Subject: Re: [ANNOUNCE] first stable release of OpenVZ kernel virtualization solution
Posted by dev on Tue, 06 Dec 2005 15:39:16 GMT
View Forum Message <> Reply to Message

Hello Andrew,

>>We are happy to announce the release of a stable version of the OpenVZ
>>software, located at http://openvz.org/.
>>
>>OpenVZ is a kernel virtualization solution which can be considered as a
>>natural step in the OS kernel evolution: after multiuser and
>>multitasking functionality there comes an OpenVZ feature of having
>>multiple environments.
>
>
> Are you able to give us a high-level overview of how it actually is
> implemented? IOW: what does the patch do?
Will be glad to do so.

On the high-level the system looks like patched Linux Kernel with a
number of user space tools. The kernel itself boots on a usual Linux
distribution like RHEL4 and works as usual. But there are extensions
which allow to create a new VPS context.
User space OpenVZ tools use these extensions to do the following, e.g.
on VPS start:
- turn on and configure quota on VPS file system subtree.
- chroot to this filesystem tree.
- create a UBC context with configured resource limits/guarantees.
- create a VPS context and exec init in this newly created environment.
- newly spawned init executes VPS initscripts as if it was a usual Linux
box which has switched power on.

There are two patches that can be found on OpenVZ site.
patch-022stabXXX-combined which is a single consolidated OpenVZ patch
including driver and mainstream updates. And the more interesting one is
patch-022stabXXX-core (available from the SRC RPM), which itself
consists of the following parts:
- mainstream fixes and security fixes
- 4GB split
- User beancounters (kernel/ub/*, include/ub/*). This includes
accounting and limiting/guarantees of resources.
- Virtualization itself (ve_struct, kernel/vecalls.c - main code for VPS
start/stop, net/ipv4 - virtualization of TPC/IP and netfilters,
drivers/net/venet* - virtual network device for VPS, virtual pids, etc.)
- fs/simfs - simple filesystem to fake VPS and return correct values on
`df` and statfs() output.
- fs/vzdq* - 2-level disk quota.

- kernel/fairsched.c and kernel/sched.c - fair CPU scheduler (SFQ-like scheduler with shares and hardlimits)

We strongly believe that Linux kernel can benefit from Virtualization and Resource Management very much, so are very interested in your and Linus comments on this. Though virtualization and resource management is not a full feature set provided by OpenVZ, they can be used for creation of more secure environments where untrusted users are involved. For example, with virtualization it is possible to isolate set of processes on the node, web server or other application, thus vulnerabilities in this application won't allow to destroy the whole system (e.g. mail server running on the same node), install some backdoors/trojans (in kernel modules) etc. Resource control can be used for the same purposes when deliberate DoS or bugs in applications should not crash the whole system down. Virtualization also allows to do simple backups/restore/migration of VPSs between nodes, thus making maintenance much easier. So the I would summary up:
- Virtualization helps to isolate service, improves managebility
- Resource Management allows to control resources and prevent DoS from utrusted users in multi-user environments. User beancounters can be used for usual users instead of VPSs.

Main user space tools for OpenVZ are:
- vzctl, which is used for most of high-level VPS operations like VPS creation, start/stop, destroy, configuring, setting UBC and other parameters etc.
```
# vzctl create VPSID
```
is used to create VPS.

After VPS creation it can be started via issuing:
```
# vzctl start VPSID
```

To see set of processes inside VPS one can execute:
```
# vzctl exec VPSID ps axf
```

And the most interesting command is 'enter' which allows to get to VPS (to 'enter') from host system via changing context to VPS one:
```
# vzctl enter VPSID
bash#
```

- vzquota, a tool used for 2level quota support. Allows to turn quota on/off, recalculate it etc.

- vzpkg, a set tool of tools allowing to easily manage VPS templates (redhat, centos, fedora etc.).

>>As virtualization solution OpenVZ makes it possible to do the same
>>things for which people use UML, Xen, QEmu or VMware, but there are

>>differences:
>>(a) there is no ability to run other operating systems
>>   (although different Linux distros can happily coexist);
>>(b) performance loss is negligible due to absense of any kind of
>>   emulation;
>>(c) resource utilization is much better.
>
> What are OpenVZ's disadvantages wrt the above?

disadvantages:
- unable to run Windows or xBSD on OpenVZ Linux.
- VPS owner can't load/use custom kernel modules
- theoretically stability of such solution has one single point of
failure - the kernel. This is mitigated by lots of auto (stress) tests
done by us to be sure the kernel is stable (we have more than >200
mainstream patches due to this). The stability is the main goal here,
since servers running multiple VPSs work under much higher load and the
cost of kernel oops is much higher.
- in some respects dealing with files and processes is harder than with
VM which is bounded to it's memory, CPU and disk state, e.g. when doing
backups. However, there are many cases when this is rather an advantage,
e.g. when you are able to accesses VPS files and processes from the host
system and do some management actions when VPS itself is stuck or
performs poorly and no remote access is available.

>>The dynamic assignment of resources in OpenVZ can significantly improve
>>their utilization. For example, a x86_64 box (2.8 GHz Celeron D, 1GB
>>RAM) is capable to run 100 VPSs with a fairly high performance (VPSs
>>were serving http requests for 4.2Kb static pages at an overall rate of
>>more than 80,000 req/min). Each VPS (running CentOS 4 x86_64) had the
>>following set of processes:
[skipped]

> Do the various kernel instances share httpd text pages?
Please note, there is a single kernel instance for all VPSs! This means
commong page cache etc.

But in this particular example each VPS has it's own file tree and it's
own httpd instance in memory with full set of pages (.text, .data, ...),
i.e. there were running 100 VPSs without any page sharing involved.

"User Beancounters" accounnting is implemented with page sharing in mind
and will be correct in this case (i.e. only page fraction will be
charged). With page sharing this example scales well up to 200+ VPSs.
And sharing itself can be achieved in multiple ways: common part of file
system tree (e.g. /lib, /usr, etc.) or via a special filesystem.

Kirill