doi:10.11959/j.issn.1000-436x.2015191

面向内容发布订阅系统的向量订阅与共享机制

尤涛，吴其蔓，王川文，钟冬，杜承烈
（西北工业大学 计算机学院，陕西 西安 710129）

摘 要：在分析典型索引结构和树型结构匹配算法变更特性、匹配效率的基础上，以匹配效率高的匹配树算法为基础，扩展其订阅结构为向量结构，提出了向量变更算法。基于向量间的关联关系，提出了向量共享算法，从而有效降低订阅变更对匹配树重构的影响，提高了订阅处理效率。理论分析和实验表明，与已有订阅变更方式相比，该方法能够在满足频繁订阅变更的同时提供高效的事件匹配，满足相关应用的要求。
关键词：内容发布订阅系统；属性；约束；向量
中图分类号：TP393 **文献标识码**：A

Vector subscriptions and sharing mechanism for content-based publish/subscribe system

YOU Tao, WU Qi-man, WANG Chuan-wen, ZHONG Dong, DU Cheng-lie
(School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China)

Abstract: The concept of vector subscriptions to support subscription adaptations was introduced. Novel algorithms were proposed for updating vector subscriptions in classic matching tree structures, and was presented sharing algorithms for vector subscriptions. These algorithms effectively reduce the impact of subscription changes in matching tree structures. Compared to re-subscriptions method, the algorithms significantly reduce the reaction time to subscription updates and can sustain higher throughput in the presence of high update rates.
Key words: content-based publish/subscribe; attribute; constraint; vector

1 引言

Internet 技术的广泛应用和移动计算、网格计算以及普适计算平台的快速发展，需要分布式系统能够满足大规模、分散控制和动态改变的特点。这就要求系统的各参与者之间，采用一种具有动态性和松散耦合特性的灵活通信范型和交互机制。发布/订阅（pub/sub）通信范型与传统的通信范型（消息传递、RPC/RMI 和共享空间）相比，具有异步、多点通信的特点，使通信的参与者在空间、时间和控制流上完全解耦，能够很好地满足大型分布式系统松散通信的需求。目前，基于内容的发布/订阅系统（CPS, content-based publish/subscribe systems）得到了广泛的应用[1]。这类系统中的事件不再依赖于外部的某个标准（如通道、主题等）分类，而是按照事件的内容本身分类。订阅者根据事件的内容来订阅事件，不必受系统预先定义主题的限制。订阅者可以自由地表达订阅信息，使系统更加灵活。

在已有的相关研究中，核心问题都是解决事件代理采用何种算法实现事件和大量订阅者之间的高效匹配。然而，伴随着系统应用领域和范围的不断扩大，诞生了对内容发布/订阅的新需求。例如，当前比较流行的高频算法交易（HFT, high frequency

收稿日期：2014-10-27；**修回日期**：2015-02-13
基金项目：航空科学基金资助项目（20135553034）；中央高校基本科研业务费专项基金资助项目（3102014JSJ0008）；2014 年西北工业大学本科毕业设计（论文）重点扶持基金资助项目
Foundation Items: Aviation Science Foundation of China(20135553034); The Fundamental Research Funds for the Central Universities(3102014JSJ0008); Undergraduate Graduation Design Key Project for 2014 Northwestern Polytechnical University

algorithmic trading）[2]，其交易量在美国甚至占到了总额的 73%。算法交易与已有的应用不同，在交易日算法交易往往应用了不同的线性回归、博弈论、神经网络、遗传算法等，对交易过程的订阅细节进行干预，这就带来了对订阅的大量动态变更。因此交易系统中就要求在满足高效匹配的前提下能够应对用户频繁更改查询（订阅）条件的需求，如果匹配算法不能同时支持频繁更改查询和高效匹配，将给客户带来不可估量的损失。又如移动网络环境下的位置感知类应用[3]，如基于位置的社会网络 loopt[4]等。这类移动发布/订阅系统中，当设备位置发生变更，该设备就需要依据新的位置关系更新订阅；设备的频繁移动，带来了大量的订阅变更请求。再如分布式虚拟环境[5]（DVE, distributed virtual environment）中典型的大规模战场环境，参与的各类军事实验实体众多，且关系复杂，随着实验进程的推进，实体间关系面临频繁变更。事件匹配与订阅变更的效率，不仅决定了系统的实时性是否能得到满足，而且会影响实验的真实性。

鉴于新应用对动态频繁订阅的新要求，本文的主要贡献如下。

1) 分析了索引结构中订阅易变更的原因和树结构中订阅不易变更的原因，并据此得到了通过降低订阅变更过程对匹配结构的影响来提高订阅变更效率的思路。

2) 针对具有高效匹配特点的树结构提出了向量订阅和共享的方法，在保障匹配树匹配效率的情况下有效提升了订阅变更的效率。

3) 对向量订阅和共享算法进行了实验测试，结果表明向量订阅和共享机制在提高订阅变更效率、事件匹配过滤吞吐量方面具有良好的效果，较传统方法有 3 倍左右的提升。

2 相关工作

目前，主流内容事件匹配算法主要基于：索引结构和树结构。计数法(counting algorithm)[6]是一种典型的基于索引结构的算法，计数法的思想是测试所有谓词建立一个表保存谓词和订阅的映射关系，即谓词被哪些订阅满足。匹配时，遍历这个表找出满足一个订阅的谓词数目，将之与这个订阅包含的谓词数目进行比较，如果相等则说明事件与订阅相匹配。计数法虽然避免了一个谓词被多次测试，但是它总会对订阅中所有的谓词进行测试，而实际上当一个订阅中的某谓词无法匹配时，其他后续测试都不需要继续进行。典型的还有汉森算法（Hanson algorithm）[7]、CEEM[8]算法及其扩展算法等。

树结构算法的一种典型代表是基于匹配树[9]的算法，在该算法中订阅被组织成为一棵树。树中的每一个非叶子节点表示对一个谓词的测试，出边标识了该测试的结果，订阅存放在叶子节点中。如果事件从树根节点沿出边一直可以到达某一个叶节点，那么存放在该叶子节点上的订阅就被满足了。Gryphon 采用了基于树的匹配方法，并且添加了一条特殊的出边，称作不关心边，通过这种边可以到达的订阅，表明该订阅不关心在该节点上的谓词测试结果。Gryphon 声称在订阅中的谓词都是相等测试时，匹配事件的时间复杂度与订阅数目呈线性关系。树结构的另一个代表是 SIENA[10]，SIENA 根据订阅之间的覆盖关系将其组织成为偏序集（Poset, partially ordered set）。典型的其他算法还包括二叉决策图（BDD, binary decisions diagrams）的算法、基于分布式 R-trees 的匹配路由算法[11]等。

通过对 2 类匹配结构的分析，可以得到如表 1 所示的两者易变更性与匹配效率的对比。索引结构中订阅易变更的原因主要有：1)订阅之间的无关联性，索引结构中订阅之间没有关联关系，单独订阅的改变不会影响其他的订阅，从而可以单独改变需要变更的订阅而无需考虑其他订阅；2)订阅条件内属性分离，索引结构中将订阅属性分离，并以散列的方式存放，可以迅速查找到需要改变的属性，从而循序变更订阅条件。但这 2 个特点也使其匹配效率低。

表 1 订阅结构与订阅变更的关系

匹配结构	订阅之间关联性	订阅内属性关联性	匹配效率	是否易变更
索引	无	弱	低	是
树型	有	强	高	否

树结构不易变更的原因是树结构中订阅强关联的特点：1)订阅之间的关联性，树结构中的订阅之间具有偏序的关系，一旦订阅内容变更，就有可能破坏这种关系，从而引起树结构的大规模变更；2)订阅内部属性的关联性，树结构中需要维护整体的偏序关系，属性不具有分离的特性，因此，一旦需要变更订阅某个属性，就需要在该订阅中线性查找该属性，这与属性分离的散列查找相比，效率会

低很多。而这 2 个特点也使树结构算法能够很好地利用订阅间的关系加速事件匹配。

从表 1 的比较不难发现，与索引结构相反，树型结构在订阅关联性、属性相关性方面的强特性虽然使其匹配效率较高，但却造成了订阅变更的低效。那么能否通过订阅变更机制的改变来减少变更对树结构中匹配树重构的影响，从而达到迅速变更的同时高效匹配的要求，这正是本文研究的主要动机。

现有的内容发布/订阅系统中，订阅变更机制采用通配符[12]和二次订阅（unsub/sub）的方式[13,14]。通配符的方式通过将内容订阅退化为通配符匹配的过程，订阅过程中很小的订阅条件变更都转换为整个订阅系统的重构，因而效率较低。二次订阅方式通过取消先前的订阅条件并再次订阅的方式完成。二次订阅方式不需要每次都变更整个订阅系统，代价较通配符方式低，因而成为当前各类系统主要的订阅变更方式。但二次订阅方式也存在许多不足：系统耗费很大吞吐量在处理订阅条件的变更，当变更频繁时很有可能使系统中大多代理者都把时间花在变更订阅条件而不是事件过滤上；这样的代价对树结构而言更是巨大，二次订阅要经过取消和再订 2 个过程，对树型结构而言每个订阅的变更都带来匹配树的 2 次调整，而实际过程中并不是所有的调整都是必需的[6]。可见，即便是二次订阅方式也不能很好地适用于匹配树结构。因此，将在匹配高效的匹配树结构基础上，研究一种新的订阅变更机制，使系统在满足频繁订阅变更的同时提供高效的事件匹配。

3　典型订阅机制

3.1　定义

定义 1　事件。事件 E 作为内容发布/订阅系统中用于系统内实体交互的单元，包含了属性 a 的序列$[a_1, a_2,\cdots,a_n]$，这些属性都是基本数据类型。而事件实例 e 可被视作属性值 u 的一个记录$[u_1,u_2,\cdots,u_n]$。

定义 2　谓词，订阅。基于典型的条件变量 $c ::= \leqslant|<|=|>|\geqslant|\neq$，谓词可以描述为 $P ::= a\ c\ u$ 的形式，即关于事件属性的一系列限制。由基本的谓词构成不同的订阅 $\Phi ::= \Phi \wedge P \mid \Phi \vee P \mid P$。

定义 3　订阅的包含关系。对于订阅 Φ_1、订阅 Φ_2 和任意事件 e，若事件 e 匹配订阅 Φ_1，有 e 也一定匹配订阅 Φ_2，则称订阅 Φ_1 包含于 Φ_2 或订阅 Φ_2 包含 Φ_1，表示为 $\Phi_1 \preceq \Phi_2$。

定义 4　订阅的相等关系。对于订阅 Φ_1、订阅 Φ_2，若订阅 Φ_1 包含 Φ_2 且订阅 Φ_2 包含 Φ_1，则称订阅 Φ_1 等于 Φ_2，表示为 $\Phi_1 = \Phi_2$。

3.2　常量订阅

定义 5　常量订阅[13,14]。常量订阅中，订阅结构 $P ::= a\ c\ u$ 的条件 c 和属性值 u 均为常数。常量订阅也是最传统的订阅方式。由于订阅信息均为常量，因此订阅的变更需要以二次订阅的方式进行。

图 1 为一个典型的常量订阅变更的例子。订阅者 s_1 和 s_2 通过代理 b_1、b_2 完成订阅。首先，s_1 通过“*price*<10”订阅；b_1 获得订阅，将它存储并传递给 b_2。然后 s_2 通过“*price*<5”订阅；b_1 获得订阅，但是不能将它转发给 b_2，因为“*price*<10”包含“*price*<5”。此时，s_1 要将其订阅内容变更为“*price*<20”，按照常量订阅的定义，只能采取二次订阅的方式进行变更。处理流程如下：s_1 将“*price*<10”退订，b_1 将“*price*<5”转发给 b_2；然后，当 s_1 订阅 “*price*<20”，b_1 重新构造了偏序集。当改变到“*price*<20”以后，b_1、b_2 将“*price*<5”退订，然后订阅“*price*<20”。订阅变更的处理是非常复杂的。

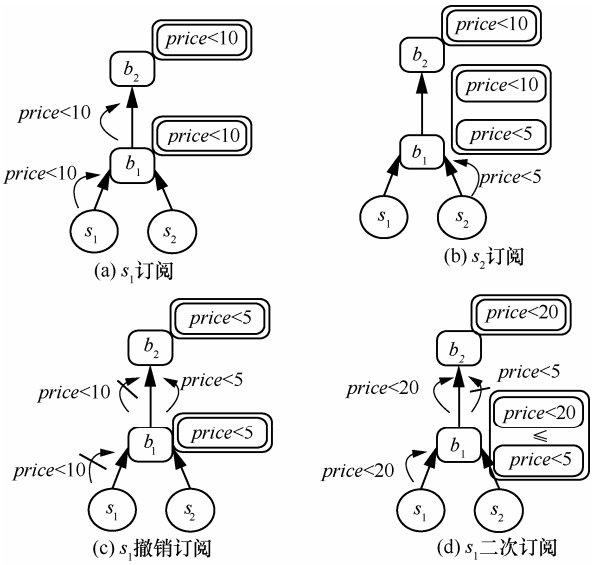


图 1　基于常量订阅的订阅变更（二次订阅方式）

4　向量订阅算法

4.1　向量订阅

常量订阅的变更需要频繁地改变订阅偏序树的结构，造成大量的时间浪费，针对这一问题，本文提出向量订阅的概念，具体如下。

定义 6 向量订阅（VS, vector subscriptions）。向量订阅中，订阅结构 $P ::= a\ c\ u$ 的条件 c 和属性值 u 均视为变量，进而形成了一个向量 $\boldsymbol{v}$，订阅结构可表达为 $P ::= a\ \boldsymbol{v}$。

按照向量订阅的定义，针对图 1 所示的订阅变更，其更改过程如图 2 所示。图 2(a)、图 2(b)为 s_1 和 s_2 的订阅过程，将值 5、条件“<”、值 10、条件“<”按照向量方式传递。此时，s_1 要将其订阅内容变更为“*price*≤20”，只需将 $\boldsymbol{v}_{s_1}.u$ 设为 20、$\boldsymbol{v}_{s_1}.c$ 设为“≤”即可，又由于这一变更仍然没有改变之前建立的“*price* $\boldsymbol{v}_{s_1}$”包含于“*price* $\boldsymbol{v}_{s_2}$”的结构，只需再将 $\boldsymbol{v}_{b_1}.u$ 设为 20、$\boldsymbol{v}_{b_1}.c$ 设为“≤”即可。在本例中，向量订阅避免了常量二次订阅带来的退订和再订过程以及相应偏序关系的重建。

4.2 面向匹配树型结构的向量订阅变更算法

下面以匹配树结构中典型的 Poset 算法为例，给出其订阅向量变更算法。

Poset 匹配树结构如图 3 所示。将订阅按照订阅内容的覆盖关系组织成偏序树，覆盖关系为上级包含下级，即满足下级订阅的条件的事件也一定满足其上级所有订阅，反之不成立。匹配树的节点主要包含以下信息：条件过滤器 fliter，表示订阅条件；订阅者集合 subscribers，代表订阅该条件的订阅者集合，有可能是一个，也有可能是多个，但不能为空，一旦为空便会将该节点删除；前驱集 precursors，表示该节点的所有前驱，有可能是一个或多个，也可能为空；后继集 successors，表示该节点的所有后继，有可能是一个或多个，也可能为空。

在 Poset 中，当订阅规模达到一定程度时，订阅之间的覆盖关系会变得十分复杂。如果在这种情况下新增加或删除一个节点，很容易导致订阅树结构的大规模改变，所需要的操作十分繁琐。向量订阅较之常量订阅，能够归并更多不必要的增/删操作。其变更过程如算法 1 所示，算法描述如下。

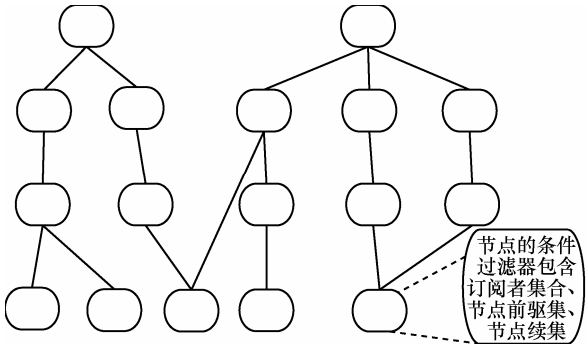


图 3 Poset 匹配树结构

Step1 找到要更改订阅向量的位置。

Step2 将新向量与旧向量的所有前驱和后继进行覆盖检测，如果新向量依然满足旧向量的前驱和后继覆盖关系，并且该节点只包含这一个订阅，则说明可以直接进行订阅向量的替换。

Step3 如果新向量不满足旧向量的覆盖关系或该节点包含多个订阅，则说明不能直接替换，需要执行 unsub/sub 操作。

可见，算法 1 中向量订阅不需要取消旧的订阅再加入新的订阅，而是直接进行变更操作，可以节省一次通信；变更过程尽可能避免了不必要的结构变更操作。

算法 1 changeInPoset(*old*, *new*)

输入: *old*: 原订阅向量

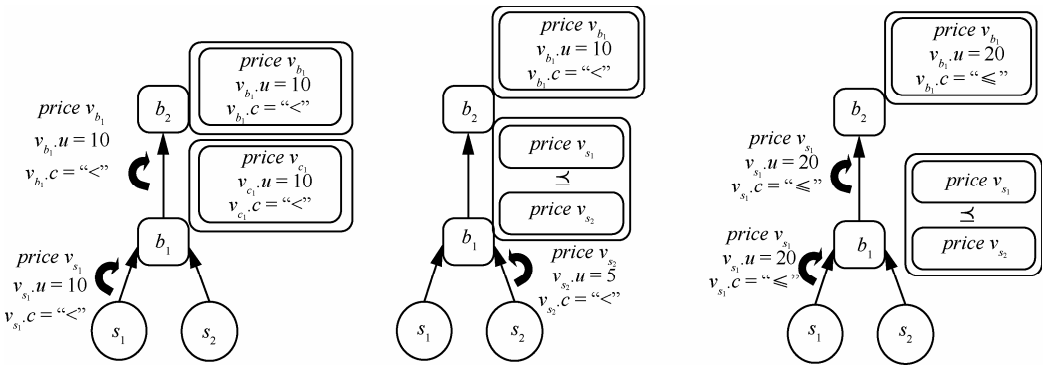
new: 新订阅向量

1) Bool *flag*=TRUE

2) FOR each *old's* Precursor *p*

3) IF !Cover(*p.fliter*, *new.fliter*)/*判断覆盖关系*/

4) *flag* = FALSE



(a) s_1 订阅 (b) s_2 订阅 (c) s_1 订阅更新

图 2 向量订阅中的订阅变更

```
5)      BREAK
6) IF   flag
7)   FOR each old's Successor s
8)       IF !Cover(new.fliter,s.fliter)
9)            flag = FALSE
10)           BREAK
11) IF flag && old.Subscriber==1
12)   old.fliter = new.fliter
13) ELSE
14)   Unsubscribe(old)
15)   Subscribe(new)
16) RETURN
```

5　向量共享算法

向量订阅体现了同一订阅者前后变更向量之间的关联带来的优化。而在内容发布/订阅系统中，代理节点往往连接大量订阅者，运行在代理上的订阅更新算法也需要面对大量的订阅向量变更。这些来自不同订阅者的订阅向量之间存在许多关联关系，可以对更新过程进行进一步的优化。

5.1　向量共享情况分析

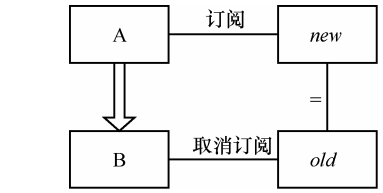
向量共享（VSS, vector subscriptions and sharing）的思想是针对某时间段内到来的订阅向量组，合并其中可以进行归约的向量，以减少对树结构的变更，从而提升变更效率。向量订阅依据具体操作的不同可分为 3 种：订阅者增加一个订阅向量(SUB (VS *new*))，订阅者取消一个订阅向量(UNSUB (VS *old*))，订阅者变更一个订阅向量(CHANGE(VS *old*, VS *new*))。这 3 类操作间可能存在的共享关系如图 4 所示。

1) 订阅与取消订阅共享情况表示的是订阅 A 要求增加一个新的订阅 *new*，同时订阅 B 要求取消一个旧的订阅 *old*，而恰好 *new* 与 *old* 相等。此时只需要将订阅 B 替换成订阅 A 即可。

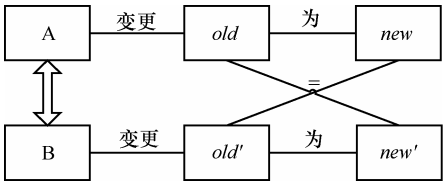
2) 订阅变更间共享情况表示订阅 A、B 同时要求变更订阅。订阅 A 要从 *old* 变更为 *new*，B 要从 *old*′变更为 *new*′，而恰好 *old* 与 *new*′相等，*old*′与 *new* 相等。即订阅 A 想要抛弃的正是订阅 B 想要获得的，而订阅 B 想要抛弃的也正是订阅 A 想要获得的。此时，只要将订阅 A、B 相互替换即可。

3) 变更与订阅共享情况表示订阅 A 要求从 *old* 变更为 *new*，同时订阅 B 要求增加一个订阅 *new*′，恰好 *old* 与 *new*′相等。这时，用订阅 B 替换订阅 A，并将订阅 A 的操作改为 SUB(VS *new*)。

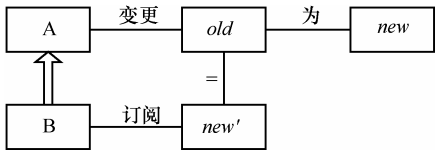
4) 变更与取消订阅共享情况表示订阅 A 要求从 *old* 变更为 *new*，同时订阅 B 要求取消一个订阅 *old*′，恰好 *old*′与 *new* 相等。这时，用订阅 A 替换订阅 B，并将订阅 A 的操作改为 UNSUB(VS *old*)。



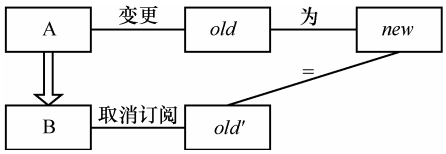
(a) 订阅与取消订阅共享



(b) 订阅变更间共享



(c) 变更与订阅共享



(d) 变更与取消订阅共享

图 4　订阅向量共享情况

5.2　向量共享算法

从上述情况可以看出，向量之间共享以取值相等为基础。向量共享算法的数据结构如图 5 所示。以向量值为键值，增加订阅的向量加入散列列表订阅散列(sub-hash)；取消订阅的向量加入散列列表取消订阅散列(unsub-hash)；变更订阅可以暂时看作取消订阅与增加订阅的合成操作，分别加入对应操作的 unsub-hash、sub-hash 中，并将分解后的 sub-hash 中的元素和 unsub-hash 中的元素都指向原变更散列(change-list)中的对应元素，并置标志位 ischange，表示是从 change-list 分解得到的。为了处理多个 SUB 操作共享同一个 UNSUB 操作的情况，为 unsub-hash 的元素加入标志位 used，表示该操作是否已被共享。

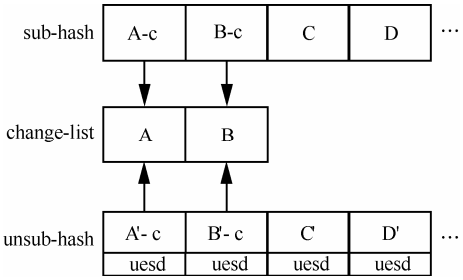


图 5 向量共享算法数据结构

向量共享算法流程如算法 2 所示，具体流程如下。

Step1 将待共享的 SUB、UNSUB、CHANGE 操作填入图 5 所示的散列表中。

Step2 对于 sub-hash 中的每一个元素 *subitem*，在 unsub-hash 中以相同键值获取元素。

Step3 如果成功获取元素，则说明配对成功，将 *subitem* 从 sub-hash 列表中删除，如果 *subitem* 或 *unsubitem* 带有 ischange 标志位，将其指向的元素从 change-list 中移除。置 used 标志位为真，在订阅结构中将 *unsubitem* 对应的订阅者替换为 *subitem* 对应的订阅者。

Step4 如果获取失败，则对下一个元素执行 Step2，直到 sub-hash 遍历完毕。

Step5 最后遍历 sub-hash 和 unsub-hash2 个列表中的元素。在 sub-hash 中，如果带有 ischange 标志位并且其指向的元素仍然存在，将该元素从 sub-hash 移除；在 unsub-hash 中，如果 used 标志位为假，但是带有 ischange 标志位，且指向的元素仍然存在，将该元素从 unsub-hash 移除，如果 used 标志位为真，将该元素从 unsub-hash 移除。

Step6 处理 3 个列表中剩余的订阅信息。

算法 2 VSShare(sub-hash,unsub-hash, change-list)

输入:sub-hash：存放所有订阅操作的散列列表
unsub-hash：存放所有取消订阅操作的散列列表
change-list：存放所有变更订阅操作的列表

1) FOR each item *subitem* in sub-hash
2) *unsubitem*=unsub-hash.get(*subitem.fliter*)
3) IF successfully get *unsubitem*
4) Replace(*subitem*,*unsubitem*)//订阅替换
5) *unsubitem.used*=TRUE
6) IF *subitem*.ischange
7) change-list.remove(*sub.change*)
8) *subitem.change*=NULL
9) IF *unsuitemb*.ischange
10) change-list.remove(*unsubitem.change*)
11) *unubitem.change*=NULL
12) FOR each item *subitem* in sub-hash
13) IF *subitem*.ischange&&*subitem.change*!= NULL
14) sub-hash.remove(*subitem*)
15) FOR each item *unsubitem* in *unsub-hash*
16) IF *unsubitem.ued*||
17) (*unsubitem*.ischange&&*unsubitem.change*! = NULL)
18) unsub-hash.remove(*unsubitem*)
19) RETURN

6 算法分析

在向量订阅中，由于向量 $\boldsymbol{v}$ 对时间敏感，某时间点 t 上的 $\boldsymbol{v}$ 可表示为 $\boldsymbol{v}(t)$。因而，订阅 Φ 与事件 e 的匹配过程不仅仅与事件 e 的参数相关，而且跟时间 t 相关，可表达为 $\Phi(e,t)$ 。

设某个订阅进程 P_i 的订阅为 Φ_i，按照上述向量订阅时间特性的描述，假设进程 P_i 的订阅在 t_0 时刻变更，之后不发生改变，那么在 Φ_i 上发生的事件匹配是确定的。即要么 $\forall e,\forall t \geqslant t_0\ \Phi_i(e,t)$ 成立，要么 $\forall e,\forall t \geqslant t_0 \neg\Phi_i(e,t)$ 成立。基于此，可以对向量订阅系统的活性分析如下。

定义 7 向量订阅活性。如果 $\Phi_i(e,t_0)$，在时间 t_0 后发布的事件 e 最终可被进程 P_i 匹配。

向量订阅的活性是针对系统层面的，其物理含义为订阅更新的延迟对订阅匹配的影响。实际订阅处理过程中，在系统中订阅变更的发生时刻 t_0，实际的订阅结构并未更新完成，记订阅变更完成时刻为 t_f。在单节点或者同步情况下，通过各种技术来使 t_f 无限接近 t_0，且 t_f-t_0 是确定的。但是在内容发布订阅这类典型的异步分布式系统中，一个进程在限定时间内不可能通知另一个进程。因此，在 CPS 中，通常 t_f-t_0 未做限定，只能通过尽量减小其差值的方式提高系统的活性，避免一些虚假事件的过滤。下面给出向量订阅和向量共享算法对活性的影响分析。

显然，向量订阅算法降低了订阅变更的时间，因而可以减小 t_f-t_0 的值，使事件过滤更加精确，提高了系统的活性。

在向量共享算法中，将 t_f-t_0 划分为两部分 t_b、t_d，即 $t_f-t_{0=}t_b+t_d$。t_b 代表预处理时间、t_d 代表处理时

间。向量共享过程中会改变 t_b、t_d 的值。但是，一方面共享预处理的时间是有限的，t_b 的增加不会很大；另一方面订阅结构往往非常复杂，通过共享减少对复杂结构的变更次数会使 t_d 显著减少。因此，大多数情况下，共享会使 t_b+t_d 减小，进而提高系统的活性。由于预处理时间远小于处理时间，t_b 的增加值也远小于 t_d，即便在向量不可共享的情况下，也不会对活性造成影响。

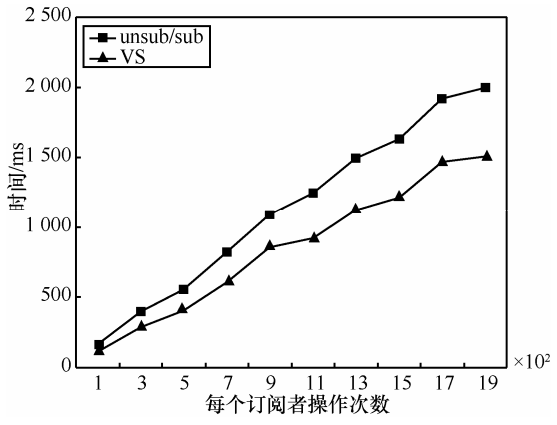
7　实验与结果

实验环境包括一台运行订阅更新算法的代理服务器、10 台用于订阅的主机、3 台用于发布的主机，百兆以太网环境。代理服务器配置为：CPU INTEL E8 500 2.93 GHz，RAM 2 GB，Windows XP Professional。为了验证面向 Poset 结构的向量订阅共享算法的效率，在 SIENA 环境中通过变更部分代码搭建了原型系统，并在典型的虚拟实验场景下，进行某武器系统的对抗实验。发布实体为若干战斗机，订阅实体为若干战术雷达。实验中的属性为飞机的 6 个自由度属性、位置属性、时间属性等，除时间属性是 int 类型外，其余为 double 类型。

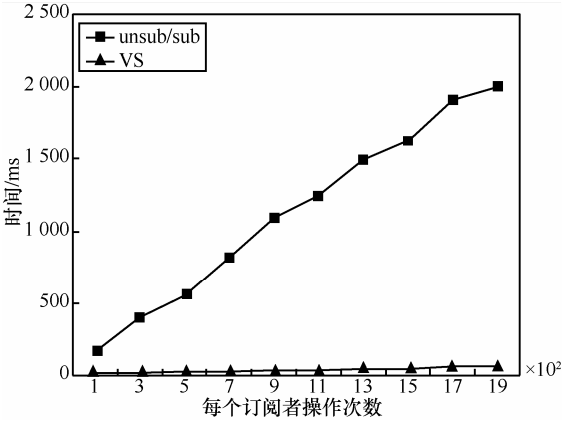
7.1　向量订阅与共享算法订阅性能分析

为了考察向量订阅、向量共享方法的订阅变更效率，构建了多种典型测试。实验环境均为一个代理服务器连接 10 个订阅者，实验过程为每个订阅者提交一定数量的订阅变更（从 100 次到 2 000 次），测试代理处理这些变更的时间。每组实验的测试结果如图 6 所示。实验中各订阅的谓词族数为 3~6 个。

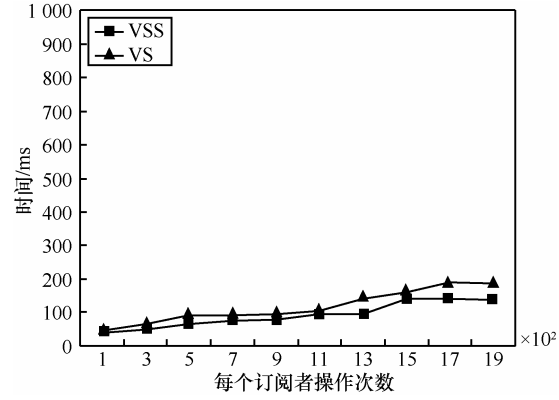
图 6(a)、图 6(b)为 VS 方式和 unsub/sub 订阅更新效率（时间）的对比，可见订阅的更新随着操作次数的增加呈线性关系增长。图 6(a)为变更改变偏序结构的情况，VS 方式的订阅更新比 sub/unsub 的速度有着一定的优势，这是因为 VS 的方式进行了部分通信原语的合并减少了通信原语的总体数量；图 6(b)为变更不改变偏序结构的情况，VS 方式的订阅更新 sub/unsub 的速度有极大幅度效率提高，这是因为如果按照传统的 unsub/sub 方式，偏序结构需要重新调整并建立 2



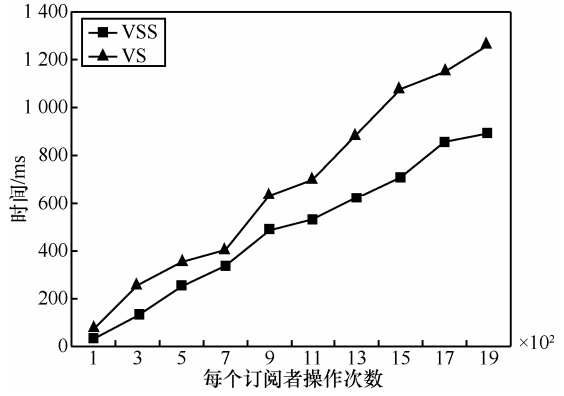
(a) VS 与 unsub/sub 对比（改变结构）



(b) VS 与 unsub/sub 对比（不改变结构）



(c) VS 与 VSS 的对比（不改变结构且 30%可共享）



(d) VS 与 VSS 的对比(改变结构且 30%可共享)

图 6　各情况下向量订阅与共享方法的订阅变更性能比较

次，而 VS 方式避免了这些重建时间。由于 2 次实验仅针对单层代理的情况，向量订阅对通信部分的优化没有凸显出来（如图 6(a)所示），在多级代理情况下，向量订阅对通信的优化将很好的体现出来。而且通常情况下，由于订阅变更间的相似性，不改变结构的情况会较多出现，因此 VS 对订阅效率的提高是很明显的。

图 6(c)、图 6(d)为向量共享方法（VSS）与 VS 操作的效率对比，假定有 30%的订阅间可以共享。图 6(c)针对的是偏序结构中 VS 不改变偏序结构的情况，此时无论是否有操作共享，对订阅更新时间的影响不大(虽然 VSS 进一步减少了处理时间)。图 6(d)列出偏序结构中 VS 改变偏序结构的情况。VSS 较 VS 又有一定的效率提高，这是因为 VSS 进一步通过向量共享减少了对偏序结构的改变。

7.2 向量订阅与共享算法综合性能分析

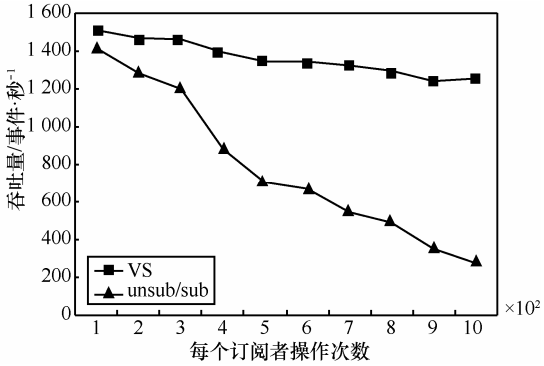
为了进一步考察向量订阅与共享方法（VSS）的优越性。在上节所述的实验条件下将实验过程变更为：发布者尽力发送事件，每个订阅者增加其订阅变更频率（从 100 次/秒到 1 000 次/秒），测试代理服务器的事件过滤吞吐量和订阅处理延迟的变化情况。为了方便进行 VSS 与同类算法的比较，在相同的环境下，设计了 2 组实验，实验参数值如表 2 所示，其中，不改变订阅结构的变更比例与订阅间可共享比例不重叠。每组实验的测试结果如图 7 和图 8 所示。

表 2 实验的参数设置

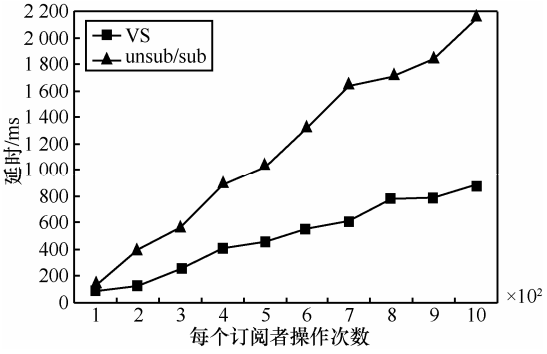
参数	实验 1	实验 2
属性数	7	9
订阅中的谓词族数	1~5	3~6
不改变订阅结构变更比例	20%	30%
订阅可共享比例	10%	20%

从图 7 和图 8 中可以看出：VSS 的吞吐量和延时都比 unsub/sub 方式有着更为优异的表现，VSS 和 unsub/sub 相比，一方面减少了一定的通信原语，另一方面避免了一部分不必要的结构变更，另外还有订阅共享对处理效率的提升(平均约 3 倍)。

横向比较实验 1 和实验 2，由于共享比例和不改变订阅结构变更比例的改变，实验 2 中 VSS 方式获得了更好的效率。

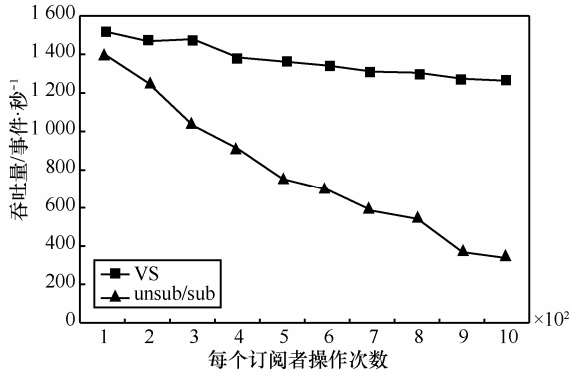


(a) VSS 和 unsub/sub 的事件过滤吞吐量对比

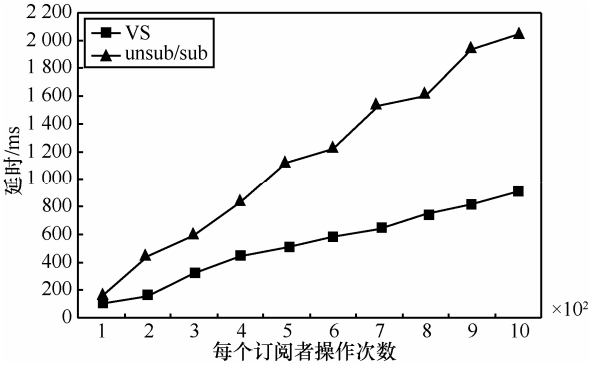


(b) VSS 和 unsub/sub 的订阅处理延时对比

图 7 实验 1 结果



(a) VSS 和 unsub/sub 的事件过滤吞吐量对比



(b) VSS 和 unsub/sub 的订阅处理延时对比

图 8 实验 2 结果

8　结束语

在已有常量二次订阅基础上，扩展订阅结构为向量结构，用以更好地表达订阅的变更；面向典型的匹配树结构提出了向量变更算法；并基于向量间的关联关系，提出了向量共享算法；进而提高了系统事件匹配吞吐量和动态订阅变更能力。在实例测试中，向量订阅与共享方法较传统的二次订阅方式有 3 倍以上的效率提升，更加适合应用在频繁动态订阅系统中。

参考文献：

[1] LI M, YE F, KIM M, *et al*. A scalable and elastic publish/subscribe service[A]. Proceeding of Parallel & Distributed Processing Symposium [C]. Anchorage, AK, 2011. 1254-1265.

[2] SCHOLTUS M, VAN D D, FRIJNS B. Speed, algorithmic trading, and market quality around macroeconomic news announcements[J]. Journal of Banking & Finance, 2014, 38(c): 89-105.

[3] AVAIDA M, BARHOURNI M, FOUCHAL H, *et al*. Joint routing and location-based service in VANET[J]. Journal of Parallel and Distributed Computing, 2014, 74(2): 2077-2087.

[4] JUSTIR C, ERAN T, JASON H, *et al*. Bridging the gap between physical location and online social networks[A]. Proceedings of the 12th ACM International Conference on Ubiquitous Computing[C]. New York, USA, 2010. 119-128.

[5] LI Z, LI X, DUONG T A, *et al*. Accelerating optimistic HLA-based simulations in virtual execution environments[A]. Proceedings of the 2013 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation[C]. New York, USA, 2013. 211-220.

[6] SADOGHI M, JACOBSEN H A. Be-tree: an index structure to efficiently match boolean expressions over high-dimensional discrete space[A]. Proceedings of 37th SIGMOD International Conference on Management of Data[C]. New York, USA, 2011. 637-648.

[7] SHEN Z H, TIRTHAPURA S, ALURU S. Indexing for subscription covering in publish-subscribe systems[A]. Proceedings of IEEE International Conference on Data Engineering[C]. Piscataway, 2005. 32-43.

[8] 陈继明, 鞠时光, 潘金贵,等. 基于内容的快速事件匹配算法[J], 通信学报, 2011, 32(6): 78-85.
CHEN J M, JU S G, PAN J G, *et al*. Content-based effective event matching algorithm[J]. Journal on Communications, 2011, 32(6): 78-85.

[9] KAZEMZADEH R S, JACOBSEN H A. Opportunistic multipath forwarding in content-based publish/subscribe overlays[A]. Proceedings of the 13th International Middleware Conference[C]. New York, USA, 2012. 249-270.

[10] ZJGOR S, AURKENE A, MIKEL L, *et al*. Mobile xsiena: towards mobile publish/subscribe[A]. Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems[C]. New York, USA, 2010. 91-92.

[11] SILVIA B, PASCAL F, MARIA G. Content-based publish/subscribe using distributed R-trees[A]. Proceedings of International Conference on Parallel and Distributed Computing[C]. Berlin, Germany, 2007. 537-548.

[12] CARZANIGA A, ROSENBLUM D S, WOLF A L. Design and evaluation of a wide-area event notification service[J]. ACM Transactions on Computer Systems (TOCS), 2001, 19(3): 332-383.

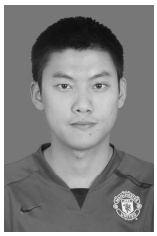
[13] JAYARAM K R, JAYALATH C, EUGSTER P. Parametric subscriptions for content-based publish/subscribe networks[A]. Proceedings of Middleware 2010[C]. Bangalore, India,2010. 128-147.

[14] JAYARAM K R, JAYALATH C, EUGSTER P. Parametric content-based publish/subscribe[J]. ACM Transactions on Computer Systems (TOCS), 2013, 31(2):491-501.

作者简介：

尤涛（1983-），男，河南陕县人，博士，西北工业大学讲师，主要研究方向为分布式事件系统。

吴其蔓（1992-），女，江苏连云港人，西北工业大硕士生，主要研究方向为数据库与数据挖掘。



王川文（1992-），男，陕西西安人，西北工业大学硕士生，主要研究方向为计算机应用技术。

钟冬（1979-），男，陕西西安人，博士，西北工业大学讲师，主要研究方向为计算机网络技术。

杜承烈（1970-），男，陕西西安人，博士，西北工业大学教授，主要研究方向为军用软件工程。