目录

- 文档属性

属性	内容
项目名称：	Taobao ABS(AutoBuildSystem)开源
文档主标题：	使用软件包管理大规模应用
文档副标题：	
文档编号：	
文档版本号：	1.0
版本日期：	2010 年 10 月 20 日
文档状态：	Active
文档提供：	Taobao
文档作者：	BuPing<buping@taobao.com>

- 文档变更过程

版本	修正日期	修正人	描述
1.0	2010 年 10 月 20 日	不平	新文件

1.前言

当应用软件构建的服务开始庞大时，头疼的事接踵而至：如何管理应用软件本身的良莠不齐？如何保证已开发完毕的产品不会丢失？如何进行各类应用的部署 和拆分？如何平顺的对公用组件更新和升级？如何管理因网络、操作系统、等环境因素不尽相同导致的如同蛛网般的复杂配置、如何在日常运作中对应用服务达到快 速建立、恢复、回滚？如何对系统运作进行日志跟踪？在工作成员频繁变动、业务频繁拆分的情况下上述工作的难度会乘方级的增加，嵌套问题会让构建和维护人员 不知所措，对维护人员的要求会不断提升，在运作成本本应最低的维护一环结果投入了不合情理的力量，当问题严重时更会拖累新业务和应用的开展和影响已有的服 务质量和应用成果。

我们把这些难题定位为是依赖于技术手段的管理工作，接下来会向大家介绍如何采取合适的技术手段对庞大的应用软件和服务进行高效管理。

2.来自日常维护的需求

下面列一些日常维护工作所必须的基本需求：能够准确的获取软件；将软件正确地安装在操作系统上；能从操作系统整洁的卸载软件而不伤害操作系统和其他 软件；能够对已安装的软件进行升级和降级；能有一个依赖与关联的校验机制来对软件和软件之间的关联进行处理和解决文件缺失、属性错误、文件冲突、重复等问 题；能够有一个中央信息库来进行软件信息的检索；能从系统中的文件反查出隶属的软件；能对软件的完整性进行校验发现更改和变动；软件来源的合法性校验；能 够有效的区分系统软件和应用软件。

3.技术方案选择

化零为整、分门别类是管理数量庞大事物的基本原则，版本控制是跟踪历史和存档的主要手段，编织依赖是整体结构的搭配和安排，因此针对应用软件我们想 要的核心功能可以总结为：封装(Packaging)、类聚(Grouping)、版本控制(Versioning)、依赖处理(Dependence handling) 。

次级功能的需求可以列举为：可检索、有充分的描述信息、标准化、便于装卸、支持多种环境、可自定制配置、便于分发、可微调、可重塑等。

提供类似功能的技术解决方案有：压缩包、安装程序、包管理系统，下表是功能比较一览

由此可见使用包管理系统进行管理是满足我们核心需求的最佳选择。

3.1 软件包管理器

我们接触的最多的包管理器恐怕就是RPM软件包管理器了，大部分企业、公司、小团体都使用了RedHat Linux操作系统发行版，各大软硬件厂商的系统程序、驱动程序、软件系统、

数据库软件几乎都会有RPM的软件包发行，数量庞大几乎占据了90%以上的能 下载到的Linux软件，以至于其他Linux发行版不得不支持安装RPM软件的安装。对于需要大规模管理

应用软件部署的情景来说，用一个软件包管理器统 一管理操作系统上文件管理是理想且必须的。

RPM在业内使用如此广泛、市场占有率之高是有原因的，我们简单了解一下RPM的历史就知道为什么各大公司和厂商会把RPM作为包管理不二的选择了。

RPM的发展历史和GNU/Linux的发展形影不离，我们使用Linux操作系统进行各种工作必然要下载、安装、配置各种软件，这些软件 的获取原始方式是通过软盘拷贝、光盘安装等现在看来很低效的方式，随着Internet的普及，获取数据的速度和途径变得异常便捷，下载软件几乎是 Internet的使用者必做的第一件事。在中国、台湾、香港等华语地区估计有99%比例的人最先接触的Linux操作系统发行版都是一个名为红帽子的 Linux发行商所发行的Linux系统，最先了解的命令前几条必然有rpm这条命令。很多人认为rpm是安装调试系统的基础命令，在功能定位上我们应该 给一个学术名词——软件包管理器， 我们可以从Windows的资源管理器来理解这个名词，资源管理器是负责管理Windows操作系统上的所有文件的管理工具，这些文件没有任何关联，使用 者很难知道哪些文件删除了会对操作系统造成不可挽救的影响，Windows在控制面板中提供一个添加/删除程序的管理工具来管理程序，用户文件和系统文件 只能用系统目录和用户目录（甚至分区）来区分了。我们可以看出这并不能解决根本问题，如果用户把应用软件、系统软件安装到其他文件夹或目录，局面就会变得 越发尴尬，仅指望软件安装程序提供的卸载、升级、配置机制经常会出现各类问题：安装程序无行为规范，会经常导致残缺遗留情况；恶意软件会不遵守保护系统的 准则做违反用户本意的事情。统一就交付一个可信任的管理器管理是应对无序和混乱的唯一出路。Linux不同于Windows，是一个完全自由的操作系统， 意向是由用户能够控制操作系统上的一切，红帽子RedHat作为早期的Linux发行版提供商研制了一个解决方案——采用化零为整的方式 把软件文件封装成一个软件包，把软件包的信息、所包含的文件记录下来，然后通过程序工具统一进行管理。当然当时的其他的Linux发行商和先行者也有类似 效仿做法，但都没有RPM把包管理会遇到的问题考虑的如此广泛和深入。

令人瞠目结舌的事情发生得如此突然，1995年成立的红帽子一下成为了早期最流行的Linux发行版（至今也是），超越了早其2年Linux发行版的先行者Slackware，其中一个重要的因素就是由于RPM的及时出炉。

RPM并不是横空出世，当今的版本也经历了历次换代才得到了大型软件、硬件厂商的认可和接纳。

RPM最早的版本叫做RPP（RedHat Package Processor），在RedHat的第一个Linux发行版上使用，RPP提供了方便的安装卸载软件包命令、Script嵌入、软件包校验、强大的软 件包信息检索查询等基础核心功能。但保存源码、重编译发行和多平台架构的高级功能并没有实现。

使用源码分析是一个深入解决问题的终极手段，二进制程序出现问题归根结底还要到源码层面去找原因，有了程序源码就可以按照实机的环境和需求 进行编译，按照实际文件路径修改源码，按照使用者意愿处理程序行为甚至重新编写和改造。如果实现了保存源码一起发行的理念会让包管理器处于王者地位。

在红帽子忙于折腾它的RPP时，Linux狂热者Rik Faith带着他的团队攻下了包管理器的保存源代码难题，他们的包管理器叫做PMS（Package Manager System）。PMS实现了单项攻关，软件包信息检索查询等功能远逊色于RPP。

很快PMS团队的核心成员Rik Faith和Doug Hoffman干脆直接加入了红帽公司与红帽工程师将PM和RPP整合在了一起，于是诞生了PM，红帽手中于是有了2个包管理器，虽然功能互补，但还是由 于系统文件管理的唯一性因素需要统一。于是Marc Ewing和Erik Troan参仿RPP、PM、PMS的优点开始塑造新的包管理器程序RPM Version 1，从大体上，RPP和PM的缺点被弥补了。

RPM Version 1新带来的优势也很惹眼，如可以自动处理编译时的configure文件，如修改configure值等，这样对升级、安装、删除可以获得比较好的管理和 控制。从某种意义上说这个灵活而强大的功能树立了RPM在包管理器软件中的强势地位；轻松快速的构建方式可以让开发者们创建大量的RPM包；在RPP、 PM基础上酝酿出的更加友好的易用性。

另一方面RPM Version 1仍然存在其他的令人不够爽的缺陷，如数据库设计脆弱、执行速度不够快、程序体积庞大等（RPM Version 1是Perl编写的），多平台的功能仍未能被支持，包文件格式的延展性很差，很难进行功能的添加。

面对这些问题，Marc Ewing和Erik Troan用c语言把整个程序重写了一遍，执行速度和大小问题最终得到了解决，摆脱了沉重的Perl依赖；包管理数据库得到了重新设计，为了使RPM使用 更加广泛，他们创建了rpmlib以供其他程序调用RPM的功能；多平台架构问题也通过重构方式得到解决；文件格式的扩展性也得到了提高。这个直到今天广 泛使用的包管理在Red Hat内部叫做RPM Version 2，也就是大家看到的RPM。

包管理器的发展和操作系统紧密相连，我们可以从软件包的管理透视到操作系统的管理。

3.2 软件仓库

软件包仓库是存储、发行软件包的集散地，主要功能是对软件进行检索、分类、管理，为用户提供检索、查询、下载等功能

在Internet上出现的比较早的软件仓库模式应该数CPAN，这种综合档案网和程序库的方式让使用者容易寻找、下载、安装、更新及管理 他们的软件，之后被有很多程序语言、操作系统发行版效仿，像程序语言PHP的PEAR，Java的Maven，Python的 EasyInstall，Ruby的 RubyGems，操作系统RPM的YUM，Ubuntu的AptTools，FreeBSD的Ports， SuSE的Zypper，Solaris的OpenCSW。

软件仓库发展是随internet普及而来，各大厂商都会有自己的软件仓库管理手段，我们能了解和下载到的是一些开源社区提供的解决方案， 企业级的软件仓库管理会涉及到用户权限、分组、分支等细化管理功能。我们选择Yum是因为提供了基本的检索和分类功能，在辅助手段帮助下也可达到类似于 Yahoo!这样全球化体系的软件仓库管理需求。

4.软件管理策略

4.1 封装 RPM 软件包及分类

封装技术细节请阅读另一篇文档《使用RPM封装软件包》，此段落只谈论封装策略和封

装原则。

RPM会分为2大类（第三类Delta包和debuginfo包本文暂不讨论），二进制包和源码包，二进制包是真正可以放到系统上安装并使 用的软件包，但会受操作系统架构和版本限制；源码包是包含未编译的程序源代码以及封装软件包指令文件——SPEC文件的特殊RPM包，安装后只是在 SOURCES目录存放源代码，不能得到可用的程序，用于根据服务器实机架构进行重新编译和生成相对应的RPM包。

我们根据常见的情况把目录如此划分：

```
|-- 4
| |-- SRPMS
| | |-- current
| | |-- stable
| | `-- test
| |-- i386
| | |-- current
| | |-- stable
| | `-- test
| |-- noarch
| | |-- current
| | |-- stable
| | `-- test
| `-- x86_64
| |-- current
| |-- stable
| `-- test
`-- 5
|-- SRPMS
| |-- current
| |-- stable
| `-- test
|-- i386
| |-- current
| |-- stable
| `-- test
|-- noarch
```

```
| |-- current
| |-- stable
| `-- test
`-- x86_64
|-- current
|-- stable
`-- test
```

我们会把RedHat Enterprise Linux的大版本分为4和5两大分支，每个大版本下会根据架构分为i386、x86_64、noarch以及源码包SRPMS，再根据生产和测试的要求 把软件分为test、current、stable三个分支，举个例子来说如果是用c等语言程序编写需要根据计算机架构编译的应用软件程序就需要在相应的 分支和架构下封装软件包，而noarch的应用软件则只关注操作系统的大版本即可，通常php、java这种解释型语言和与系统架构无关的语言应用程序可 以在任意服务器上运行，可以享受一次编写到处运行的便利。追求效率的C语言程序就需要多费一番经历把程序封装成相对应的二进制包。需注意的是：同一个版本 的软件包传到对应的大版本-系统架构即可，test/current/stable是用于分割软件包适用场景的，同一个软件包可以通过客户端管理工具在三 个场景分支进行转移。

4.2 基础软件与公用软件

可以说整个RedHat操作系统都是由RPM包构成的，鉴于系统文件管理的唯一性，RedHat很自然的将所有组件都打成了RPM包进行管理和发 行，核心组件近千个，部分核心组件是操作系统的基本软件，一旦卸载整个操作系统将无法运行。随着应用软件对系统核心的渗透，越来越多的程序和软件都希望成 为基础软件和公用软件，RedHat也是逐步纳入这些需要被依赖的底层软件包。我们进行管理时也会把整个全局的应用中最核心最有动力的软件逐步纳入基础软 件管理，让强劲的软件发挥最大的效力。

4.3 软件包命名

RPM包的命名结构如下
软件包名称-版本号-发行版本号.架构.rpm
例： php-devel-5.2.9-3.el5.x86_64.rpm
- php-devel是软件包名称
- 5.2.9是版本号 5是主版本 2是次版本 9是patch版本
- 3.el5是发行版本号 el5意为在RedHat Enterprise Linux 5下使用的RPM

4.4 颗粒度划分与垂直化疏导

同操作系统核心组件和基础公用软件一样，自开发应用也会封装成RPM包的方式，自开发软件包的命名会和系统提供软件名称区分开，如taobao- xxx应用、taobao-xxx-mod-A、taobao-xxx-mod-B等，颗粒度可以看作应用软件架构设计延展性优良程度的衡量标准，软件模 块化和好的扩展性会使软件更易于维护和继续发展。

垂直化是一个从全局层面的解决方案和发展方向，意在削弱应用与应用之间耦合度，应用与应用之间的互动关系应该趋于契约化、协议化，避免应用 联动性过大而导致出的连锁效应从而降低问题的影响范围。以垂直化为导向后，业务的调整、拆分、整合将有一个优质的起点，对观察现状和整体决策起到至关重要 的作用。

4.5 连接起开发与生产的桥梁

在推行RPM封装软件的过程中遇到比较大的困难就是大部分开发人员对系统软件包和软件包管理方面的知识知之甚少，尽管操作比较简单，但仍需相当的学 习成本，在开发人员众多的时候很容易把RPM打包学习和培训的问题放大。红帽内部有个大规模的RPM生产线项目叫Koji，专门负责从源码编译出各种架构 的二进制包，流水线作业批量制造RPM；从而使得开发人员无需上编译机即可生产获得最终的RPM包，淘宝使用了集成构建软件Hudson来进行编译和构建 最终的RPM软件包，使得开发人员的成果可以更加容易的变为可用的软件包。

5.结语

我们用软件包管理解决应对了日渐庞大的应用和服务增长问题，并从中学习到了更多规模化运作、系统管理和集成构建方面的经验，获得了极强的业务恢复和 重塑能力，提高了整个运维的生产力，最大范围上保存了开发的劳动成果，使得业务更加便于调整和改进，对整体应用架构有了实质性的描绘，也希望本文对读者有 所帮助。