许上下文感知的细粒度的访问控制直接集成到任何MongoDB部署中。它被定义为一个专门的MongoDB线协议解释器，因此它可以在MongoDB的驱动程序以及与MongoDB平台的不同版本中运行。它作为一个代理截获，分析，并有可能改写MongoDB客户端和服务器交换的线消息，他们不返回数据未经授权的数据。

出于演示目的，我们考虑图2的情景：一个MongDB部署由n个mgb客户端mc1,mc2...mcn和一个mgb服务器ms组成。Cdm位于客户端和服务器之间，每一个客户端都定义了一个上下文感知模块csi,csi的作用有：1）检测当用户请求发出时，用户调用执行查询时的上下文；2）实例化一个代表在查询执行时间的上下文对象 3）发送这个对象到cdm.

cdm是一个多线程应用：1）持续等待某个潜在客户MCI新的连接请求，2）

激活客户端mci与服务器的通信信道ci，在信道中相关的客户端-服务器交互的消息被截获和转发3）激活线程mti用于分析和处理ci的信息。

监控线程mti分析所有来自mci,csi和ms的线消息。一些消息只是简单地被mti转发，而指定的消息如客户端请求，上下文对象和服务响应（被编码成OP\_QUERY 和OP\_REPLY消息），触发特定的执行活动，比如防止查询的执行，mti会重写收到的消息，发送消息到服务器来查询有关用户相关信息。

B.策略绑定

我们提供支持三种类型的访问控制策略，即基于角色的，在不同的粒度级别上的基于上下文的和基于内容的政策。因此，不同需要为他们制定不同的绑定标准。

对于基于角色的策略，我们使用MongoDB的RBAC指定的绑定标准。绑定的方法是通过指定的mgb系统集来记录记录在数据库中的角色和和它的权限。绑定有以下定义组成:1)权限由键值对<cl,ac>表示，其中cl表示约束的数据库/数据库集，ac表示授予的权限（e.g. 读，写），2)权限的集合就组成了角色。

基于内容的策略绑定我们也采用同样的策略。系统集cnpS用于保存授权给给定的角色/用户的数据集中导出的基于内容的视图。这些集合由文档组成，每个文档代表一个基于内容的策略的三元组。

相反，基于上下文策略允许调整在不同细粒度下的访问。因此，不同的策略粒度需要采用不同的绑定标准。与基于角色和内容的策略相似，一个系统集cxpS,用于记录数据库集级别的基于上下文的策略。而系统集cxtS和tptS分别用于记录上下文类型和目标属性类型，收集在cxpS上的文档有一个有定义5定义的结构标准。相反，使用一个系统集来记录在文档或域级的策略似乎不合适，因为这样将需要记录每个对文档/字段的访问以获得相应的政策。因此，为了更有效的访问指定的策略，文件和域级的基于上下文的策略指定一个专门的cxps，添加到每个受保护文档的结构中。指定策略的cxps域是JSON对象对应有定义5中定义的cxp。当查询q处理文档d时，在d中的域级/文档级别的策略可以很容易推导出能访问的内容。

C执行机制

现在我们讨论cdm如果执行访问控制。考虑一个目标集cl,在cl中的文档d，假设一个用户u，提交查询q访问cl中的文档。co是q被调用是的上下文，作为上下文类型cxt的一个实例。定义Rs为被考虑的应用场景的角色的集合