# 发布/订阅系统中的原子订阅管理和匹配

齐凤亮[1] 金蓓弘[1] 陈海彪[1] 龙震岳[2]
(中国科学院软件研究所软件工程技术中心 北京 100190)[1]
(中国科学院软件研究所计算机科学国家重点实验室 北京 100190)[2]

**摘 要** 如何有效地管理原子订阅并将事件与原子订阅高效地匹配,是发布/订阅系统需要关注的关键问题。首先将原子订阅组织成为一个覆盖森林,然后在这个结构上执行原子订阅的匹配,同时使用谓词的多级索引结构为原子订阅匹配提供支持。此方法已在基于内容的发布/订阅系统 OncePubSub 上实现。给出了用于验证算法性能和开销的实验。实验结果表明,上述方法具有良好的匹配性能和可伸缩性。

**关键词** 发布/订阅,原子订阅管理,原子订阅匹配

## On Primitive Subscription Management and Matching in Publish/Subscribe Systems

QI Feng-liang[1] JIN Bei-hong[1] CHEN Hai-biao[1] LONG Zhen-yue[2]
(Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[1]
(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)[2]

**Abstract** *Management of primitive subscriptions and matching events with them are fundamental issues in a publish/subscribe system.* This paper organized primitive subscriptions into a covering forest and then executes primitive subscription matching on the basis of multi-level predicate indexes. This approach was implemented in our content-based publish/subscribe system OncePubSub. This paper also gave the descriptions of the experiments which were conducted to evaluate performance and overhead. Experimental results prove that our approach has efficient matching performance and good scalability.

**Keywords** Publish/subscribe, Primitive subscription management, Primitive subscription matching

## 1 引言

发布/订阅系统作为一种信息交互和共享的中间件,在信息的生产者和消费者之间建立了一个松耦合的信息分发渠道。在发布/订阅系统中,信息的消费者被称作订阅者;与之对应,信息的生产者称为发布者。订阅者预先向系统发出订阅声明其感兴趣的事件;发布者向系统发布事件;发布/订阅系统将事件与系统中的订阅进行匹配,并且向订阅了该事件的订阅者发出事件通知。

通常,在基于内容的发布/订阅系统中,一个原子订阅是若干谓词的合取,一个事件是若干事件项的合取。在发出订阅和发布事件时,需要指明谓词和事件项的取值类型。发布/订阅系统 OncePubSub(通用版)为谓词和事件项定义了数值和字符串两种数据类型,同时为每种数据类型定义了丰富的操作符,提高了原子订阅的表达能力。

在本文中,根据原子订阅之间的覆盖关系将其组织成为订阅森林结构,订阅匹配过程能利用订阅森林结构进行剪枝,迅速过滤掉大量不能被匹配的订阅。同时,将谓词组织在多级索引结构中,用以加速原子订阅的匹配。实验结果证明,我们的算法能够高效地进行原子订阅匹配。即使是在匹配的过程中动态执行添加和删除订阅操作,匹配算法也具有很高的性能。同时,匹配算法在大量订阅和事件到来时也具有很好的可伸缩性。

## 2 相关工作

原子订阅的匹配作为发布/订阅系统的核心内容,目前已有的算法包括谓词计数算法、基于树的算法、二叉判定图算法等。并且,Kale[1]通过使用一种简化的通用模型,证明了原子订阅匹配算法与 Partial Match Problem 是等价的。

谓词计数算法在匹配一个事件时需要测试所有的谓词。如果一个订阅中被满足的谓词数目等于它所包含的谓词总数,则说明这个订阅被满足了。谓词计数算法做了许多的无用功,因为如果一个谓词不能被满足,该订阅中的后续谓词仍然会被测试,而实际上后面的测试已经不需要了。Ashayer[2]改进了谓词计数算法,但仍然存在上述问题。

在基于树的算法中,订阅被组织成为一棵树。树中的每一个非叶子节点表示是对一个谓词的测试,出边标识了该测试的结果,订阅存放在叶子节点中。如果事件从树的根节点

到稿日期:2009-01-09 返修日期:2009-03-17 本文受国家高技术研究发展计划 863 资助项目(编号 2006AA04A119)资助。
**齐凤亮** 硕士研究生,CCF 会员,研究方向为分布式计算、软件工程,E-mail:qfl06@otcaix.iscas.ac.cn;**金蓓弘** 博士,研究员,CCF 高级会员,研究方向为分布式计算、软件工程;**陈海彪** 硕士研究生,研究方向为分布式计算、软件工程;**龙震岳** 博士研究生,CCF 会员,研究方向为分布式计算、软件工程。

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[3]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成线性关系。但是Gryphon并不能及时地反映用户更新订阅的要求，因为添加和取消订阅是以一种增量的方式处理的。

使用BDD[4]的方法将订阅组织成为一个BDD(二叉判定图)，其中非终点的节点表示一个谓词，出边被标识为0或1，由此将订阅的匹配转化成了对BDD的判定问题。MBDD[5]修改了BDD并且使用订阅的历史信息、谓词间的逻辑关系等提高了事件路由和匹配的性能。虽然BDD方法的优点在于它支持谓词之间“与”和“或”关系，但是如果可以使用复合订阅[6]表示原子订阅之间的逻辑关系，那么这个优势将不复存在。

SIENA[7]根据订阅之间的覆盖关系将其组织成为Poset(Partially ordered set，偏序集)，但是Poset结构在添加和取消订阅时会引入大量的订阅覆盖关系的判断。同时，在事件匹配过程中，不同订阅的同一个谓词可能会被多次测试，这样会对匹配的性能造成不良的影响。

## 3 订阅管理和匹配

### 3.1 订阅森林构造

在OncePubSub中，原子订阅将根据覆盖关系组织成为一个订阅森林，森林中的每棵树被称为订阅树。在一个订阅树中，每个节点表示一个订阅，所有的节点满足下面两个约束：(1)父节点的订阅覆盖孩子节点的订阅；(2)在同一层次中的节点之间可能存在覆盖关系，也可能不存在覆盖关系。图1显示了一个订阅森林结构。

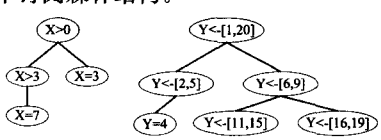


图1 订阅森林

由于在一棵订阅树中，不同层次之间的订阅存在着覆盖关系，订阅树在匹配事件的过程中存在着两个特性：(1)如果一个节点匹配了一个事件，那么这个节点的所有祖先节点也一定匹配这个事件；(2)如果一个节点与事件不匹配，那么该节点的所有子孙节点都与这个事件不匹配。

订阅森林的结构就是使用第二个性质来加速匹配的过程。是否进入以一个节点为根的子树里继续搜索，取决于事件是否与该节点匹配成功。

在添加订阅newSub时，系统首先判断系统中是否有与newSub一样的订阅。如果有，只需把新的订阅者加入到可复用订阅的订阅者集合中即可。否则，就将新订阅加入到森林中，即在森林中找到这样一个节点pNode，它覆盖了newSub，并且pNode的所有孩子都不能覆盖newSub，这样的节点将作为newSub的父节点。然后将被newSub覆盖的pNode的孩子节点转而链接到newSub的孩子节点集合中。如果没有任何树的根节点覆盖newSub，那么就将它作为一个新的订阅树的根节点，并且将被其覆盖的其他树的根节点加入到它的孩子集合中。当然，在查找覆盖newSub的节点过程中，如果某个节点覆盖了newSub，并且订阅者集合包含了newSub的订阅者，那么添加过程将会终止，因为已经有相同的订阅者订阅了一个约束范围更大的订阅。

一旦newSub被加入到订阅森林，在以newSub为根的子树中，将newSub的订阅者从该子树中所有订阅的订阅者集合中删除。如果一个节点由于删除了newSub的订阅者而导致订阅者集合为空，那么就将这个节点删除，并且将其子节点集合加入到它们祖父的子节点集合中。

当用户取消订阅unSub时，系统将会在订阅森林里的每一棵订阅树中执行订阅的取消操作。在每一棵订阅树中，从树的根节点开始，如果有任何一个节点被unSub覆盖了，那么就将unSub的订阅者从以该节点为根的子树的所有节点的订阅者集合中删除。如果在删除后某个节点的订阅者集合为空，那就将这个节点从订阅树中删除，并且将它的所有孩子链接到它们的祖父节点上。

### 3.2 谓词管理

如前所述，系统中存在着数值型和字符串型两种谓词。将不同类型的谓词存储在不同的谓词存储结构(Predicate Storage Structure，PSS)中，并且为它们建立了多级索引结构。

将数值类型的操作符号为>，<，=，>=，<=，！=等的谓词存储在红黑树结构中，为区间类型<-(属于符号)和<+(不属于符号)的数值型谓词建立区间树，作为其存储结构。对于字符类型谓词，我们定义的操作符号包括*>(前缀)、>*(后缀)、*=(包含)、=(相等)、!=(不等)、<-(属于集合)，<+(不属于集合)等，使用Trie树作为存储结构。然后，为谓词建立分别以谓词类型、属性名称和操作符为键的多级索引结构，如图2所示。

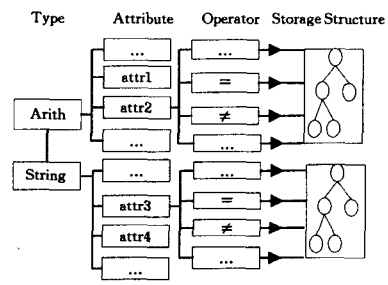


图2 谓词管理结构

在我们的系统中，每个谓词都被赋予了一个唯一的ID，并且都包含有一个标记，记录着该谓词对于一个到来的事件是否被测试过、经过测试被匹配或者没有被匹配上的3种不同的状态。并且每个PSS包含有一个是否被访问过的标记。

按照上述谓词管理方式，即使多个订阅中包含着相同的谓词，谓词仍然只被存储了一份，在事件到达并与订阅进行匹配时一个谓词也最多只被测试了一次。

### 3.3 原子订阅匹配

原子订阅匹配算法是在订阅森林中搜索可以匹配成功的订阅。搜索过程类似一个宽度优先搜索，使用了一个先进先出队列来管理需要被访问的节点。算法首先将森林中所有树的根节点放入到队列中。当队列不为空时，从队列中取出一个节点，验证它是否能够被到达的事件满足；如果经过测试，证明是与事件匹配的，就把该节点的所有孩子节点都放入到队列中。如果队列不为空，就从队列中取出另外一个节点，再

次执行上面的过程。

一个订阅中的所有谓词都被事件满足，则该订阅将被事件满足，通过检查谓词的标记可以知道该谓词的测试状态。如果一个谓词还没有被测试过，并且事件中包含有相同类型和属性名的事件项，那么就在该谓词类型、属性和操作符所确定的 PSS 中测试该谓词。通过使用 PSS，可以避免冗余的谓词测试。例如，在 Trie 树中，如果经过测试判断出一个事件 $e$ 满足一个前缀类型的字符串谓词(STRING: name * >ThinkPad X60s)，那么该测试之后无需再进行任何其他的测试，就可以知道 $e$ 肯定也满足类似(STRING: name * > ThinkPad X) 或者 (STRING: name * > ThinkPad)等这些值以"ThinkPad X60s"为前缀的谓词。图 3 给出了算法的伪代码。

```
    MATCH-FOREST(event)
1 for earch predicate p
2     do state[p]←TO-MATCH
3 for each PSS pss
4     do visited[pss]←FALSE
5 S←∅ ▷S is a set that stores matched subscriptions
6 Q←∅ ▷Q is a FIFO queue
7 for each tree T in the subscription forest
8     do ENQUEUE(Q,root[T])
9 while Q≠∅
10     do sub←DEQUEUE(Q)
11         if MATCH-SUB(sub,event)=TRUE
12             then INSERT(S,sub)
13                  for each child c of sub
14                      do ENQUEUE(Q,e)
15             else continue
16 return S
MATCH-SUB(sub,event)
1 for each predicate p in sub
2     do value←GETVALUE(event, attr[p])
3         if value=NIL
4           the state[p]←NOT-MATCHED
5               return FALSE
6         if MATCH-PREDICATE(p,value)=TRUE
8           then continue
9           else return FALSE
9 return TRUE
MATCH-PREDICATE(p,value)
1 if state[p]=MATCHED
2     then return TRUE
3 if state[p]=NOT -MATCHED
4     then return FALSE
5 pss←GETPSS(p) ▷Get the PSS of p
6 if visited[pss]=TRUE
7     then state[p]←NOT-MATCHED
8         return FALSE
9     else visited[pss]←TRUE
10        R←∅ ▷R is a temporary set
          ▷ Then get all matched predicates in pss
11        R←SEARCH-PSS(pss,value)
12        for each predicate q in R
13            do state[q]←MATCHED
14        if state[p]=TO-MATCH
15        then state[p]←NOT -MATCHED
16            return FALSE
17        else return TRUE
```

图 3　原子订阅匹配算法

最后，分析算法的时间复杂度。假定 $M$ 是系统中订阅的数目，$K$ 是系统中总的谓词数目，$N$ 是系统中 PSS 的数目，$P_s$ 是当事件到达后订阅可能被测试到的概率。有了上面的假设，那么 $c_n=K/N$ 是 PSS 中谓词的平均数目，$P_s \cdot M$ 是调用 Match-Sub 方法的总数。

对于一个包含有 $n$ 个谓词的订阅，第 $i(i>1)$ 个谓词可以被测试到，当且仅当它前面的 $i-1$ 个谓词都经过了测试并且可以被事件满足。假定一个谓词可以被事件满足的概率为 $P_e$，则 $P_e^i$ 表示订阅中的第 $i$ 个谓词被测试并且被事件满足的概率；$(1-P_e) \cdot P_e^{i-1}$ 表示第 $i$ 个谓词被测试并且没有被事件满足的概率。根据图 3，当事件满足了该订阅的所有谓词，或者遇到某一个不满足的谓词时，将会从 Match-Sub 方法退出，所以 Match-Sub 的执行开销为 $\sum_{i=1}^{n}(1-P_e)P_e^{i-1} \cdot i \cdot r + P_e^n \cdot n \cdot r$，其中 $r$ 是调用 Match-Predicate 方法一次的开销，$r=(N/K)CostOfSearchPSS+(1-N/K) \cdot O(1)$，其中 $CostOfSearchPSS$ 是在一个 PSS 中查找到所有被满足的谓词的开销。当在红黑树里测试数值类型的谓词时，这个开销为 $O(\log_2 c_n)+O(c_n)$；在 Trie 树中测试字符串类型的谓词时，这个开销为 $O(strLen)$，其中 $strLen$ 是字符型事件项中字符串的平均长度。

所以，匹配一个事件的时间开销可以表示为：

$$Cost=(\sum_{i=1}^{n}(1-P_e)P_e^{i-1} \cdot i \cdot r + P_e^n \cdot n \cdot r) \cdot P_s \cdot M$$

## 4　实验与分析

在 OncePubSub 中实现了第 3 节提出的原子订阅管理和匹配算法，并通过实验将 OncePubSub(在实验中简称 ONCE)与 SIENA 进行了性能对比。实验用的计算机采用单核 Pentium 4、主频为 3.0 GHz 的 CPU、1GB RAM 并使用 Windows XP SP2 系统，算法实现和实验均采用 JDK1.6。

实验所使用的订阅包含 2 到 5 个谓词，其中属性名称为 price 和 category 的两个谓词是必选的。一个包含有 5 个谓词的订阅具有下面的形式：(ARITH: price<-($v1$, $v2$), ARITH: discount > $v3$, STRING: category = $v4$, STRING: name * >$v5$, STRING: supplier<-$v6$)，其中 $v1$, $v2$ 和 $v3$($v1>v2$)是整数类型，$v4$ 和 $v5$ 是字符串常量，$v6$ 是一个字符串的集合。发布的事件至多包含有 5 个事件项。包含有 5 个事件项的事件具有以下的形式：(ARITH: price=$v1$, ARITH: discount = $v2$, STRING: category = $v3$, STRING: name = $v4$, STRING: supplier= $v5$, $v6$)，其中 $v1$ 和 $v2$ 是整数，$v3$, $v4$ 和 $v5$ 是字符串常量，$v6$ 是事件发生的时间戳。

实验所用的订阅是从一个大小为 100,000 的订阅样本空间中取出的，所有的订阅被划分到 1000 棵订阅树中，每个订阅树中包含着 100 个订阅并且每个订阅的订阅者都互不相同。在一棵订阅树中，父节点覆盖子节点，每个节点包含有 1 到 3 个孩子节点。不同订阅树中的订阅之间没有覆盖关系。

实验 1 是测试事件匹配性能的压力测试。在这个实验中，首先向系统中添加一定数目的订阅，然后不断地向系统中

注入 10,000 个事件。这 10,000 个事件中,有 50%的事件在系统中是有满足的订阅的,并且满足的订阅是随机的。订阅集合中订阅的取法如下:首先从 1000 棵订阅树的森林中随机选择出一棵树,然后从这个树中选出 1 到 100 之间随机数目的订阅,再继续从这个森林中随机选出另外一棵树并选出随机数目的订阅。这个过程一直持续下去,直到选出需要数目的订阅为止。这样,从一棵树中选出的订阅的平均数目为 50,并且这些订阅都是存在覆盖关系的,在实验中记录事件匹配的时间和系统内存的占用开销。

图 4 显示了不同系统在该实验中的时间和空间开销。其中,$X$ 轴表示原子订阅的数目,原子订阅从 1,000 增长到 10,000。实验结果表明,与 SIENA 相比,OncePubSub 在原子订阅匹配上具有很大的优势。当系统中包含有 10,000 条订阅时,ONCE 比 SIENA 在时间开销上低了 87.1%,在空间开销上低了 21.0%。

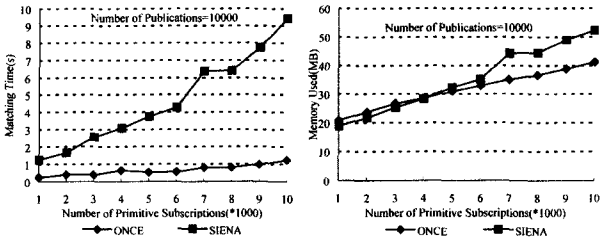


图 4 实验 1 记录的原子匹配时间和空间开销

在发布/订阅系统的实际运行中,订阅与事件匹配、订阅的添加、订阅取消经常会交织进行。实验 2 就是对这一场景进行模拟,进而记录系统在这些操作中的性能和开销。

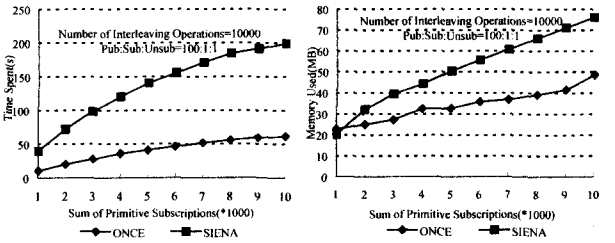


图 5 实验 2 记录的操作的时间和空间开销

首先向系统中注入一定数目的订阅,然后开始下面场景的模拟过程:系统接收到一系列订阅请求,然后又接收到相同数目的取消订阅请求,这个请求数目从{10,20,30,40,50}中随机选取。然后,系统将接收到数量为订阅请求数目 100 倍的事件,进行订阅匹配。订阅、取消的订阅和事件的格式都符合上一个实验所作的规定。3 种操作一直循环执行,直到操作总数目到达 10,000 为止。在这一过程中,系统中的订阅总数总是保持恒定。我们记录这 10,000 个操作的时间和空间开销。

如图 5 所示,OncePubSub 在这种交织操作中的性能也有很大的优势。例如,当系统中订阅数目维持在 10,000 时,OncePubSub 执行 10,000 次交织操作所用的时间仅为 SIENA 的 30.1%,所使用的内存仅为 SIENA 的 60.0%。通过观察图 4 和图 5 的实验数据,可以看出 OncePubSub 在大量的订阅和事件数据下具有良好的可伸缩性。

**结束语** 本文设计了一种高效的原子订阅管理和匹配算法,并在发布/订阅系统 OncePubSub 中实现。该算法是将订阅组织成订阅森林结构,这样可以迅速地将大量不匹配的订阅过滤出来,同时对谓词加以组织,对其进行有效的存储和高效匹配,从而达到快速匹配事件的目的。该算法也能够高效地支持大量订阅的动态添加和删除。实验验证了上述优点,并且与 SIENA 相比,我们的系统在时间和空间开销上具有很大的优势。

## 参 考 文 献


[1] Kale S, Hazan E, Cao F, et al. Analysis and Algorithms for Content-based Event Matching[C]// the 4th International Workshop on Distributed Event-based Systems. Columbus, Ohio, USA, June 2005

[2] Ashayer G, Leung H K Y, Jacobsen H A. Predicate Matching and Subscription Matching in Publish/Subscribe Systems[C]// the 22nd International Conference on Distributed Computing Systems Workshops. Vienna, Austria, July 2002

[3] Aguilera M K, Strom R E, Sturman D C, et al. Matching Events in a Content-based Subscription System[C]// the 18th ACM Symposium on Principles of Distributed Computing. Atlanta, GA, USA, May 1999

[4] Campailla A, Chaki S, Clarke E, et al. Efficient Filtering in Publish-Subscribe Systems Using Binary Decision Diagrams[C]// the 23rd International Conference on Software Engineering. Toronto, Ontario, Canada, May 2001

[5] Li G, Huo S, Jacobsen H. A Unified Approach to Routing, Covering and Merging in Publish/Subscribe Systems Based on Modified Binary Decision Diagrams[C]// the 25th International Conference on Distributed Computing Systems. Columbus, Ohio, USA, June 2005

[6] Zhao X C, Jin B H, Yu S, et al. Composite Subscription and Matching Algorithm for RFID Applications[C]// the 22nd IEEE International Conference on Advanced Information Networking and Applications. Ginowan, Okinawa, Japan, March 2008

[7] Carzaniga A, Rosenblum D S, Wolf A L. Design and Evaluation of a Wide-area Event Notification Service[J]. ACM Transaction on Computer Systems, 2001, 19(3): 332-383


(上接第 88 页)


[5] Caceres R, Duffield N G, Horowitz J, et al. Multicast-Based Inference of Network-Internal Characteristics Accuracy of Packet Loss Estimation[C]// IEEE INFOCOM. New York, 1999

[6] Duffield N G, Presti L F, Paxson V, et al. Inferring link loss using striped unicast probes[C]// IEEE INFOCOM 2001. Anchorage, Alaska, 2001, 2: 915-923

[7] Presti F L, Horowitz J, Towsley D, et al. Multicast-based inference of network-internal delay distributions[J]. IEEE/ACM Transactions on Networking, 2002, 10(6): 761-775

[8] Coates M, Nowak R. Network loss inference using unicast end-to-end measurement[C]// ITC Seminar on IP Traffic, Measurement and Modeling. Monterey, CA, 2000, 28: 1-9

[9] Bestavros J, Byers W, Harfoush K A. Inference and labeling of metric-induced network topologies[J]. IEEE Transactions on Parallel and Distributed Systems, 2005, 16(11): 1053-1065

[10] Lawrence E, Michailidis G, Nair V. Network Delay Tomography Using Flexicast Experiments[J]. Journal of Royal Statistical Society Series B, 2006, 68(5): 785-813