普适计算环境下的作业迁移

李彬 杨士强

清华大学计算机科学与技术系，北京，100084

摘要：作业迁移问题是个比较老的问题。通常作业迁移有两种做法，第一种做法是，作业迁移基本都是由用户自己手工完成，用户自己要完成不同平台间作业任务的同步，不同平台间完成任务所需要的环境的配置等。这种做法的优点是可以充分发挥各个平台的性能，缺点是对用户能力的要求过高。另一个缺点是浪费用户的注意力资源。第二种做法是虚拟机的做法。通过定义一个虚拟机，使得平台间的差异对用户来说消失了。这种做法的优点是用户不必关心各种各样的平台了，对用户来说只有虚拟机平台。这种做法的缺点是它会对各个平台的性能造成一定程度的浪费。本文提出在普适计算环境无缝的作业迁移与描述方法，尝试在普适计算环境下克服传统方法的缺点。

关键词：描述方案；作业迁移；普适计算；

1. 引言

普适计算（Pervasive Computing）又叫无所不在的计算（Ubiquitous Computing），其研究领域涉及移动数据访问、支持移动用户通信的机制、智能空间、上下文感知、人们使用各种设备与环境交互的方法，以及怎么在一个设备上配置新的功能以充分利用它针对特定任务的接口特征等等。建立普适计算环境的目的，就是使我们管理信息能够更加容易，使人们的生活变得更加简单，使我们更加自由。

在普适计算的世界里，我们可以从任何地方，在任何时间里访问和操纵信息。那时候计算设备和网络将变得无所不在，计算将不再局限于台式机，工作站，大型主机，服务器等。在普适计算这个大环境下，我们就可以从一个新的角度看问题。作业迁移这个问题应该算是一个比较老的问题了。本文就尝试着探讨普适计算环境下的作业迁移，从一个新的角度看待作业迁移这个显得有点老套的问题。

2. 作业迁移

作业迁移就是将一个没有做完的任务从一个环境中搬到另外一个环境中接着做。现在对于作业迁移大体上有这么几种做法。第一种做法是，作业迁移基本都是由用户自己手工完成，用户自己要完成不同平台间作业任务的同步，不同平台间完成任务所需要的环境的配置等。这个做法的优点是针对各个任务能够充分发挥各个平台的各自的性能特点，使得任务的执行更有效率。这个做法的优点是很明显的，但是它的缺点也是同样的

突出。这个做法要求用户要非常地熟悉各种平台，能够在各种平台间游刃有余地完成任务的迁移。这就对用户的能力提出了相当大的要求，使得任务迁移成为少数这方面专家的专利，普通的用户对这只能是望洋兴叹，可望而不可及。但是普通的用户也有对任务迁移这方面的要求。从上面分析看，第一种做法是根本没法满足普通用户这方面的要求的。于是就有了第二种做法。这第二种做法最典型的就是虚拟机的做法。通过定义一个虚拟机（如 Java 虚拟机），使得平台间的差异对用户来说消失了，用户能够看到的只是虚拟机，至于虚拟机下面的那个真实的机器用户是见不到的，虚拟机对用户屏蔽了真实的机器。这种做法的优点是很明显的。通过虚拟机对用户屏蔽了真实的机器之间的差别，使得对用户来说，这个世界上只有一种机器，平台间的差异不再存在了，用户不再需要掌握各个不同平台的知识，任务的迁移变成只是在同一个平台上的事情了，这使得普通用户也能进行任务迁移了。但是这种做法也有其自身固有的缺点。现在对用户来说，所有平台都一样，看不到差别，用户所看到的都只是虚拟机。但是，虚拟机总是建构在实际的平台上面的，然而各个不同平台间是千差万别的，虚拟机只能是提取它们的共性，只有这样才能使得各个平台对用户提供同样的服务，使得用户看起来都是一样的，这样的话，各个平台自己的特点就消失了，相应的各个平台针对特定问题的特长也就发挥不出来了，虚拟机的性能向各个平台中最差的那个看齐了。也就是说性能问题成了第二种做法的一个的问题。这两种的做法的缺点在传统计算环境下，尚可以忍受。到了普适计算环境以后，这两种的做法的缺点就有点让人难以忍受了。

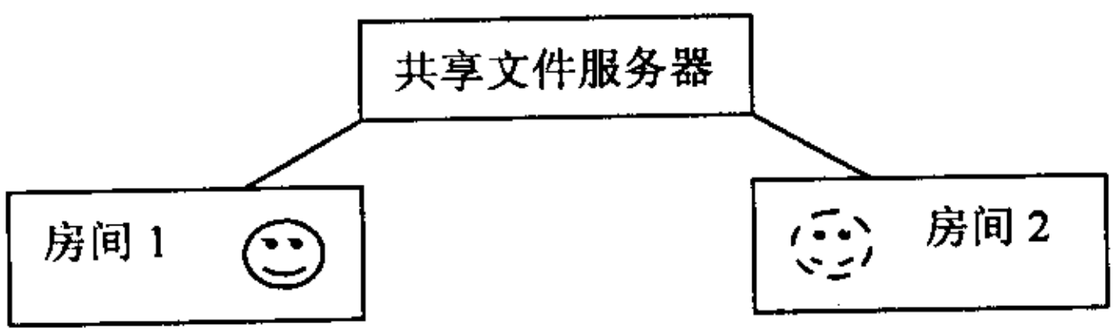
在普适计算环境下，用户的注意力成为一种稀缺的资源，计算资源极大丰富，不存在缺乏的问题，这与过去的情况正好相反。过去的情况是计算资源极度稀缺，相比之下用户的注意力资源问题不算突出。现在情况变了，我们考虑问题的方向也就跟着变了。现在既然用户的注意力成为稀缺的资源了，那么我们就要研究怎么在作业迁移过程中降低对用户注意力的分散，使得用户可以专心于他所要完成的任务，而不是将他的注意力浪费在配置不同平台的环境这些与他所要完成的任务没什么大的关系这些方面的内容上。在第一种做法中，用户要将极大的一部分注意力浪费在对各个不同平台环境的配置的过程中。在传统计算环境下，一个用户占有的计算资源相对来说比较少，可能就是几台台式机，工作站，服务器，这种情况下，配置不同平台环境所浪费的用户的注意力和用户花在完成所要完成的任务上的注意力相比，比例还算比较小，还处于可以接受的范围。到了普适计算的环境以后，一个用户占有的计算资源极大地丰富了，计算不再仅仅局限于台式机、工作站、服务器了，PDA、移动电话等等都可以成为计算资源，这时候如果再是由用户自己去配置不同平台环境，那么配置不同平台所花费的用户的注意力就不再是小比例了，而是占据了用户注意力的绝大部分，用户花在真正要完成任务的注意力反而少的可怜。为了克服些问题，应尽量避免浪费用户的注意力资源。在第二种做法中，各个平台的性能将会极大地浪费。

在传统计算环境下，我们所考虑的平台基本就是台式机、工作站、服务器，这里性能最差的应该是台式机，根据我们上面的讨论，虚拟机的性能是向最差的看齐，这里就是向台式机看齐，性能的差别还不会让我们无法容忍。但是，到了普适计算环境，平台各式各样，从移动电话到服务器，应有尽有。按照虚拟机的性能与最差的等价原则，我

们的虚拟机只具有移动电话的性能。那么使用虚拟机的其他平台的性能就完全浪费了。我们的想法很简单，就是想把两种做法的各自的优点结合起来，而尽量将两种做法的各自的缺点克服掉。我们想通过对任务进行符合一定规则的描述来实现这一点的。

3. 无缝的作业迁移与描述

我们要实现无缝的作业迁移，就要解决怎样在不同的平台间传递一个任务，并让不同的平台协调完成同一个任务，来满足用户的需求。正如引言中所讲的那样，在普适计算环境下，用户的注意力成为了稀缺资源，因此我们在实现作业的无缝迁移过程中不能浪费用户的注意力，这就要求各种不同的平台能够自动识别出各种任务来，而不是让用户自己进行干预，从而耗费用户的注意力。为了在不同的平台间传递任务，并且让不同的平台能够自动识别出各种任务来，我们就必须对任务进行描述，而且这个任务描述必须是规范的，标准的，并且能够跨越不同平台的。任务的描述方案可能有很多种，并且每种可能都不尽相同。我们的想法是利用类似于 HTML 这样的标记语言，来对每个任务进行描述，形成一个描述方案，存储在共享文件服务器中，不同平台都利用这个任务描述方案，通过解析这个任务描述方案来构造出一个任务。由于这样的标记语言具有一定的通用标准，并且格式比较固定，因此就可以用程序实现自动解析，从而不耗费用户的注意力资源。下面以播放一段电影为例来说明。假设有如下这么一个场景，一个人从房间 1 走到房间 2，在房间 1 时，这个人正在看一部电影，他临时有事要到房间 2 去，但是他还想在房间 2 中接着看同一部电影，场景如下图所示。



于是，我们就得将播放电影这个任务从房间 1 迁移到房间 2 去。我们是这样对任务进行描述的。初始情况是他在房间 1 中想看电影，于是当他开始观看电影时，我们就在共享文件服务器上创建这个任务的描述，当监测到他离开房间 1 时，共享文件服务器修改这个任务描述，使得它反应任务完成的情况，当他走到房间 2 后，又请求同一个任务时，就根据这个任务描述来重构出这个任务来，使得这个任务处在他离开房间 1 时的状态，在本场景中，就是从他离开房间 1 时播放到的地方开始播放。具体描述方案如下所示：

1. 初始情况为：

```
<task id=”sample”>
    <type=”playMovie”>
        <file=”a.mpg”>
            <state>
                <length=500/>
                <position=0/>
                <completed=n/>
            </state>
        </file>
    </type>
</task>
```

其中 id 是用来标记这一个任务，type 是用来说明这个任务的类型，file 用来指明与这个任务关联的文件，state 是用来指明当前任务的状态，下面包括三个属性：length，position，completed。Length 是用来指明这个任务有多长，position 是用来指明当前任务完成了多少，completed 用来指明这个任务结束了没有，position 等于 length，那么这个任务可以认为结束了。

2. 离开房间 1 时的情况为：

```
<task id=”sample”>
    <type=”playMovie”>
        <file=”a.mpg”>
            <state>
                <length=500/>
                <position=100/>
                <completed=n/>
            </state>
        </file>
    </type>
</task>
```

当他离开房间 1 时，电影还没播放结束，假设只播放到五分之一，那么当监测到他离开房间 1 后，相应的任务描述方案就被修改成如上所示的样子，其他描述没有变，只是表示当前任务完成状况的 position 被修改成 100 了。

3. 最后的情况为：

```
<task id="sample">
    <type="playMovie">
        <file="a.mpg">
            <state>
                <length=500/>
                <position=500/>
                <completed=y/>
            </state>
        </file>
    </type>
</task>
```

当他到达房间 2 后，请求继续观看电影时，房间 2 中的电影播放机就从共享文件服务器中调出这份任务描述，并根据这份任务描述来重构出这个任务来，使得这个任务的状态与他离开房间 1 时一致，也就是从他离开房间 1 时电影播放到的地方开始接着播放。假设在房间 2 中，电影最后播放结束。那么任务描述就被重置成如上所示的样子，position 被置成 500，completed 被置成 y。然后这个任务描述就会从共享文件服务器中删除。

4. 结论

利用这种描述方案来描述任务，就可以很好地自动地在不同的平台间传递任务。通过解析任务描述，然后将相应的任务与本平台上的对应软件联系起来，从而达到无缝作业迁移的目的。

总之，在普适计算环境下，用户的注意力资源成为一种稀缺资源，普适计算环境下的任何研究都要围绕怎样让有限的用户注意力资源用在他所要完成的任务上面，而尽量避免在与他所要完成的任务无关的方面浪费有限的用户注意力资源。

参考文献：

[1]Sousa,J.P.,Garlan,D., "Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments", Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture 2002, August 25-31, 2002, Montreal.

[2]David Garlan, Daniel P. Siewiorek, Asim Smailagic, and Peter Steenkiste, "Project Aura:Toward Distraction-Free Pervasive Computing", IEEE Pervasive Computing, April-June 2002, P22-31

[3]Guruduth Banavar, James Beck, Eugene Gluzberg, Jonathan Munson, Jeremy Sussman, Deborra Zukowski, "Challenges:An Application Model for Pervasive Computing", Proceedings of the 6th ACM MOBICOM, August, 2000, Boston, MA.

Job Migration Under Pervasive Computing

Bin Li, S.Q. Yang
Department of Computer Science & Technology,Tsinghua University,Beijing,100084,China

Abstract: Job migration is an old problem. Generally there are tow methods to complish job migration. One is that user almost finishes the job migration by hand. He must synchronize the job among all the platforms and configure all the platforms. This method is good in using the performance of all the platforms. But it requires user to have so much ability and it will waste too much the abstraction of the user. The second method is using Virtual Machine. By defining a VM, the difference among all the platforms dissappears. But it will waste the performance of the platforms. The problems of the two methods are not serious if we are not in the pervasive computing environment. But unfortunately, we are going toward pervasive computing world. These problems become serious. We must consider them now. We attempts to discuss them here.

Key Word: Description Schemes; Job Migration; Pervasive Computing.