

构建一个分布式操作系统的简单方案—答陈硕的“分布式系统中的进程标识”一文

by 沈东良 <http://blog.csdn.net/shendl/>

2011.03.29

简介

本文提出了构建一个分布式操作系统的简单方案。核心是使用 ssh 的强大能力，构建一个简单、安全、灵活、高效的分布式操作系统！并且没有单点失效的问题。注意，是通用的分布式操作系统，而不是专用的分布式系统。

对“分布式系统中的进程标识”一文的疑问

刚才看到陈硕先生的一篇 blog:“分布式系统中的进程标识”，地址：
<http://www.cnblogs.com/Solstice/archive/2011/03/29/1998412.html>

我不同意该文的观点，因此在这里抛砖引玉，提出一个构建一个分布式系统的简单方案。

文中说：“正确做法：以四元组 ip:port:start_time:pid 作为分布式系统中进程的 gpid，其中 start_time 是 64-bit 整数，表示进程的启动时刻。”

第一个问题：没有 **port** 的进程怎么命名？

文中这样回答：“根据陈硕在《[分布式系统的工程化开发方法](#)》一文中的观点“在程序里内置 http 服务器”，分布式系统中的每个进程都应该提供一个管理接口，对外提供一个维修探查通道，可以查看进程的全部状态。这个管理接口就是一个 TCP server，它会侦听某个 port。”

1，cat，uuid，utime 这些程序都要提供一个内嵌的 http 服务器？有必要吗？第三方程序你怎么该？

2，我至今没有见过一个操作系统自动给所有进程提供 http 服务的。如果真的应该在每一个程序里内置 http 服务，那么肯定有操作系统会提供这种服务。

3，如果程序提供 http 服务，那么程序的安全就会有疑问。http 通讯很容易被拦截。http 服务如果开着，那么攻击方可以不停测试用户名、密码，从而攻进操作系统。成为安全隐患。

第二个问题：**time** 作为唯一的区分是否可靠？

如果操作系统的时间发生调整，那么有可能 time 会回到从前，从而可能导致多个进程有一个进程号。可能性虽小，但不能排除。

构建一个分布式操作系统的简单方案

给进程唯一命名

陈硕的“分布式系统中的进程标识”一文讲的就是怎样给进程唯一命名的问题。前一节中，我已经证明了该文提出的方法是不正确的。现在讲一下我怎么解决这个问题。

很简单，执行 uuid 命令，就会返回一个全球唯一的字符串。这已经在理论上证明是不可能重复的。Uuid 已经在数据库,.NET 等众多应用程序中广发使用。

我们可以这样建立一个结构体，来标示一个进程：

```
struct Process{
    char[32] uuid,
    int ip, //(其实计算机也应该定义一个 struct，使用 uuid 唯一标示。这里为了说明简单，一切从简了)
    int pid,
    char * program,
    int valid //进程是否还存在
};
```

然后建立一个 hashtable: (char * programName, struct Process* processes)

这样的 key, value 结构。

这样就可以找到一个程序的所有进程。 还可以定义多个类似的 List 和 Hashtable，方便管理和查询。

使用 **ssh** 实现远程调用

使用 ssh 可以实现远程调用任意远程计算机上的程序和资源。ssh 就是一个安全的 shell 程序，通过 ssl 安全协议访问远程计算机。

因此，使用 ssh 就可以构建一个简单、安全、有效的分布式计算系统！

使用 SSH 的优点还有：

- 1，不再需要程序提供远程访问的途径。如陈硕的方案就是需要每一个程序都内嵌一个 http 服务器。ssh 不需要。而且它可以访问远程计算机上的所有程序。
- 2，提供了安全访问的机制。Ssh 建立在 ssl 上非常安全。
- 3，提供了授权的机制。ssh 使用操作系统的用户登录。这样就可以使用操作系统本身的用户授权实现任意的授权。ssh 用户可以有自己的 home 文件夹，可以设置用户使用的内存数，硬盘数，可以创建的进程数量等等。不允许访问其他文件，不允许运行其他用户的程序等等诸多优点。

通过提供一个包括：计算机，用户，程序-，进程等对象的数据库管理系统，就可以使用 **ssh** 实现一个简单、安全、灵活、高效的分布式计算系统！

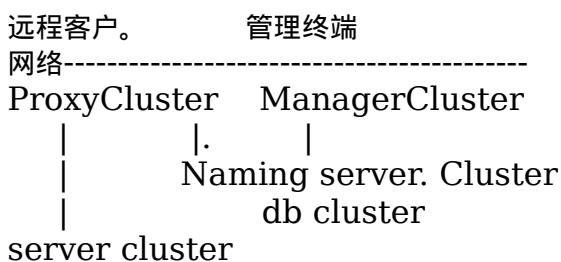
架构设计

我给这个想法起个名字 sshDos，如果有时间，把它实现出来。

sshDos,不是用于 dos 操作系统的软件，Dos 是分布式操作系统的缩写。sshDos 指的是使用 ssh 技术构建的分布式操作系统。

使用 ssh 构建分布式操作系统这个想法始于去年的某一天的顿悟。后来一直都没有实施。惭愧！

架构图



上图中网络内的部分就是 sshDos 分布式操作系统。

主要分为三个部分：

- 1，命名服务器部分，这是整个系统的大脑。它保存了 proxyCluster 和 serverCluster 中所有服务器的信息。

服务器和代理必须向命名服务器注册才能使用。

命名服务器的数据保存在数据库集群中。

命名服务器是无状态的服务器集群。可以使用 memcache 集群缓存数据，提高性能。

命名服务器提供类 rest 的 openapi 向外提供服务。

ManagerCluster 是 web 形势的管理服务器，使用命名服务器提供的服务。管理员可以通过它检测运行状况，注册服务器和代理，取消注册，授权访问等等。

2, ProxyCluster, 它们接受远程客户请求, 并向 serverCluster 发送命令的服务器。ProxyCluster 使用命名服务器提供的 api 服务, 查找合适的 server。然后使用 ssh 向 server 发起请求。

3, serverCluster, 这是真正执行工作的集群。它们都安装了 ssh 服务器, 从而 proxy 和 naming server 可以访问它。

Naming Server 对他执行配置, 管理和监控等任务。

Proxy。向它发起工作负载。

此架构设计的优点:

1, 整个系统的所有模块都是集群, 可以热插拔任意节点, 没有单点失效的问题。

2, 使用 ssh--安全的 shell, 因此可以向 server 发起任何命令。

因此, sshDos 是一个通用的分布式操作系统。

他不是单纯的分布式 map-reduce 系统, 不是超级计算机, 不是分布式存储系统。但它也同时可以是它们全部。

只需要编写几个 shell 脚本, 或者程序, 然后发送给 proxy, 它就可以实现任意分布式功能。

3, 可以使用 linux 的用户授权机制, 控制不同用户使用的资源和权限, 实现细粒度的管理。

使用 sshDos 的例子

我是做虚拟化的, 就以虚拟化为例, 说明 sshDos 分布式操作系统在虚拟化方面的使用例子。

虚拟机集群管理系统

需求

比如说, 我们有一个虚拟化的服务器集群, 需要对它进行管理。

这样, 在 sshDos 中, worker Cluster 就是一个个 Host, 每一个 Host 内可以同时启动很多虚拟机。

这些虚拟机供外部用户通过 ssh, vnc 等方式使用。假设我们使用 kvm, 当然使用任何其他 hypervisor 对于 sshDos 来说都是一样的。

因为我们这个是虚拟机集群, 我假设我们使用了统一的分布式文件系统。

如 SAN, HFS, Ceph 等分布式文件系统。这样, 我们不需要把 vm 的 img 从一个 host 迁移到另一个 Host 上。只需要迁移 kvm 命令行 (死迁移) 和内存状态 (活迁移)。

实现

worker Cluster 就是一个个 Host。通过 Naming Server 提供的 Web 形式的注册接口, 我们可以向 sshDos 分布式操作系统中添加和移除 Host Server。

Naming Server 通过 ssh 访问每一 Host。然后定时执行 Host 上的 top 程序, 查看系统的 CPU, 内存使用率, 查看每一个 CPU 的使用率, 从而获得每一个 Host 的实时压力, 存放到数据库和 memcache 中。kvm 虚拟机在 Host 内就是一个进程。通过查看 kvm 进程的资源使用率, 我们就知道了 kvm 虚拟机的资源占用情况。

远程管理程序通过 proxy server 发起要求启动一个 vm 的命令。Proxy 到 Naming Server

上查询应该使用哪一个 Host Server。

通过编写一个程序，我们可以知道 vm 内部每一个进程的资源使用情况。编写一个 VM 内部运行的监控程序，然后通过 kvm 模拟的串口等设备发送到 Host 上。

这样 Naming Server 就可以通过 ssh 得到 VM 内部各个进程的运行情况。知道 VM 的操作系统是否运行正常，各个服务是否运行正常。

Vm 内的一些服务是非常关键的，如 sshServer，还有用户启动的服务器，如 apache, tomcat 等。Naming Server 都可以知道的一清二楚。

你还可以使用 **sshDos** 分布式操作系统满足你的任意需求，因此 **sshDos** 是一个通用的分布式操作系统，而不是专用的分布式系统。