分布式停车场管理系统的实现

周昊一 MF1233055

1. 任务描述

某停车场共有 TOTAL-NUM 个车位, ENTRY-NUM 个进口, EXIT-NUM 个出口. 现需要一个用于停车场控制汽车进出的分布式系统, 在该系统中没有集中的管理者(central server), 每个进(出)口通过通信平等协商保存当前车库的状态信息(如空闲车位数 UNOCCUPIED-NUM 等), 并据此决定是否允许车辆进入, 为简便计, 假定通信是可靠的.

1) 不考虑节点/进程失效的情形, 设计用于该停车场控制的分布式系统, 并给出汽车进出时使用该系统的方法.

2) 证明你所设计的分布式系统中使用的同步算法满足 ME1-ME3*.

3) 如果新增一个进口节点/进程, 请考虑如何使该进口能参与工作.

4) ME1: (safety) At most one process may execute in the critical section(CS) at a time.

5) ME2: (liveness) Requests to enter and exit the critical section eventually succeed.

6) ME3: (ordering) If one request to enter the CS happened-before another, then entry to the CS is granted in that order.

2. 问题分析

为了简化模型,并且更贴近实际情况,我们考虑入口和出口是一体的.即每个节点既可以当作入口,也可以当作出口.

结合 ME1~ME3 以及具体的情况,总共需要满足以下 4 个条件

1) 如果有空位,总有一辆车能停进去
2) 先来的车(不管在那个节点)总是先停进停车场
3) 当停车场中没有空余车位时,没有车可以停入.
4) 任一节点能够通过一定方法知道在某一个时刻的真实剩余车位数

其中 1~3 是正确管理停车场所需要满足的条件.条件 4 是想要中途添加新节点所必需满足的条件,和前三个条件是相互独立的.因此在下文中,我们先考虑前三个条件,最后再考虑第四个条件.

由于出停车场时采用不阻塞的方式,不需要满足任何条件.

因此对于每一个节点来说,需要知道(直接或间接)所有正在等候的车辆的顺序,这个顺序不仅要对每个节点都是相同的,而且还要和车辆排队的次序相同.

并且每一个节点还要能直接或间接的知道当前停车场中是否有车位的准确信息.

3. 简单的模型(满足条件 1)

如果只需要满足第一个条件,用一个非常简单的模型就可以实现:

作为入口,每个节点N_i在本地记录一份剩余车位数P_i.当$P_i > 0$时,让本地队列中的第一辆车进入停车场,并将$P_i - 1$,并通知其他所有节点这个消息.

作为出口,每个节点在观察到有一辆车从这个节点驶出时,先将本地的$P_i - 1$,再将这个消息发送给其他所有节点.

由于通信是可靠的,因此当有一辆车从停车场驶出时,至少有一个仍有车辆在排队的节点会先观察到本地副本>0,此时这个节点就会让队列中的第一辆车停入.所以这样的模型可以满足条件 1.

对于条件 2,由于每当一个节点观察到本地的$P_i > 0$,就会放行本地队列中的第一辆车,因此车辆停入的顺序实际上不是车辆到来的全局次序,而是按照节点观察到出现空余车位的次序.

对于条件 3,由于每当一个节点观察到本地的$P_i > 0$,就会停入一辆车,因此如果有两个节点N_i, N_j同时观察到$P_i = 1, P_j = 1$,就会有两辆车同时停入.导致此时真实剩余车位数为-1.

4. 保证停车次序(条件 2)

如果想要保证车辆停入的次序和车辆到来的次序是一致的,就需要每个节点对于所有车辆停入的顺序都一个相同的认识,并且这个顺序还要和真实的全局顺序是相同的.

那么如何使所有节点对于车辆停入的顺序达成这样一致呢?

如果每个节点都有一个"准确"的时钟,对于任意两个车辆到来的时间都能给出相同并且正确的排序.那么这个问题也就解决了.因为这样的话,所有节点总能够对于车辆停入的顺序达成一致(用各自的时钟进行排序),并且又是正确的.

显然,这样的时钟不一定要是物理时钟.并且使用物理时钟本身会有许多难以解决的问题,比如时钟本身的同步.

我们需要的仅仅是能够正确的标出任意两个事件发生的先后顺序(或者能标明这两件事没有先后顺序).

因此这里我们采用向量时钟来解决这个问题.

每个节点维护一个向量时钟VC_i, VC_i的维数就是节点的个数.

当某个节点发生了一个事件时,它的向量时钟中自身的那一维+1.

两个向量时钟进行同步的结果是每一维取两者的最大值.

向量时钟$VC_1 < VC_2$当且仅当VC_1中的每一维都小于等于VC_2中的每一维,并且至少有一维是严格的小于关系.向量时钟小的时间一定在向量时钟大的事件之前发生.如果$!(VC_1 \leq VC_2)\&\&!(VC_2 \leq VC_1)$,则两个时钟对应的事件是并发的.

特别地,对于$VC_1 = VC_2$的情况,我们认为 id 号较小的节点上的时钟要较小.

在第三节的内容与向量时钟上,我们就可以实现满足条件二的模型:

每个节点有一辆车来临时,将这个事件加入本地的事件队列中,并向其他所有节点发一个消息.当某个节点收这个消息后,也将这个事件加入本地的队列中.

在某个节点观察到本地剩余车位数>0 时,检查事件队列中最早的那一个,如果是一个来自其他节点的事件,就向那个节点发送一个确认;如果是本地的事件,就在收到所有其他节点发来的确认后执行这个动作(车位数-1,并将队列中的第一辆车停入),并向其他节点再发一个通知,其他节点收到这个通知后将本地车位数-1.

在这样的模型下,一个动作只有在所有节点都同意后才会被执行,又由于所有节点对于所有事件的次序的认识都是相同的,所以,每次实际执行的那个动作一定是所有节点公认的第一个.

5. 保证剩余车位数大于 0(条件 3)

4 中的模型是可以保证条件三的.因为只有当所有节点数观察到的车位数都大于 0 时,才会有动作被执行.并且显然同一时间只有一个动作可以被执行.

6. 获取真实车位数(条件 4)

某一时刻的真实车位数其实就是任意一个节点的本地车位数再加上所有正在向这个节点传送的消息中包含的操作.因此,这个问题就可以转化为获取某一时刻的全局状态

我采用 Chandy-Lamport's Snapshot 算法来获取全局状态.

在任意时刻,假设某个节点想要获取时刻 A 的全局状态,就立即(在发送任何其他消息之前)向其他所有节点发送一个标记信息,并记录当前车位数.当一个节点收到标记信息后,如果自身还未记录当前车位数,就记录下当前车位数,并向所有其他节点发送一个标记信息.

当一个节点在已经记录下车位数时收到其他的消息(包含+1 或-1 操作),就将这个操作同步执行到记录的车位数上.

当一个节点向所有其他节点都发送过标记信息,并且收到了其他所有节点的标记信息后,该节点结束记录状态,此时所记录的车位数就是时刻 A 的真实车位数.

7. 添加新节点

在 6 的基础上,很容易就可以添加新节点.首先,新节点向所有已有节点发一个消息,告知其自己的存在.随后,新节点发起一次全局状态的获取(设为时刻 A),并要求其他节点在完成全局状态的获取后告知自己.

在随后过程中,新节点就像一个已有节点一样参与事务决策.

由于其他节点获取的全局状态肯定是相同的,并且都是时刻 A 的真实车位数,因此新节点只需要在收到第一个通知后,加上通知中附带的时刻 A 的真实车位数来更新此时的状态.具体方法是用自身现在的车位数加上 A 时刻的真实车位数,再减去自身在 A 时刻的车位数的 snapshot.

8. 实验

我在上述分析的基础上,用 JAVA RMI 实现了一个分布式的停车场管理系统,并进行了模拟测试.

实验环境为 Windows 7 64bit + eclipse 4.2,所用语言是 Scala.

为了观测,设置了一个观测节点,并为了方便起见同时将观测节点作为 Name Server 使用.

这个观测节点负责生成新节点(但并不负责添加新节点),安排车辆到来和离开的事件,并随机地将这些事件分配到某一个节点上去.

当有车辆停入停车场时,观测节点会判断这个动作是否满足以上前三个条件.并通过在停止一段时间的事件后对所有节点进行观察来检查是否所有节点记录的车位数副本一致并且正确.

以下是实验过程中的一些输出:

```
Monitor ready, java.RMI listening on //localhost:10000/monitor
Node 1 ready, java.RMI listening on //localhost:8001/parking
Node 2 ready, java.RMI listening on //localhost:8002/parking
Node 3 ready, java.RMI listening on //localhost:8003/parking
Node 4 ready, java.RMI listening on //localhost:8004/parking
Node 5 ready, java.RMI listening on //localhost:8005/parking
----------global state---------------
real parkingPlace=30
[Node 1]: parking place=30
[Node 2]: parking place=30
[Node 3]: parking place=30
[Node 4]: parking place=30
[Node 5]: parking place=30
-------------------------------------
```

图 1. 开始时添加 5 个节点

```
PARKED: car(No.0,Mon Dec 24 22:28:57 CST 2012) has parked in parking node 4, current parkingPlace=29
PARKED: car(No.1,Mon Dec 24 22:28:57 CST 2012) has parked in parking node 1, current parkingPlace=28
LEFT: car(No.1) has left from node 2 current parkingPlace=29
PARKED: car(No.2,Mon Dec 24 22:28:57 CST 2012) has parked in parking node 5, current parkingPlace=28
LEFT: car(No.2) has left from node 1 current parkingPlace=29
PARKED: car(No.3,Mon Dec 24 22:28:57 CST 2012) has parked in parking node 2, current parkingPlace=28
PARKED: car(No.4,Mon Dec 24 22:28:57 CST 2012) has parked in parking node 1, current parkingPlace=27
PARKED: car(No.5,Mon Dec 24 22:28:57 CST 2012) has parked in parking node 1, current parkingPlace=26
LEFT: car(No.5) has left from node 5 current parkingPlace=27
PARKED: car(No.6,Mon Dec 24 22:28:57 CST 2012) has parked in parking node 3, current parkingPlace=26
LEFT: car(No.6) has left from node 4 current parkingPlace=27
PARKED: car(No.7,Mon Dec 24 22:28:57 CST 2012) has parked in parking node 4, current parkingPlace=26
PARKED: car(No.8,Mon Dec 24 22:28:57 CST 2012) has parked in parking node 2, current parkingPlace=25
LEFT: car(No.7) has left from node 4 current parkingPlace=26
PARKED: car(No.9,Mon Dec 24 22:28:57 CST 2012) has parked in parking node 5, current parkingPlace=25
PARKED: car(No.10,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 5, current parkingPlace=24
LEFT: car(No.10) has left from node 5 current parkingPlace=25
PARKED: car(No.11,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 4, current parkingPlace=24
PARKED: car(No.12,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 4, current parkingPlace=23
```

图 2.可以观察到车辆驶入的顺序和到来顺序是一致的,此时没有排队的车辆

```
PARKED: car(No.54,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 1, current parkingPlace=4
LEFT: car(No.47) has left from node 2 current parkingPlace=5
PARKED: car(No.55,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 4, current parkingPlace=4
Node 6 ready, java.RMI listening on //localhost:8006/parking
PARKED: car(No.56,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 2, current parkingPlace=3
LEFT: car(No.54) has left from node 1 current parkingPlace=4
PARKED: car(No.57,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 4, current parkingPlace=3
PARKED: car(No.58,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 4, current parkingPlace=2
LEFT: car(No.52) has left from node 5 current parkingPlace=3
PARKED: car(No.59,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 2, current parkingPlace=2
PARKED: car(No.60,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 6, current parkingPlace=1
LEFT: car(No.51) has left from node 6 current parkingPlace=2
PARKED: car(No.61,Mon Dec 24 22:28:59 CST 2012) has parked in parking node 1, current parkingPlace=1
PARKED: car(No.62,Mon Dec 24 22:28:59 CST 2012) has parked in parking node 5, current parkingPlace=0
LEFT: car(No.16) has left from node 2 current parkingPlace=1
PARKED: car(No.63,Mon Dec 24 22:28:59 CST 2012) has parked in parking node 5, current parkingPlace=0
LEFT: car(No.44) has left from node 5 current parkingPlace=1
PARKED: car(No.64,Mon Dec 24 22:28:59 CST 2012) has parked in parking node 4, current parkingPlace=0
LEFT: car(No.35) has left from node 1 current parkingPlace=1
PARKED: car(No.65,Mon Dec 24 22:28:59 CST 2012) has parked in parking node 4, current parkingPlace=0
LEFT: car(No.39) has left from node 6 current parkingPlace=1
PARKED: car(No.66,Mon Dec 24 22:28:59 CST 2012) has parked in parking node 4, current parkingPlace=0
LEFT: car(No.65) has left from node 4 current parkingPlace=1
```

图 3.即使有车辆在排队(current parkingPlace=0),车辆停入的顺序仍然是正确的

```
LEFT: car(No.50) has left from node 3 current parkingPlace=6
PARKED: car(No.53,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 5, current parkingPlace=5
PARKED: car(No.54,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 1, current parkingPlace=4
LEFT: car(No.47) has left from node 2 current parkingPlace=5
PARKED: car(No.55,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 4, current parkingPlace=4
Node 6 ready, java.RMI listening on //localhost:8006/parking
PARKED: car(No.56,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 2, current parkingPlace=3
LEFT: car(No.54) has left from node 1 current parkingPlace=4
PARKED: car(No.57,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 4, current parkingPlace=3
PARKED: car(No.58,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 4, current parkingPlace=2
LEFT: car(No.52) has left from node 5 current parkingPlace=3
PARKED: car(No.59,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 2, current parkingPlace=2
PARKED: car(No.60,Mon Dec 24 22:28:58 CST 2012) has parked in parking node 6, current parkingPlace=1
```

图 4.添加了一个新节点(node 6),可以发现不久之后就有车辆可以停入其中

```
PARKED: car(No.112,Mon Dec 24 22:29:00 CST 2012) has parked in parking node 1, current parkingPlace=0
LEFT: car(No.80) has left from node 1 current parkingPlace=1
PARKED: car(No.113,Mon Dec 24 22:29:00 CST 2012) has parked in parking node 5, current parkingPlace=0
LEFT: car(No.88) has left from node 6 current parkingPlace=1
PARKED: car(No.114,Mon Dec 24 22:29:00 CST 2012) has parked in parking node 2, current parkingPlace=0
Node 7 ready, java.RMI listening on //localhost:8007/parking
LEFT: car(No.93) has left from node 1 current parkingPlace=1
PARKED: car(No.115,Mon Dec 24 22:29:00 CST 2012) has parked in parking node 6, current parkingPlace=0
LEFT: car(No.89) has left from node 1 current parkingPlace=1
PARKED: car(No.116,Mon Dec 24 22:29:00 CST 2012) has parked in parking node 3, current parkingPlace=0
LEFT: car(No.100) has left from node 1 current parkingPlace=1
```

图 5.又添加了一个新节点(node 7)

```
PARKED: car(No.296,Mon Dec 24 22:29:03 CST 2012) has parked in parking node 4, current parkingPlace=0
LEFT: car(No.33) has left from node 6 current parkingPlace=1
PARKED: car(No.297,Mon Dec 24 22:29:03 CST 2012) has parked in parking node 5, current parkingPlace=0
LEFT: car(No.277) has left from node 3 current parkingPlace=1
PARKED: car(No.298,Mon Dec 24 22:29:03 CST 2012) has parked in parking node 2, current parkingPlace=0
LEFT: car(No.266) has left from node 2 current parkingPlace=1
PARKED: car(No.299,Mon Dec 24 22:29:03 CST 2012) has parked in parking node 2, current parkingPlace=0
LEFT: car(No.286) has left from node 3 current parkingPlace=1
LEFT: car(No.293) has left from node 3 current parkingPlace=2
LEFT: car(No.253) has left from node 4 current parkingPlace=3
LEFT: car(No.270) has left from node 5 current parkingPlace=4
LEFT: car(No.298) has left from node 7 current parkingPlace=5
LEFT: car(No.294) has left from node 6 current parkingPlace=6
LEFT: car(No.192) has left from node 4 current parkingPlace=7
LEFT: car(No.170) has left from node 4 current parkingPlace=8
LEFT: car(No.162) has left from node 6 current parkingPlace=9
LEFT: car(No.287) has left from node 1 current parkingPlace=10
LEFT: car(No.297) has left from node 6 current parkingPlace=11
LEFT: car(No.295) has left from node 3 current parkingPlace=12
LEFT: car(No.260) has left from node 1 current parkingPlace=13
LEFT: car(No.263) has left from node 5 current parkingPlace=14
LEFT: car(No.292) has left from node 2 current parkingPlace=15
LEFT: car(No.283) has left from node 6 current parkingPlace=16
LEFT: car(No.276) has left from node 3 current parkingPlace=17
LEFT: car(No.113) has left from node 1 current parkingPlace=18
LEFT: car(No.225) has left from node 3 current parkingPlace=19
LEFT: car(No.296) has left from node 4 current parkingPlace=20
----------global state---------------
real parkingPlace=20
[Node 1]: parking place=20
[Node 2]: parking place=20
[Node 3]: parking place=20
[Node 4]: parking place=20
[Node 5]: parking place=20
[Node 6]: parking place=20
[Node 7]: parking place=20
-------------------------------------
```

图 6. 在最后,所有节点所保存的车位数副本都是一致,并且正确的,包括中途添加的两个新节点