Iniciar apresentação

Aula 01 - Apresentação e introdução

Sobre esta aula

- Apresentar o programa do curso.
- Introduzir os conceitos básicos e fundamentos de virtualização.
- Histórico da virtualização.
- Questões de projeto de ambientes virtualizados.

Sobre o curso

1. Participação colaborativa de todos (Wiki, Email).
2. Forte conhecimento de sistemas operacionais.
3. Foco em soluções livres, mais especificamente KVM. Porém conceitos aprendidos podem ser transportados para outras soluções.
4. Foco em virtualização de servidores e ambientes de pequeno a médio porte.
5. Não temos como objetivo construir um datacenter virtualizado.
6. Calendário.
7. Avaliação.

Introdução a virtualização - O que é virtualização?

- Particionamento de recursos, dividindo uma unica máquina física em múltiplas "maquinas lógicas", mais conhecidas como máquinas virtuais.
- Cada máquina virtual pode rodar um sistema operacional e seus aplicativos de maneira independente e isolada.
- Hypervisor, ou Virtual Machine Monitor (VMM) é a camada de software entre o hardware e as máquinas virtuais.

> **É Hypervisor ou VMM?**
>
> Hypervisor e Virtual Machine Monitor (VMM) hoje em dia são a mesma coisa.

Histórico

- A virtualização foi implementada há mais de 30 anos atrás pela IBM como uma forma de particionar de maneira lógica os computadores de mainframe em máquinas virtuais separadas.
- Essas partições permitiam que os mainframes assumissem múltiplas tarefas, ou seja, que

executassem vários aplicativos e processos ao mesmo tempo.
- Como os mainframes eram muito caros, eles foram desenvolvidos para serem particionados, como uma maneira de aproveitar completamente o investimento.

Histórico

- Devido ao alto custo para aquisição de um mainframe, empresas passaram a adquirir servidores de plataforma x86 de acordo com a demanda, processo este chamado de low-end (várias máquinas pequenas fazendo o trabalho de um grande servidor).
- Neste cenário, ao invés de ter um alto custo inicial com a aquisição de um mainframe, optava-se por adquirir servidores menores de acordo com a necessidade.

Histórico

- Com o passar dos anos a virtualização começou a cair no esquecimento devido a criação de novas aplicações cliente/servidor e ao declínio da plataforma mainframe que perdeu força frente a ascensão da plataforma x86 pela década de 80 e 90.
- Ao contrário dos mainframes, as máquinas x86 não foram desenvolvidas para aceitar a virtualização total, e existiam desafios incríveis para se criar máquinas virtuais a partir de computadores x86.
- Os servidores eram superdimensionados para a aplicação que iriam executar, e por conseqüência, acabavam por sofrer do mesmo problema dos mainframes da década de 1960, isto é, não se aproveitava toda sua capacidade computacional, tornando-se subutilizados.

Histórico

- Então, em 1999, a VMWare Inc. introduziu o conceito de virtualização na plataforma x86 como uma maneira mais eficiente para utilizar o equipamento desta plataforma, aproveitando servidores x86 para fornecer uma estrutura computacional que possibilitasse o total aproveitamento dos recursos computacionais destes servidores.
- A partir de 2005 fabricantes de processadores como Intel e AMD deram mais atenção a necessidade de melhorar o suporte via hardware em seus produtos. A Intel com sua tecnologia Intel VT e a AMD com a AMD-V.

Introdução a virtualização - Hypervisor

- Gerencia o acesso das máquinas virtuais à recursos.
- Dá a cada máquina virtual a ilusão de que ela está sendo executada em um hardware próprio.
- Executa diretamente em cima do hardware da máquina física, também chamado de *bare metal*.

Introdução a virtualização - Mais jargões

- Máquina virtual: guest.
- Máquina física: host.

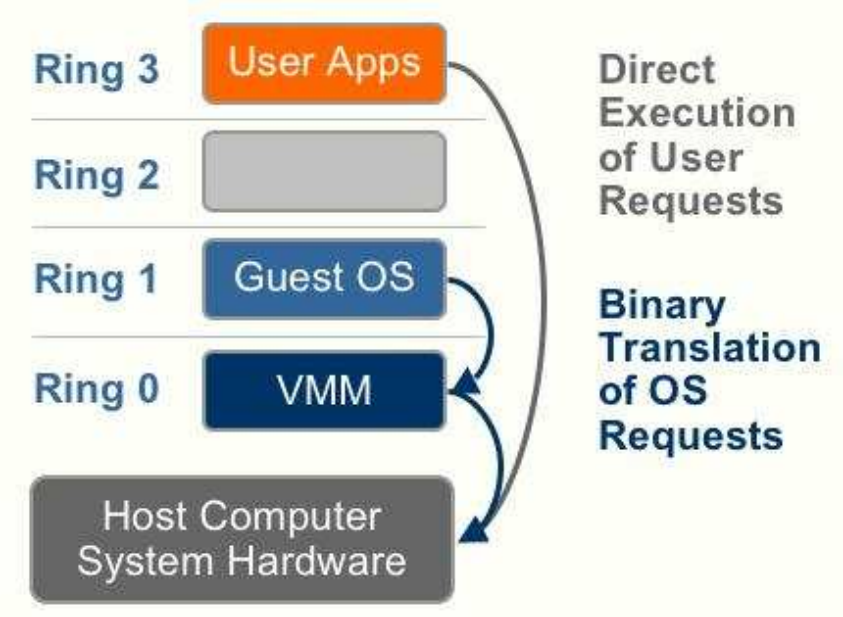
Técnicas de virtualização - Completa/Full

- É uma camada de software que simula **todos** os dispositivos de hardware de um sistema.
- Processo conhecido como **emulação**.
- Exemplo: computadores, video games, calculadoras, celulares.

Técnicas de virtualização - Completa/Full

- Hardware simulado 100% em software.
  - Grande perda de performance.
- Evolução: hardware real (mais especificamente CPU) exposto para a máquina virtual.
  - Interceptação de instruções privilegiadas (ring 0) e reescrita das mesmas (binary patching).
  - Instruções não privilegiadas podem ser executadas diretamente são simplesmente repassadas pelo VMM (hypervisor).

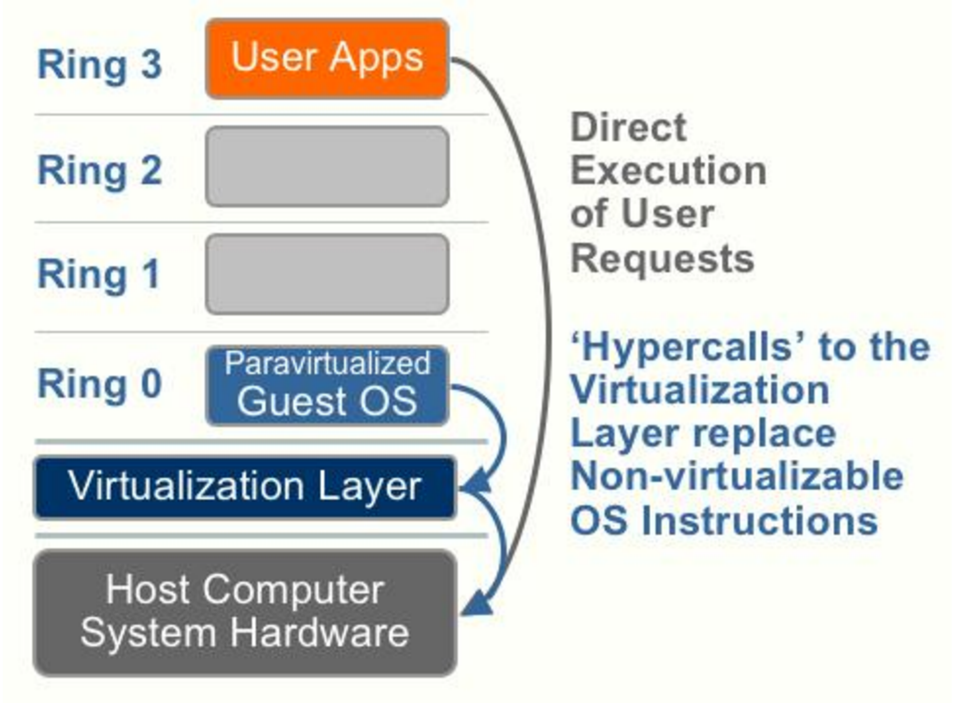
Técnicas de virtualização - Completa/Full



Técnicas de virtualização - Paravirtualização

- A máquina virtual roda sobre o hypervisor e comunica-se diretamente, resultando em melhor performance.
- É necessário mudanças nos Sistemas Operacionais estes passem a usar instruções especiais em detrimento das instruções de máquina padrão.

Técnicas de virtualização - Paravirtualização

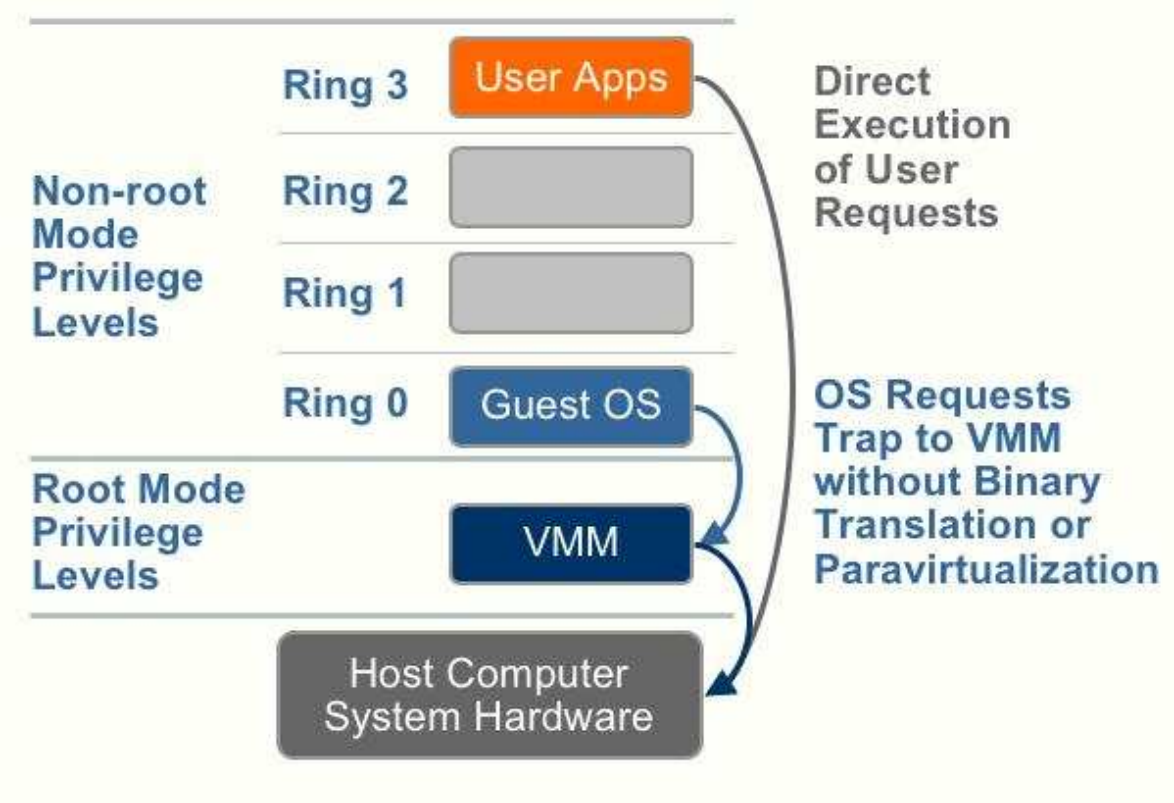


Técnicas de virtualização - Assistência do Hardware

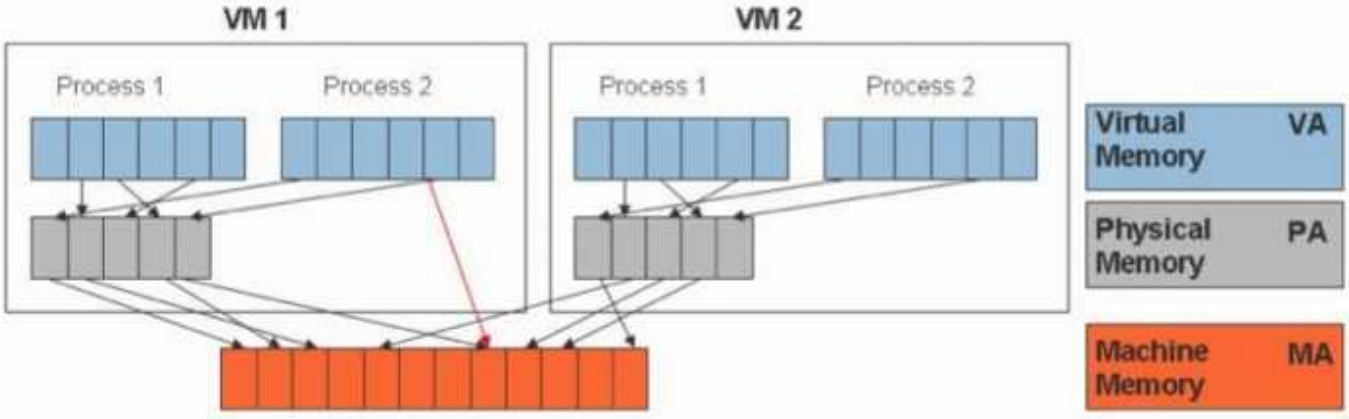
- Processador e no controlador de memória permitem que o hardware rode simultaneamente mais de um sistema operacional.
- Intel-VT e AMD-V
- Privileged and sensitive calls are set to automatically trap to the hypervisor, removing the need for either binary translation or paravirtualization.
- O estado da máquina virtual é armazenado em estruturas de dados que ficam dentro do processador.
    - Virtual Machine Control Structures (VT-x)
    - Virtual Machine Control Blocks (AMD-V).

> Para saber mais sobre virtualização em hardware x86:
>
> Fisher-Ogden, John. "Hardware Support for Efficient Virtualization"
>
> http://www.cse.ucsd.edu/~jfisherogden/hardwareVirt.pdf

Técnicas de virtualização - Assistência do Hardware



Técnicas de virtualização - Assistência do Hardware



Por que virtualizar? Problemas atuais

- Baixa utilização média dos recursos computacionais (CPU, memória, armazenamento e rede).
  - Carga por volta de 2% a 15%.
- Aplicações amarradas ao sistema operacional, dificultando atualizações, manutenções e testes.
  - Questões de legado.
- Demora na disponibilização de novos ambientes operacionais.
  - Alocação de um novo servidor (ativação de ponto de rede, no-break, espaço no rack)

Por que virtualizar? Problemas atuais

- Gastos com energia e refrigeração.
- Acumulo de servidores e custo de mão de obra.
- Síndrome do "um servidor por serviço".

Por que virtualizar? Vantagens

- Aumentar em várias vezes a utilização média dos recursos computacionais.
- Reduzir as necessidades de espaço físico e de consumo de energia.
- Ter independência entre a aplicação e o hardware, o que permite a adequação dos recursos às demandas do momento.
- Aumentar a disponibilidade de serviços, com a movimentação das máquinas virtuais entre servidores físicos, sem interrupção, facilitando atualização e manutenção de hardware.
- Flexibilidade no gerenciamento do ambiente, disponibilizando de forma rápida e padronizada novas máquinas virtuais
- Melhor planejando de capacidade futura.

Por que virtualizar? Desvantagens

- Curva de aprendizado alta.
- Custos iniciais de implantação.
- Falta de profissionais capacitados.
- Criação desordenada de máquinas virtuais.
- Concentração de mais de um serviço em um mesmo hardware.

Por que virtualizar? Cuidados

- Definir bem quais servidores podem ser virtualizados para não comprometer o desempenho.
- Administração de ambientes com máquinas virtuais.
- Segurança (política de uso).
- Não misturar com ambientes de teste ou homologação.
- Dependência de algum hardware específico (sensores, leitor de cartão, trava de segurança).

Objetivos de se criar um ambiente virtualizado

- Permitir fácil crescimento.
- Aumentar a disponibilidade.
- Portabilidade de serviços, independência do hardware.
- A complexidade do ambiente físico (energia, ar condicionado e espaço).

Estratégias para implementação de ambiente

virtualizado

- Inicie a virtualização com poucos serviços.
- Começar pequeno tende a ser uma opção muito bem vista.
- Geralmente a implantação se dará em duas fases:
    1. Consolidação de servidores, redução de custos e aumento do uso do hardware disponível.
    2. O foco muda para a entrega de novos serviços e aumento da qualidade e da velocidade dos mesmos.

Aplicações

- Virtualize as aplicações certas.
- Nem toda aplicação é viável para ser virtualizada (banco de dados).
- Aplicações com muita demanda de I/O podem tornam-se ineficientes em máquinas virtuais.

Armazenamento

- Defina uma estratégia de armazenamento.
    - Centralizado, descentralizado (distribuído), replicado.
- Decidir como e onde armazenar as máquinas virtuais.
    - Storage, AoE, iSCSI.

Licenciamento

- Verifique o suporte e licenciamento do software que será hospedado em máquina virtual.
- O mercado de software ainda está reagindo a essa tendência.

Balanceamento

- Combine as máquinas virtuais de forma eficiente.
- É muito mais importante encontrar uma forma de dinamicamente realocar a capacidade dos servidores do que ter um mapa de consolidação perfeito e estático.
- Ser capaz de lidar com o balanceamento de cargas dinamicamente é importante para o sucesso desse tipo de projeto (mas não fundamental, depende do tamanho).

Segurança

- Reforçar isolamento de máquinas virtuais.
    - Na execução no host.
    - No acesso a rede.
- Exploits onde uma máquina virtual *sai* de seu contexto e acessa o host não são impossíveis.
- Garantir fixação de recursos para não permitir que a máquina virtual faça um DoS no host.

Dimensionamento - Vai caber?

- Colete números de utilização de seus servidores físicos.
- Utilize o **dstat** ou alguma ferramenta que meça utilização de discos, rede, cpu e memória.

Dimensionamento - dstat

- **-d**: I/O de disco geral.
- **-n**: tráfego de rede.
- **-c**: consumo de CPU.
- **--output dados.csv**: salva o resultado em um arquivo CSV. Saída do dstat:

```
# dstat -d -n -c
-dsk/total- -net/total- ----total-cpu-usage----
 read  writ| recv  send|usr sys idl wai hiq siq
5232B   30k|   0     0 |  2   0  97   0   0   0
   0    40k| 268B  388B|  1   0  98   1   0   0
   0     0 |   0     0 |  1   0  99   0   0   0
   0     0 |   0     0 |  1   0  99   0   0   0
   0     0 |   0     0 |  6   3  91   0   0   0
   0     0 |   0     0 |  0   0  99   0   0   0
   0    28k|   0     0 |  1   1  97   1   0   0
```

Recuperação de desastres

- Faça backup normalmente, como se a máquina fosse real (Amanda, Bacula, etc).
- Tenha uma máquina física para o backup fora do ambiente virtualizado.
- Torne a instalação de uma máquina nova o mais automatizada possível.

Referências

- Histórico:
  - A história da virtualização: http://www.vmware.com/br/technology/history.html
  - Virtualização - Um pouco de história: http://hbueno.wordpress.com/2009/04/29/virtualizacao-um-pouco-de-historia/
- Conceitos básicos:
  - ABC da virtualização: http://cio.uol.com.br/tecnologia/2007/08/14/idgnoticia.2007-08-14.5515750576/
  - Definições básicas: http://www.sensedia.com/br/anexos/wp_virtualizacao.pdf (Páginas de 1 a 3)
- Virtualização: da teoria a soluções: http://www.gta.ufrj.br/ensino/CPE758/artigos-basicos/cap4-v2.pdf
- Virtualização e Consolidação de Ambientes - 7o. Geinfo:

http://www.ifsc.usp.br/~8geinfo/8geinfo/images/stories/7geinfo/gr-dis/Grupo%20de%20Trabalho%20Virtualizacao.pdf

Referências extras

- Mais questões de virtualização (interessante): 
  http://www.semanainformatica.xl.pt/935/est/100.shtml
- Alta disponibilidade em virtualização de servidores. Estudo de caso: Universidade Estadual de Ciências da Saúde de Alagoas-UNCISAL 
  http://www.aedb.br/seget/artigos10/26_Virtualizacao_SEGET_2010.pdf
- Artigo com dados mais atualizados e orientações gerais: 
  http://www.ibm.com/expressadvantage/br/articles_businessunit_4Q03.phtml

Iniciar apresentação

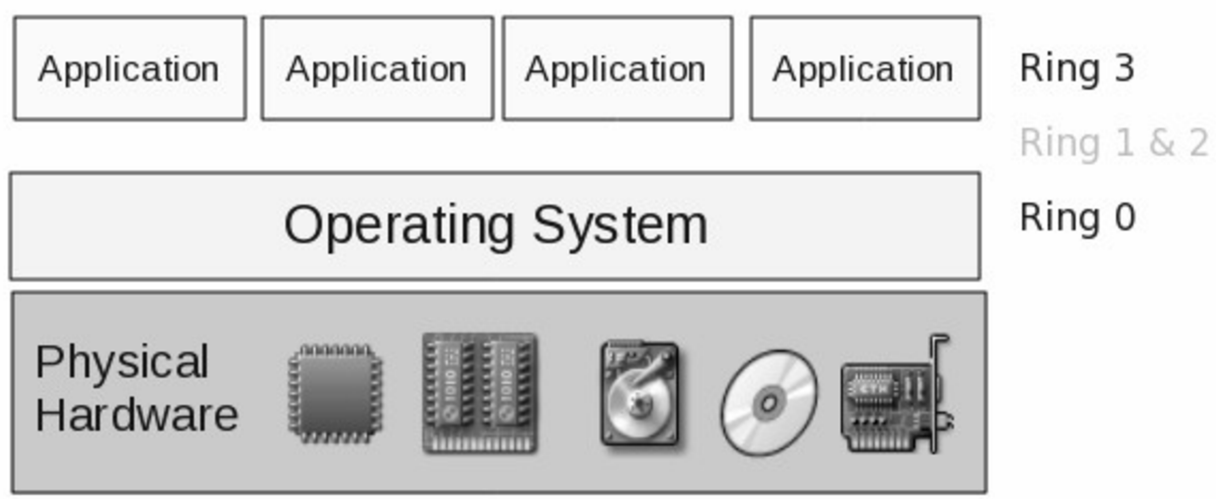
Aula 02 - Conhecendo as soluções livres de virtualização

Sobre esta aula

- Linha do tempo das soluções de virtualização livres.
- Evolução tecnológica das técnicas de virtualização.
- Distribuições, versões, aplicativos.

A arquitetura x86/x86_64

- A fim de proporcionar um ambiente seguro para um sistema operacional, a arquitetura x86 fornece um mecanismo para isolar aplicações de usuários do sistema operacional usando a noção de níveis de privilégio.
- O processador fornece 4 níveis de privilégio, também conhecidos como rings (anéis) que são dispostos em uma forma hierárquica, de 0 a 3. O ring 0 é o mais privilegiado, com acesso total ao hardware e é capaz de executar instruções privilegiadas.

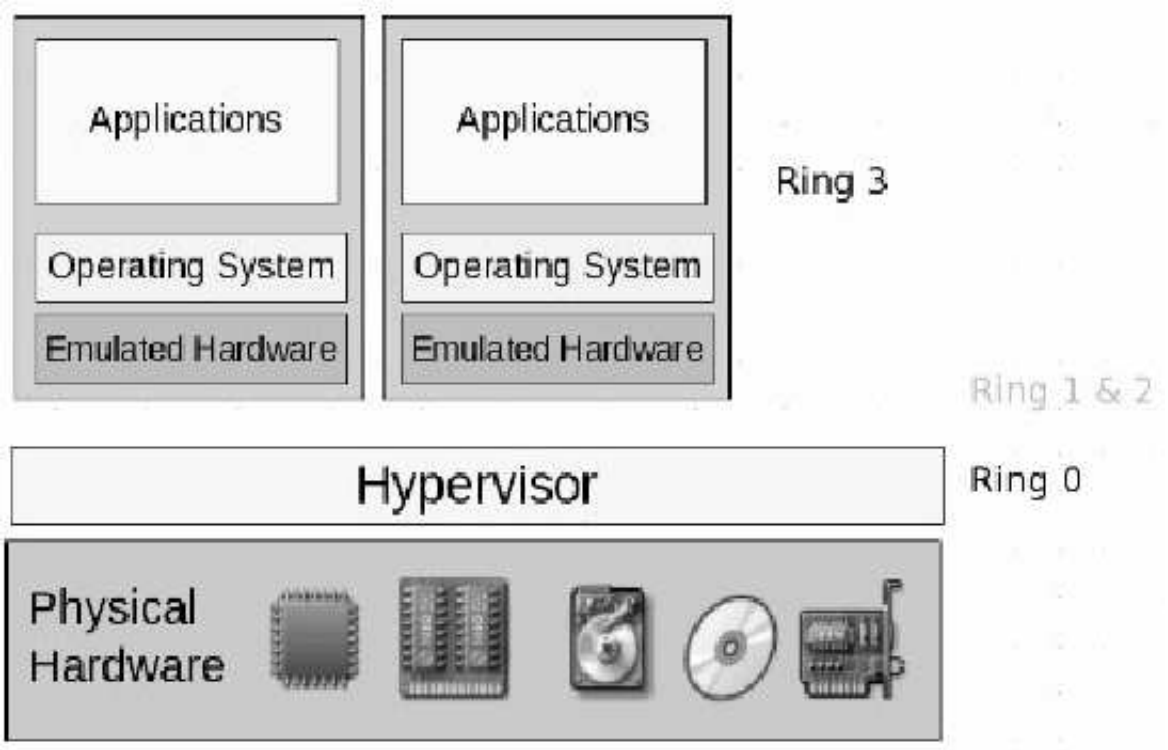


- O sistema operacional roda no ring 0 com o seu kernel controlando o acesso ao hardware.
- Os rings 1, 2 e 3 funcionam em um nível mais baixo de privilégios e são impedidos de executar instruções reservadas para o sistema operacional.
- Sistemas operacionais comumente utilizados, tais como Linux e Windows da Microsoft funcionam em ring 0 e programas de usuário rodam em ring 3. Rings 1 e 2 historicamente não tem sido usados.

A arquitetura x86/x86_64

- Embora este modelo proporcione benefícios para as tradicionais implementações "bare metal" de SOs, ela apresenta desafios em um ambiente virtualizado. Em um ambiente virtualizado o hypervisor deve ser executado no nível mais privilegiado, controlando todo o hardware. Neste modelo, as máquinas virtuais são executadas em um ring inferior desprivilegiado, geralmente no ring 3.
- Dentro do ring 3, podemos ver a máquina virtual em execução com um sistema operacional rodando no hardware virtual (emulado). Como o sistema operacional foi originalmente concebido para ser executado diretamente no hardware e espera ser executado no anel 0, ele vai fazer chamadas que são privilegiados e não autorizadas em ring 3.

A arquitetura x86/x86_64



A arquitetura x86/x86_64

- Quando o sistema operacional faz essas chamadas privilegiadas, o hardware irá interceptar as instruções e emitir uma falha, o que poderá destruir a máquina virtual.
- Grande parte do trabalho realizado antigamente em soluções de virtualização x86 é centrada em torno do manejo de remoção dos privilégios do sistema operacional em execução na máquina virtual, movendo o kernel do sistema operacional do anel de 0 para a 1 (ou superior).

- Os primeiros hypervisors, como o Bochs, criavam um sistema totalmente emulado, com um processador x86 e periféricos em software. Esta técnica resulta em um desempenho muito pobre, então técnicas mais avançadas.

No começo, havia a VMware

- 1998: VMware começa a trabalhar em um Virtual Machine Monitor (VMM) para arquitetura x86
    - Aplica a seguinte patente: Virtualization system including a virtual machine monitor for a computer with a segmented architecture
- 1999: VMware aparece para o mercado e lança o VMware Workstation 1.0 para Windows e Linux.
- 2001: VMware ESX e GSX Server 1.0.
- 2003: VMware ESX 2.0 - Suporte a SMP virtual e VMotion.

Tradução binária (binary translation)

- Neste modelo, criado pela VMware, ao invés de emular o processador, a máquina virtual é executada diretamente na CPU.
- Quando instruções privilegiadas são detectadas, a CPU irá emitir um *trap* que será tratado pelo hypervisor. No entanto, há uma série de instruções x86 que não podem ser tratadas com *traps*, por exemplo: pushf / popf.
- Para lidar com estes casos, uma técnica chamada de tradução binária foi desenvolvida. Neste modelo, o hypervisor analisa a memória da máquina virtual e intercepta essas chamadas antes de serem executadas e dinamicamente reescreve o código na memória.
- O kernel do sistema operacional não tem conhecimento da mudança e funciona normalmente.Esta combinação de *trap-and-execute* e tradução binária permite que qualquer sistema operacional possa executar inalterado em cima do hypervisor.
- Essa abordagem é complexa de implementar, porém rendeu significativos ganhos de desempenho em relação a emular totalmente uma CPU.

O Xen chegou chegando

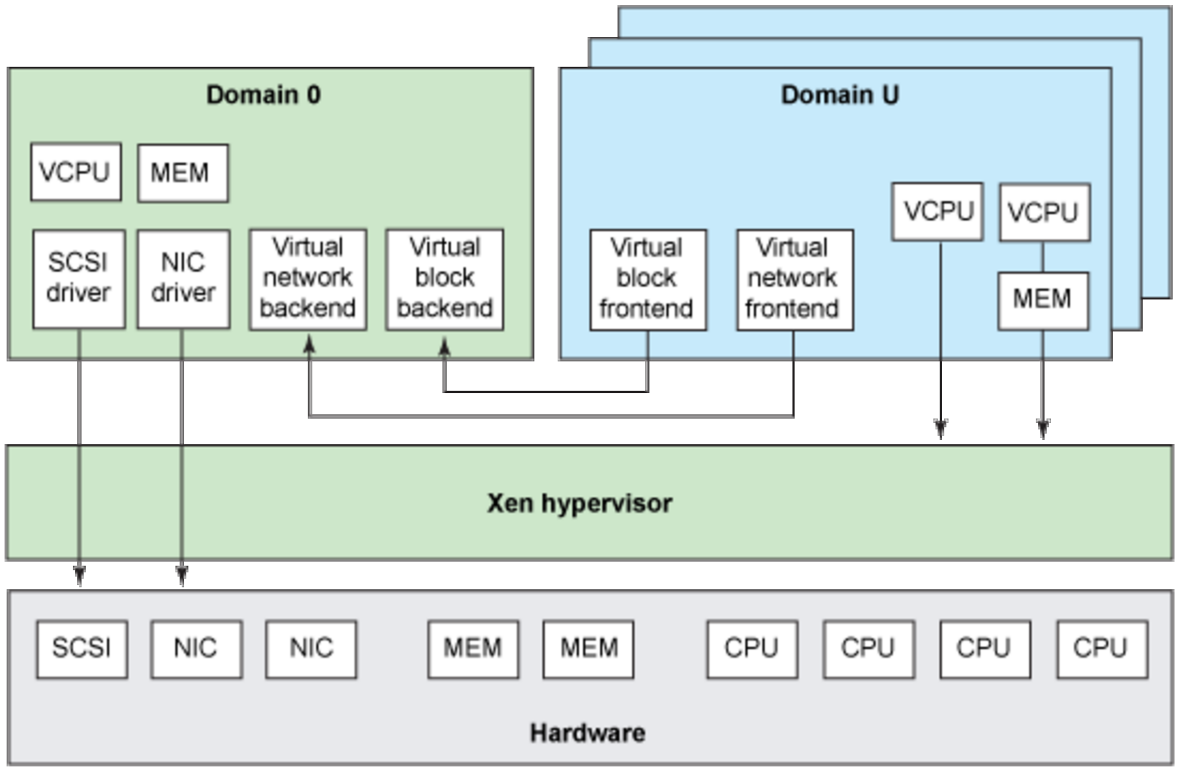
- 2002/2003: Xen 1.0 se torna público com o paper Xen and the Art of Virtualization, cria o termo paravirtualização.
    - Usando o Linux 2.4 para demonstrar a técnica, bateu e muito no VMware em performance.
- 2004:
    - Debian inclui Xen 1.2 na unstable
    - Pesquisadores de Cambridge fundam a XenSource, para que tornar o Xen um

produto corporativo e não apenas o resultado de trabalho acadêmico. Projeto continua como GPL e relativamente aberto à comunidade.
    - Xen 2.0: Linux 2.4.27/2.6.9 and NetBSD: http://lwn.net/Articles/109789/
- 2005
    - Junho: Fedora Core 4 inclui suporte a um snaphost do Xen 3.0
    - Novembro: Sun começa a portar Solaris para o Xen: http://virtualization.info/en/news/2005/11/xen-to-be-integrated-in-solaris-10.html
    - Dezembro:
        - Xen 3.0 é lançado: http://lwn.net/Articles/162841/
        - **Intel lança CPUs Xeon com VT-x**

Paravirtualização

- Em vez de lidar com uma instruções privilegiadas, a abordagem da paravirtualização é de modificar o sistema operacional em execução na máquina virtual e substituir todas as instruções privilegiadas com chamadas diretas ao hypervisor.
- Neste modelo, o sistema operacional modificado está ciente de que ele está sendo executado em um hypervisor e pode cooperar com o hypervisor para agendamento de I/O, eliminando a necessidade de emular dispositivos de hardware como placas de rede e controladores de disco.

Xen



Xen

- A plataforma Xen é composta por dois componentes: o hypervisor Xen, que é responsável pelo núcleo de atividades como CPU, virtualização de memória, gerenciamento de energia e escalonamento de máquinas virtuais.
- O hypervisor Xen carrega uma máquina virtual privilegiada especial chamada Domínio0 ou dom0. Esta máquina virtual tem acesso direto ao hardware e fornece drivers de dispositivo e gestão de I/O para as máquinas virtuais.
- Cada máquina virtual, conhecida como um domínio sem privilégios ou domU, contém uma versão modificada do kernel do Linux que ao invés de comunicar-se diretamente

com interfaces de hardware, fala com o hypervisor Xen.

- CPU e acesso à memória são tratados diretamente pelo hypervisor Xen, mas I/O é direcionado para o domínio 0. O kernel do Linux inclui o "front end" dispositivos de rede e bloco I/O. Os pedidos de I/O são passados para o "back-end" no domínio do processo 0, que gerencia o I/O.
- Neste modelo, o kernel hóspede, em domU é executado em ring 1, enquanto o espaço de usuário executado no anel 3.
- Enquanto o Xen é geralmente categorizada como sendo uma solução *Type 1* de hypervisor, toda a plataforma exige um sistema operacional Dom0 para acessar o hardware.

Surge o KVM

- 2006
  - Março: Fedora Core 5 com Xen 3.0. SUporte HVM precisa de CPU com virtualização.
  - Maio: AMD lança CPUs com suporte a virtualização.
  - Julho
    - VMware vai suportar Xen: http://virtualization.info/en/news/2006/07/vmware-could-decide-to-support.html
    - Novell lança SuSE Enterprise 10 com Xen, primeira empresa fora XenSource a dar suporte corporativo ao Xen.
  - Outubro: KVM é submetido para o Linux pela Quamranet: https://lkml.org/lkml/2006/10/19/146

Descrição do KVM

```
The following patchset adds a driver for Intel's hardware
virtualization
extensions to the x86 architecture.  The driver adds a character device
(/dev/kvm) that exposes the virtualization capabilities to userspace.
Using
this driver, a process can run a virtual machine (a "guest") in a fully
virtualized PC containing its own virtual hard disks, network adapters,
and
display.

Using this driver, one can start multiple virtual machines on a host.
Each
virtual machine is a process on the host; a virtual cpu is a thread in
that
process.  kill(1), nice(1), top(1) work as expected.
```

KVM em funcionamento

```
#include <linux/kvm.h>

open("/dev/kvm")
ioctl(KVM_CREATE_VM)
ioctl(KVM_CREATE_VCPU)
for (;;) {
     ioctl(KVM_RUN)
     switch (exit_reason) {
     case KVM_EXIT_IO:  /* ... */
```

```
    case KVM_EXIT_HLT: /* ... */
    }
}
```

- Mais detalhes em: http://blog.vmsplice.net/2011/03/qemu-internals-big-picture-overview.html

KVM

```
modprobe kvm
modprobe kvm_intel ou kvm_amd
qemu-img create -f qcow vm-disk.img 4G
kvm -m 384 -cdrom guestos.iso -hda vm-disk.img -boot d
```

KVM

- KVM é implementado como um módulo de kernel carregável que converte o Linux em um hypervisor *bare-metal*.
- Há dois princípios fundamentais de projeto que ajudaram o KVM a amadurecer rapidamente em um hypervisor estável e de alto desempenho.

Use o hardware, Luke

- O KVM foi projetado após o advento da virtualização assistida por hardware.
- O hypervisor KVM requer processadores com Intel VT-x ou AMD-V habilitados e utiliza esses recursos para virtualizar o CPU. Ao exigir o suporte de hardware ao invés de apenas otimizar com o mesmo se disponível, o KVM foi capaz de projetar uma solução otimizada sem a necessidade de modificações no sistema operacional virtualizado.

Não reinvente a roda

- Há muitos componentes que um hypervisor exige, além da capacidade de virtualizar a CPU e memória, por exemplo
  - Gerenciador de memória
  - Escalonador de processos
  - Pilha de I/O e rede
  - **Drivers de dispositivo**
  - Gerenciamento de segurança.

Simplificando

- Na verdade um hypervisor é realmente um sistema operacional especializado, diferindo apenas dos de uso geral pois que se executam máquinas virtuais ao invés de aplicações.

Por que o Linux?

- Uma vez que o kernel do Linux já inclui os principais recursos exigidos por um hypervisor e foi amadurecido em uma plataforma da madura e estável por mais de 15 anos de apoio e desenvolvimento, é mais eficiente de construir sobre essa base ao invés de escrever todos os componentes necessários, tais como um gerenciador de memória,

escalonador, etc a partir do zero.

- Neste contexto, o KVM se beneficiou da experiência do Xen. Um dos principais desafios do arquitetura do Xen é a arquitetura da divisão de domínio0 e o hypervisor Xen. Desde o hypervisor Xen fornece os recursos da plataforma central dentro da pilha, ele tem a necessidade de implementar esses recursos, como escalonador e gerenciador de memória a partir do zero.

Por que o Linux?

- Por exemplo, enquanto o kernel do Linux possui um gerenciador de memória madura e comprovada, incluindo suporte para NUMA e sistemas de grande escala, o hypervisor Xen necessita construir este apoio a partir do zero. Da mesma forma recursos como gerenciamento de energia que já estão maduros e comprovado em campo no Linux tinha que ser

re-implementado no hypervisor Xen.

- Outra decisão importante tomada pela equipe KVM era incorporar o KVM no kernel do Linux o quanto antes. O código KVM foi apresentado à comunidade do kernel Linux em dezembro de 2006 e foi aceito em o kernel 2.6.20, em janeiro de 2007.
- Neste ponto KVM tornou-se parte do núcleo do Linux e é capaz de herdar recursos chave do kernel do Linux.

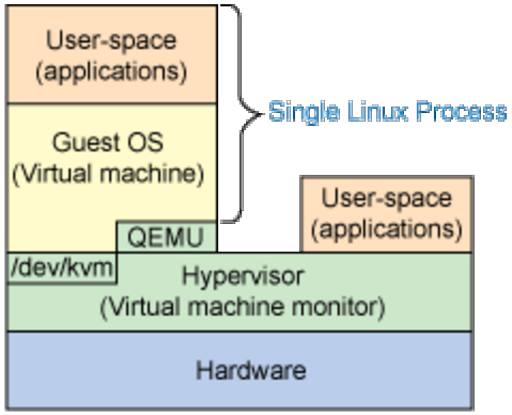
Por que o Linux?

- Em contrapartida, as correções necessárias para construir o Domínio0 Linux para o Xen ainda não faz parte do kernel Linux e obriga os fornecedores a criar e manter um

fork do kernel Linux.

- Isto tem levar a um aumento dos encargos com os distribuidores de Xen, que não podem alavancar facilmente os recursos do montante kernel. Qualquer novo recurso de correção de bug ou um patch para o kernel acrescentado a montante deve ser back-portados para trabalhar com o patch define Xen.

Arquitetura KVM



Arquitetura KVM

- I/O e dipositivos são virtualizados através de um processo QEMU levemente modificado (cara processo executa uma máquina virtual).
- Execução de I/O de um sistema operacional convidado é fornecido com o QEMU.
- QEMU é uma solução de virtualização de que permite a virtualização de um ambiente de PC inteiro (inclusive discos, placas gráficas, dispositivos de rede). Quaisquer solicitações de I / O de um sistema operacional convidado faz são interceptadas e direcionadas para o modo de usuário para ser emulado pelo processo QEMU.

Arquitetura KVM

- KVM permite a virtualização de memória através do dispositivo **/dev/kvm**.
- Cada sistema operacional convidado tem seu próprio espaço de endereço que é mapeado quando o convidado é instanciado.
- A memória física que é mapeada para o sistema operacional guest é a memória virtual mapeada para o processo.

KVM e Xen iniciam competição

- 2007
  - Fevereiro:

        - Linux 2.6.20 é lançado com KVM incluído (Xen não existia upstream).
        - KVM começa a fazer barulho: KVM steals virtualization spotlight
    - Março: Red Hat Enterprise 5 lançado com suporte a Xen 3.0.
    - Abril: Entrevista com Avi Kivity: http://kerneltrap.org/node/8088
    - Julho: Microsoft anuncia parceria com a XenSource na especificação de um para Hypervisor (o que viria a ser o Hyper-V) e HP apoia: http://virtualization.info/en/news/2006/07/microsoft-to-support-xen-virtual.html
    - **Agosto: Citrix compra XenSource por US$ 500 milhões (produtos como XenServer entre outros).**
    - Novembro: Oracle VM, baseado no Xen: http://virtualization.info/en/news/2007/11/oracle-announces-its-own-xen-based.html

KVM e Xen iniciam competição

- 2008
    - Abril: Canonical adota KVM como sua solução de virtualização: http://www.theregister.co.uk/2008/02/11/ubuntu_hardy_heron_kvm/
    - Junho: Microsoft lança Hyper-V, "compatível" com Xen: http://virtualization.info/en/news/2008/06/microsoft-hyper-v-day-after.html
    - Agosto: Lançado Xen 3.3.
    - Setembro: Red Hat compra Quamranet: http://www.redhat.com/about/news/prarchive/2008/qumranet.html
- 2009
    - Setembro: Red Hat incluí KVM no RHEL 5.4 junto com Xen: http://arstechnica.com/open-source/news/2009/09/red-hat-moves-forward-with-kvm-virtualization-in-rhel-54.ars
- 2010
    - Março: Novell vai suportar KVM no SLES 11 e diz que KVM é o futuro: http://www.h-online.com/open/news/item/Novell-supports-KVM-963031.html
    - Abril: RHEL 6 não terá Xen como hypervisor: http://www.computerworld.com/s/article/9175883/Red_Hat_drops_Xen_from_RHEL

Porque KVM

- Linux é o Host/Hypervisor/VMM
- Máquinas virtuais são simples processos.
- kill, top e outros comandos podem ser utilizados normalmente.
- Utilização dos escalonadores de I/O, memória e CPU do Linux.

- Usa modelo de segurança do Linux
  - As máquinas virtuais rodam com privilégios de usuário.
- Drivers paravirtualizados para melhor performance onde necessário.
- Possibilidade de fazer swap com a memória da máquina virtual.
- NUMA
- Se beneficia diretamente de todas as melhorias que o Linux recebe.

Paravirtualização não faz mais sentido?

```
From: Alok Kataria <akataria-AT-vmware.com>
To: Ingo Molnar <mingo-AT-elte.hu>, Thomas Gleixner <tglx-AT-
linutronix.de>, "H. Peter Anvin" <hpa-AT-zytor.com>

We ran a few experiments to compare performance of VMware's
paravirtualization technique (VMI) and hardware MMU technologies
(HWMMU)
on VMware's hypervisor.
...
In most of the benchmarks, EPT/NPT (hwmmu) technologies are at par or
provide better performance compared to VMI.
The experiments included comparing performance across various micro and
real world like benchmarks.
...
In light of these results and availability of such hardware, we have
decided to stop supporting VMI in our future products.
```

- http://lwn.net/Articles/353853/

paravirt_ops

- Mais coisas sobre isso nos comentários: http://lwn.net/Articles/353852/
- Comentários muito bons sobre Xen VS KVM: http://lwn.net/Articles/374191/

Resumindo

- O que é KVM: Linux + drivers para CPUs Intel e AMD + QEMU
  - Simples de usar, rápido e robusto.

Ferramentas de gerenciamento

- libvirt (Virt-Manager, virsh)
- Ganeti

- OpenNebula
- Eucalyptus
- OpenStack
- OpenQRM
- ConVirt
- Red Hat Enterprise Virtualization
- Proxmox
- Enomaly
- OpenNode
- oVirt
- Karesansui
- Univention Virtual Machine Manager

Referências

- Paper sobre VT-x: http://download.intel.com/technology/itj/2006/v10i3/v10-i3-art01.pdf
- Discover the Linux Kernel Virtual Machine: http://www.ibm.com/developerworks/linux/library/l-linux-kvm/
- Documento interessante sobre KVM, acho que dá um bom overview: http://www.redhat.com/f/pdf/rhev/DOC-KVM.pdf
- http://blog.vmsplice.net/2011/03/qemu-internals-overall-architecture-and.html
- Slides sobre KVM: http://www.geeksofpune.in/drupal/files/8025301-kvmway.pdf
- http://www.linux-kvm.org/wiki/images/4/42/Kvm-device-assignment.pdf
- kvm: the Linux Virtual Machine Monitor: http://www.kernel.org/doc/ols/2007/ols2007v1-pages-225-230.pdf

Aula 03 - Primeiros passos com KVM

Sobre

- Objetivos:
  - Conhecer o hardware do host.
  - Instalação de pacotes e módulos necessários.
  - Dar uma visão geral da mecânica do KVM, usando os principais parâmetros de linha comando e o monitor.
  - Compreender o que é backend e frontend.
  - Criar uma máquinas virtual simples.

Versões do KVM nas distribuições

Distribuição	Kernel	qemu-kvm/kvm
Debian Squeeze 6.0	2.6.32	0.12.5
RHEL 5.6	2.6.18 + muitos backports	0.12.5 + backports
RHEL 6.0	2.6.32 + backports	0.12.1 + patches e backports
Ubuntu 10.04 LTS	2.6.32	0.12.3 + patches

Carregando módulo

```
modprobe kvm kvm_intel
```

Verificando /dev/kvm:

```
ls -l /dev/kvm
crw-rw---- 1 root kvm 10, 232 Dec 14 17:20 /dev/kvm
```

Boot de um CD/DVD

Iniciando uma VM dando boot em uma imagem de LiveCD:

```
kvm -drive file=/caminho/imagem.iso,media=cdrom -m 512
```

Com esse comando uma janela será aberta e a máquina virtual estará em funcionamento com 512 Mb de RAM. O boot será dado na imagem ISO. Apesar do parâmetro **media** se chamar

cdrom, essa imagem pode ser de DVD também.

Existe também essa sintaxe:

```
kvm -cdrom /caminho/imagem.iso -m 512
```

O parâmetro **-cdrom** é mantido apenas para garantir compatibilidade e seu uso não é mais recomendado.

É possível também utilizar uma mídia real, apontando o dispositivo onde ela se encontra:

```
kvm -drive file=/dev/sr0,media=cdrom -m 512
```

- Dentro da máquina virtual, dar um lspci para listar os dispositivos padrões (vídeo, rede, etc).
- Dica para teclado: adicionar **-k** pt-br caso tenha problemas com a tecla / **?**.

Usando e conhecendo o monitor

Clique na janela da máquina virtual para que ela receba o foco do mouse. Teclas de atalho importantes:

- [Ctrl] + [Alt]: Tira o mouse do foco da janela
- [Ctrl] + [Alt] + [1]: Mostra o saída de vídeo da máquina virtual.
- [Ctrl] + [Alt] + [2]: Muda para o console de comando do qemu-kvm, o monitor.
- [Ctrl] + [Alt] + [3]: Mostra a saída da porta serial

O monitor do qemu-kvm tem várias funções, dentre elas:

- Changing or eject removable media (CD / DVD-ROMs, floppy disks).
- The freezing and further run a virtual machine.
- Backing up and restoring various states of the virtual machine.
- Inspecting the state of a virtual machine.
- The migration of a virtual machine to another host.
- Changing the hardware (USB, PCI, ...).
- The injection of emulated hardware failures.

Com a combinação [Ctrl] + [Alt] + [2] é aberto o monitor do QEMU.

Para conhecer mais comandos do monitor:

```
(Qemu) help
```

Comandos básicos do monitor

- The info command gives status information on the current instance. If you give info only, parameter is a list of possible output. The QEMU version is version info to determine.

```
(Qemu) info version
0.12.5
```

- The kvm command info shows if the KVM hardware virtualization is enabled or not.

```
(Qemu) info kvm
kvm support: enabled
```

- The quit command terminates the instance. This corresponds to switching off at a real machine and can cause data loss.

```
(Qemu) quit
```

- The reset button corresponds to a real machine, the command system_reset.

```
(Qemu) system_reset
```

```
O daemon acpid tem quem que estar instalado no guest.
```

- Some keyboard shortcuts may not be forwarded to the host system. For instance, several keyboard layouts cause problems. There are also catch some keyboard shortcuts from the host system and not forwarded to the host system. This behavior can happen with the Windows NT login screen annoying. Here one needs the [Ctrl] + [Alt] + [Del]. The QEMU monitor offer the possibility of such key combinations to the host system to send. The following example sends the key combination [Ctrl] + [Alt] + [Del].

```
(Qemu) sendkey ctrl-alt-delete
```

- The command sets Screendump the host system to display photos. This is useful in host systems that can not take screenshots. The screenshots are saved as PPM file (Portable Pixmap).

```
(Qemu) screendump imagem.ppm
```

Manipulando dispositivos - monitor

```
kvm -drive file=/caminho/imagem.iso,media=cdrom -m 512 -monitor stdio
```

Ou:

```
kvm -drive file=/caminho/imagem.iso,media=cdrom -m 512 \
        -chardev stdio,id=mon0 -mon chardev=mon0
```

Ou múltiplos monitores, em diferentes protocolos:

```
kvm -chardev stdio,id=mon0 -mon chardev=mon0 \
-chardev socket,id=tcpmon0,port=5000,host=localhost,server,nowait -mon
chardev=tcpmon0
-drive file=/caminho/imagem.iso,media=cdrom -m 512
```

O monitor em Ctrl+Alt+2 não existe mais. Pode dar **nc localhost 5000** que o monitor estará acessível. Consulte a manpage do kvm para conhecer todos os dispositivos de backend para chardevs.

Fazendo a instalação de uma VM

- . Escolha a distribuição de sua preferência e baixe a iso:
  - CentOS 5.5
  - Debian Squeeze 6.0
  - Ubuntu LTS 10.04
- Crie um arquivo para armazenar a VM:

```
qemu-img create -f raw nome-do-arquivo.img 5G
```

- Bootar a VM:

```
kvm -m 1024 -drive file=/caminho/imagem.iso,media=cdrom,index=1,boot=on
\
 -drive file=/caminho/nome-do-arquivo.img,media=disk,index=0
```

- Instale sua distro
- Mate a VM e boote ela sem CDROM:

```
kvm -m 1024 -drive file=a.img,media=disk,index=0
```

Saindo do ambiente gráfico - VNC

```
kvm -m 1024 -drive file=a.img,media=disk,index=0 -vnc :0
```

O kvm estará ouvindo na porta 5900+d a porta que foi passada na linha de comando. Pode-se usar qualquer cliente VNC para ver o display VGA.

```
$ netstat -nltp|grep kvm
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:5900            0.0.0.0:*
LISTEN      3589/kvm
```

- Colocando senha:

```
kvm -m 1024 -drive file=imagem.img,media=disk,index=0 -vnc :0,password
-monitor stdio
QEMU 0.14.0 monitor - type 'help' for more information
(qemu) change vnc password
Password: ******
(qemu)
```

Saindo do ambiente gráfico - porta serial

- Edite os arquivos de configuração de sua distro para que exista um console disponível em uma porta serial.
- Dica (CentOS e Debian):
  - /etc/inittab: **s1:2345:respawn:/sbin/getty 38400 ttyS0**
  - No grub **(/etc/grub.conf)**, coloque essa linha como parâmetro de boot: **console=ttyS0**
- Dica (Ubuntu)
  - **cp /etc/init/tty1.conf /etc/init/ttyS0.conf**, alterando o conteudo de **tty1** para **ttyS0**
  - Editar **/etc/default/grub**
  - Na linha de **GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"** mudar para **GRUB_CMDLINE_LINUX_DEFAULT="console=ttyS0"**
  - Não esquecer de atualizar o grub: **update-grub**

```
kvm -m 1024 -drive file=imagem.img,media=disk,index=0 -serial stdio
```

```
kvm -m 1024 -drive file=imagem.img,media=disk,index=0 -serial
unix:/tmp/portaserial,server,nowait

socat UNIX:/tmp/portaserial STDIO,raw,echo=0,escape=0x1d
```

Sintaxe moderna:

```
kvm -m 1024 -drive file=imagem.img,media=disk,index=0 \
```

```
-chardev socket,path=/tmp/portaserial,server,nowait,id=serial0 \
-device isa-serial,chardev=serial0
```

Indo para background

- Juntando tudo:

```
kvm -m 1024 -drive file=imagem.img,media=disk,index=0 \
-chardev socket,path=/tmp/portaserial,server,nowait,id=serial0 \
-device isa-serial,chardev=serial0 \
-vnc :0 \
-chardev socket,id=tcpmon0,port=5000,host=localhost,server,nowait \
-mon chardev=tcpmon0 \
-daemonize
```

- Acessando a porta serial:

```
socat UNIX:/tmp/portaserial STDIO,raw,echo=0,escape=0x1d
```

- Acessando o console:

```
vncviewer :0
```

- Acessando o monitor:

```
nc localhost 5000
```

Referências

- http://www.gdhpress.com.br/blog/kvm-facil/
- http://en.wikibooks.org/wiki/QEMU/Monitor

Aula 04 - Criando máquinas virtuais

Sobre

- Objetivos:
  - Criar máquinas virtuais seguindo uma especificação pré-determinada.
  - Descobrir mais recursos do KVM.
  - Criar ambiente com diversas máquinas virtuais para os futuros experimentos do curso.

Atividade de laboratório

Criar pelo menos 3 máquinas virtuais, seguindo a seguinte configuração:

Nome	Armazenamento	Quantidade de discos	Tamanho de cada disco	Memória
didi	imagem raw	2	4GB / 1GB	1GB
dede	imagem raw	2	4GB / 1GB	1GB
mussum	LVM	1	5G	1GB
zacarias	LVM	1	5G	1GB

Essas máquinas devem possuir as seguintes características:

- Qualquer uma das distribuições: Debian Squeeze 6.0, CentOS 5.5 ou Ubuntu 10.04 LTS.
- As máquinas didi e dede devem possuir o ponto de montagem /srv apontado para os discos de 1GB.
- Crie um volume lógico usando *lvcreate* para as máquinas mussum e zacarias.
- Um console serial acessível com prompt de login via socket unix.
- Monitor acessível via localhost tcp ou socket unix.
- Funcionar em background.
- Não depender de janela gráfica.
- Todas as máquinas devem estar em execução ao mesmo tempo.
- Salve em um script os comandos para criação de cada máquina virtual.

Dicas:

- Faça uma instalação o mais enxuta possível para não perder tempo e não ocupar muito espaço.
- **Use a criatividade!** Comandos como cp, rsync, mount -o loop ou dd para acessar e copiar o file system de uma máquina virtual podem te ajudar a ganhar **muito** tempo 😬
- Precisamos dessas máquinas virtuais prontas para as configurações que faremos com rede.

Resultado

Ambiente:

Nome	Armazenamento	Quantidade de discos	Tamanho de cada disco	Memória	VNC Port	Monitor Port	Console Serial
didi	imagem raw	2	4GB /	1GB	5902	5002	/var/lib/images/serial/didi

			1GB				
dede	imagem raw	2	4GB / 1GB	1GB	5903	5003	/var/lib/images/serial/dede
mussum	LVM	1	5G	1GB	5901	5001	/var/lib/images/serial/mussum
zacarias	LVM	1	5G	1GB	5900	5000	/var/lib/images/serial/zacarias

- Script de Boot final das maquinas

```
kvm -m 1024 \
        -name zacarias \
        -drive file=/dev/volumes/zacarias,media=disk,index=0,boot=on \
        -vnc :0 \
        -chardev socket,path=/var/lib/images/serial/zacarias,server,nowait,id=serial0
\
        -device isa-serial,chardev=serial0 \
        -chardev socket,id=tcpmon0,port=5000,host=localhost,server,nowait \
        -mon chardev=tcpmon0 \
        -daemonize

sleep 1

kvm -m 1024 \
        -name mussum \
        -drive file=/dev/volumes/mussum,media=disk,index=0,boot=on \
        -vnc :1 \
        -chardev socket,path=/var/lib/images/serial/mussum,server,nowait,id=serial0 \
        -device isa-serial,chardev=serial0 \
        -chardev socket,id=tcpmon0,port=5001,host=localhost,server,nowait \
        -mon chardev=tcpmon0 \
        -daemonize

sleep 1

kvm -m 1024 \
        -name didi \
        -drive file=/var/lib/images/didi-disco1.img,media=disk,index=0,boot=on \
        -drive file=/var/lib/images/didi-disco2.img,media=disk,index=1 \
        -vnc :2 \
        -chardev socket,path=/var/lib/images/serial/didi,server,nowait,id=serial0 \
        -device isa-serial,chardev=serial0 \
        -chardev socket,id=tcpmon0,port=5002,host=localhost,server,nowait \
        -mon chardev=tcpmon0 \
        -daemonize

sleep 1

kvm -m 1024 \
        -name dede \
        -drive file=/var/lib/images/dede-disco1.img,media=disk,index=0,boot=on \
        -drive file=/var/lib/images/dede-disco2.img,media=disk,index=1 \
        -vnc :3 \
        -chardev socket,path=/var/lib/images/serial/dede,server,nowait,id=serial0 \
        -device isa-serial,chardev=serial0 \
        -chardev socket,id=tcpmon0,port=5003,host=localhost,server,nowait \
        -mon chardev=tcpmon0 \
        -daemonize
```

Aula 05 - Configuração de rede

Sobre

- Objetivos:
  - Entendimento do funcionamento de bridges e interfaces tap.
  - Conhecimento das opções de dispositivos para frontend.
  - utilização de scripts para configuração de interfaces.

Configuração padrão

- Por padrão o KVM cria uma pilha de rede em espaço de usuário dentro do processo que contém a máquina virtual.
  - Nesse ambiente (dentro do processo) existe uma rede virtual que possui:
    - Servidor DHCP
    - Servidor DNS
    - Gateway
- A interface eth0 dentro da máquina virtual, a grosso modo, está atrás de um NAT.

Checando a configuração padrão

- Dê boot em uma VM, vamos verificar como está a configuração de rede obtida via DHCP:

```
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:34:56
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:3456/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1986 (1.9 KB)  TX bytes:1818 (1.8 KB)
          Interrupt:10 Base address:0xe000

# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
Iface
10.0.2.0        0.0.0.0         255.255.255.0   U     0      0        0
eth0
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0        0
eth0
```

```
# dmesg |grep 8139
[    1.097676] 8139cp: 10/100 PCI Ethernet driver v1.3 (Mar 22, 2004)
[    1.100073] 8139cp 0000:00:03.0: PCI INT A -> Link[LNKC] -> GSI 10
(level, high) -> IRQ 10
[    1.105789] eth0: RTL-8139C+ at 0xffffc900004fe000,
52:54:00:12:34:56, IRQ 10
[    1.107021] 8139cp 0000:00:03.0: setting latency timer to 64
[    1.230095] 8139too Fast Ethernet driver 0.9.28
```

Configuração de rede padrão

- Por padrão, o kvm usa os seguintes parâmetros para configurar um dispositivo de rede:

```
# kvm -netdev type=user,id=net0 -device rtl8139,netdev=net0
```

Apenas como referência, essa é a sintaxe antiga para se configurar o mesmo dispositivo de rede:

```
# kvm -net user,vlan=0 -net nic,model=rtl8139,vlan=0
```

- Em ambos os casos o endereço MAC apresentado para o guest é sempre o mesmo.
- O processo do KVM está tunelando tudo todo o tráfego em espaço de usuário.
- Prático para configurações simples.
- A performance é limitada e baixa.
- Difícil e inconveniente configuração para acesso externo.
- Não existe uma "interface de verdade" no host utilizada pelo guest.

Mudando o MAC

```
# kvm -netdev type=user,id=net0 -device
rtl8139,netdev=net0,mac=00:00:00:00:AB:10
```

Opções de frontend e backend

- Principais dispositivos suportados (frontend):
  - rtl8139 (Realtek 8139)
  - e1000 (Intel E1000)
  - virtio-net-pci (Interface paravirtualizada VirtIO)
- Principais backends suportados:
  - tap
  - slirp (Pilha de rede em modo-usuário)

- VDE (Virtual Distributed Ethernet)

Conhecendo interfaces tap

- Vamos criar uma interface do tipo tap:

```
# tunctl -t tap0
Set 'tap0' persistent and owned by uid 0

# ifconfig tap0
tap0      Link encap:Ethernet  HWaddr 9a:a8:03:d2:d9:b8
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- Interfaces tap fazem encaminhamento de quadros em nível 2 (Ethernet).
- Residem dentro do kernel.
- Podem ser manipuladas normalmente como se fossem uma interface de rede real.
- O endereço MAC é aleatório.
- Levantando a interface tap0 criada:

```
# ifconfig tap0 up

ou

# ip link set tap0 up
```

Usando a interface tap com o kvm

- Abra um shell como root, e inicie um tcpdump na interface tap0

```
# tcpdump -i tap0
```

- **IMPORTANTE**: Edite o script /etc/kvm/kvm-ifup no host e comente **TODAS** as linhas antes de iniciar a máquina virtual.
- Agora vamos iniciar uma VM qualquer e associar a interface tap0 do host com o dispositivo do guest:

```
# kvm -netdev type=tap,ifname=tap0,id=net0 -device rtl8139,netdev=net0
```

- Configurando o MAC (opcionalmente)

```
# kvm -netdev type=tap,ifname=tap0,id=net0 -device
rtl8139,netdev=net0,mac=00:00:00:00:AB:10
```

- Como a máquina virtual estava configurada para DHCP, você verá no tcpdump os broadcasts (que não estão indo para lugar nenhum).
- Saída parecida com essa:

```
# tcpdump -i tap0
tcpdump: WARNING: tap0: no IPv4 address assigned
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on tap0, link-type EN10MB (Ethernet), capture size 65535
bytes
00:08:40.586856 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP,
Request from 00:00:00:00:00:10 (oui Ethernet), length 300
00:08:43.952360 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP,
Request from 00:00:00:00:00:10 (oui Ethernet), length 300
00:08:49.952271 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP,
Request from 00:00:00:00:00:10 (oui Ethernet), length 300
```

Configurando interfaces tap

- Agora nós temos a interface virtual eth0 do guest "plugada" na interface tap0 do host. Todos os quadros Ethernet são copiados um para o outro. Vamos avançar na configuração.
- Coloque um IP para na interface tap0:

```
# ifconfig tap0 192.168.0.1 netmask 255.255.255.0 up
```

- Coloque um IP para na interface eth0 da máquina virtual:

```
# ifconfig eth0 192.168.0.2 netmask 255.255.255.0 up
```

- Agora vamos pingar! Do host:

```
# ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_req=1 ttl=64 time=0.279 ms
64 bytes from 192.168.0.2: icmp_req=2 ttl=64 time=0.313 ms
64 bytes from 192.168.0.2: icmp_req=3 ttl=64 time=0.478 ms
```

- De dentro da máquina virtual:

```
# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
```

```
64 bytes from 192.168.0.2: icmp_req=1 ttl=64 time=0.679 ms
64 bytes from 192.168.0.2: icmp_req=2 ttl=64 time=0.513 ms
64 bytes from 192.168.0.2: icmp_req=3 ttl=64 time=0.488 ms
```

- Podemos ter várias interfaces tap, cada uma plugada em um guest. Não é possível usar o mesmo dispositivo tap para mais de um guest.
- Atividade rápida:
  - Configure a rede em outra VM, da mesma maneira que a anterior porém com IPs 10.0.2.0/24, repetindo o procedimento acima com suas devidas altereações.

Configurando acesso ao mundo externo

- Para dar acesso externo para aos guests, devemos habilitar o encaminhamento de pacotes no host e fazer NAT.

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

- Características gerais dessa configuração:
  - Guests não tem acesso ao segmento de rede físico.
  - O acesso ao meio externo precisa ser explicitamente configurado no host.
  - Não é necessário utilizar somente NAT, pode-se utilizar roteamento também.
  - Comunicação entre os guests precisa incluir rotas no host.

XXX Estabelecemos um link direto entre as VMs, fazer diagrama que mostra onde estão as filas de transmissão. Fila do guest, fila do qemu, fila do kernel.

Utilizando bridges

- Como facilitar a comunicação entre os guests? Usando bridges!
  - A bridge é uma maneira de conectar dois segmentos Ethernet, independentemente de protocolo.
  - Os quadros são encaminhados com base no endereço Ethernet, ao invés de endereços IP (como um roteador).
  - Uma vez que o encaminhamento é feito na camada 2, todos os protocolos podem circular de forma transparente através de uma bridge.
  - O código de ponte Linux implementa um subconjunto do padrão ANSI / IEEE 802.1d.

Criando uma bridge

- Para criar uma bridge:

```
# brctl addbr br0
```

- Cria uma brigde lógica chamada br0.
- Sempre é necessário pelo menos uma instância lógica a fazer qualquer ponte em tudo.
- Você pode imaginar a bridge como um agregador de interfaces.
- Cada bridge é representada por uma nova interface de rede.

```
# ifconfig br0
br0       Link encap:Ethernet  HWaddr a2:87:29:42:07:e1
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

- Para remover uma bridge:

```
# brctl delbr br0
```

> Encare a bridge como um simples switch de camada 2 virtual, que conhece os endereços MAC de cada porta e faz o encaminhamento de quadros porta-a-porta.

Adicionando interfaces a uma bridge

```
# brctl addif br0 tap0
```

- Acrescenta a interface de rede tap0 à brigde br0.
- Todas as interfaces contidas em uma bridge agem como em uma mesmo segmento de rede.
- Não é possível adicionar uma mesma interface em várias bridges.
- Quando uma interface é adicionada a bridge, ela leva um tempo para aprender a endereço Ethernet antes de começar a transmitir.
- Vendo as interfaces em uma bridge:

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.baac3fccc978       no              tap0
```

Usando a bridge

- Agora vamos utilizar a bridge. Primeiramente, temos que colocar as interfaces tap dentro da bridge:

```
# brctl addif br0 tap0
# brctl addif br0 tap1
```

- Feito isso, inicie duas máquinas virtuais, cada uma utilizando uma interface tap. Coloque endereços MAC diferentes para as interfaces de cada máquina virtual.
- Configure ambas as máquinas virtuais com IPs pertencentes à mesma rede, ex: 192.168.0.2/24 e 192.168.0.3/24

No HOST devemos iniciar a bridge

```
# ifconfig br0 up
```

- De dentro das máquinas virtuais, tente pingá-las entre si.

Acesso externo com a bridge+NAT

- Nesse momento as interfaces das VMs estão isoladas dentro da bridge, sem nenhuma possibilidade de acesso externo.
- Para que as VMs possam "sair", precisamos colocar um IP na própria bridge.

```
# ifconfig br0 192.168.0.1 up
```

- As bridges tem uma porta especial que pode receber um IP e está atrelada ao host, realmente como uma "saída".
- Tente agora, a partir do host, pingar uma das VMs.
- Se estiver tudo funcionando, nas VMs coloque o IP da 192.168.0.1 como rota padrão:

```
# route add default gw 192.168.0.1
```

- Lembre-se que deixamos o **ip_forward** a regra de **NAT do iptables** configurados, portanto de dentro das VMs já será possível acessar o mundo externo.
- Características dessa configuração:
  - Interfaces de rede das VMs estão no mesmo segmento de camada 2.
  - O tráfego entre as VMs acontece totalmente dentro do kernel do host.
  - Acesso de fora do host para dentro só é possível com regras de iptables.
  - Pode-se utilizar roteamento IP também, ao invés de NAT. (tente fazer como exercício)

Acesso ao segmento de rede físico com

bridges

- Uma outra opção de configuração é colocar as interfaces de rede virtuais em contato praticamente "direto" com o meio físico. Para isso, precisamos colocar a interface eth0 do host dentro da bridge.
- Tire o IP da bridge e tire o IP da interface eth0. **Antes, anote qual é o IP de sua máquina!**

```
# ifconfig br0 0.0.0.0
# ifconfig eth0 0.0.0.0
```

- Agora colocamos a interface eth0 na bridge e nela configuramos também o IP que estava em eth0.

```
# brctl addif br0 eth0
# ifconfig br0 10.1.1.X
```

- Vamos desativar o NAT e o ip_forward.

```
# iptables -t nat -F
# echo 0 > /proc/sys/net/ipv4/ip_forward
```

- Verifique a conectividade do host:

```
# ping 10.1.1.1
```

- Configure suas VMs com IPs da rede 10.1.1.0/24 e de dentro da VM tente pingar 10.1.1.1.

Mais sobre bridges

- Uma vez que a bridge está em funcionamento, o parâmetro showmacs mostra informações sobre os endereços de rede e de tráfego que está sendo encaminhado.

```
# brctl showmacs br0
port no mac addr                is local?       ageing timer
  3     00:16:3e:57:13:a5       no                18.28
  2     00:16:3e:57:20:a5       no                18.28
  1     00:16:3e:57:20:b5       no                59.78
```

- O tempo de envelhecimento (ageing time) é o número de segundos que um endereço MAC será mantido no banco de dados de encaminhamento, depois de ter recebido um pacote a partir deste endereço MAC.

- As entradas no banco de dados de encaminhamento são periodicamente apagadas para garantir que não ficarão para sempre.

Arquivos de configuração

- CentOS/RHEL

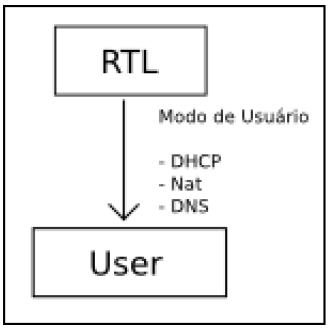
```
# cd /etc/sysconfig/network-scripts/
# cat ifcfg-br0
DEVICE=br0
ONBOOT=yes
BOOTPROTO=static
IPADDR=143.106.7.3
NETMASK=255.255.255.192
NO_ALIASROUTING=yes
GATEWAY=143.106.7.1
TYPE=Bridge

# cat ifcfg-eth0
# Intel Corporation 80003ES2LAN Gigabit Ethernet Controller (Copper)
DEVICE=eth0
ONBOOT=yes
TYPE=Ethernet
HWADDR=00:15:17:4E:CA:DE
BRIDGE=br0
BOOTPROTO=none
```

- Debian / Ubuntu

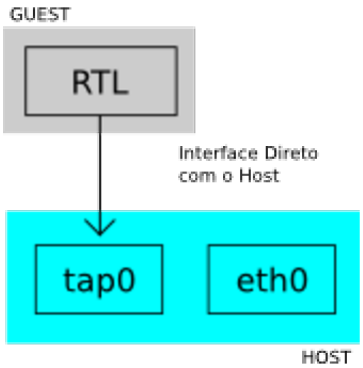
```
# cat /etc/network/interfaces
auto br0
iface br0 inet static
        address 192.168.2.50
        netmask 255.255.255.0
        gateway 192.168.2.1
        bridge_ports eth0
        bridge_stp off
        bridge_fd 0
```

Cenários

Modo usuário



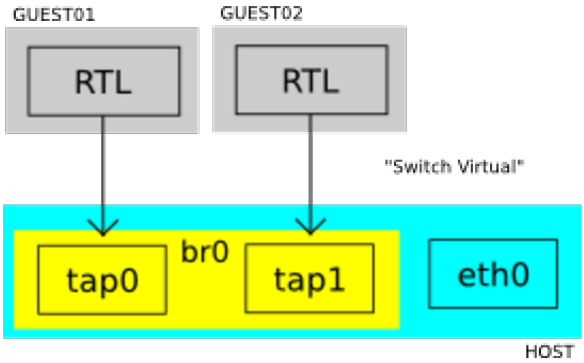
Acesso direto entre Host e Guest



Exemplo:

Host_eth0	143.106.16.1
tap0	192.168.0.1
Guest01	192.168.0.2

Acesso entre 2 Guest e roteamento através do Host

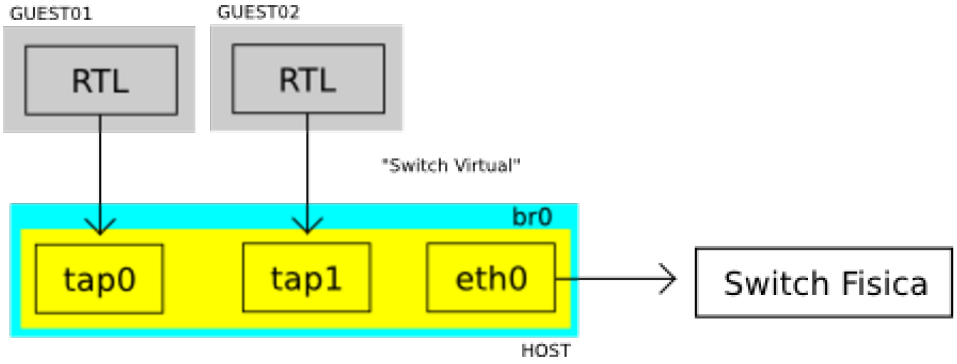


Exemplo:

Host_eth0	143.106.16.1
Bridge	192.168.0.1

Guest01	192.168.0.2
Guest02	192.168.0.3

Acesso dos 2 Guest e Host diretamente na rede física.



Exemplo:

Host_br0	143.106.16.1
Guest01	143.106.16.2
Guest02	143.106.16.3

Referências

- Esquema de rede interno do QEMU: http://people.gnome.org/~markmc/qemu-networking.html
- Gerador de endereços MAC: http://www.easyvmx.com/software/easymac.sh
- Documentação oficial sobre bridges no Linux: http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge
- Análise de performance de bridge do Linux: http://facweb.cti.depaul.edu/jyu/Publications/Yu-Linux-TSM2004.pdf
- Documentação do kernel sobre interfaces tap
- Interface tap centos: http://www.rootninja.com/create-a-network-bridge-for-virtual-machines

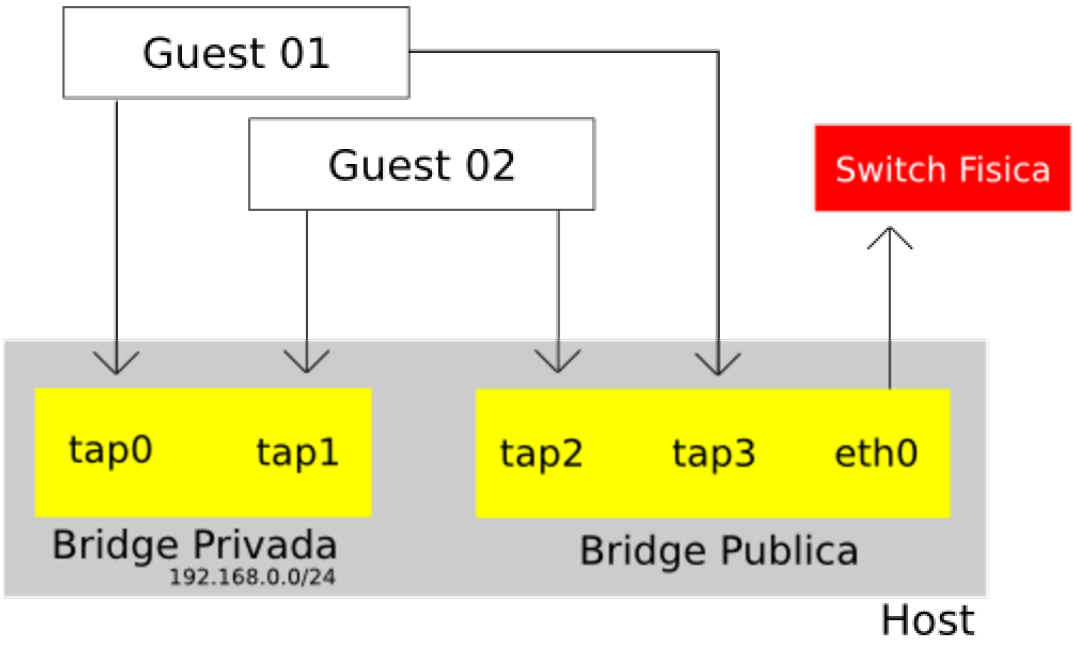
Aula 06 - Criar ambientes de rede seguindo especificação

Sobre

- Metodologia: atividade em laboratório
- Objetivos:
  - Configurar um ambiente de rede para hospedagem de máquinas virtuais.
  - Verificar a performance da rede virtual.

Aula

Configure suas máquinas virtuais de acordo com as regras abaixo:



- Cada máquina virtual deve ter duas interfaces de rede, chamaremos uma de pública e outra de privada.
- A interface pública da máquina virtual deve ter acesso ao barramento local do laboratório
- A interface privada deve ser colocada em uma rede virtual interna no host.
- Para as interfaces públicas use os IPs que foram alocados para seu host.
  - Caso queira mais um IP adicione +100 e/ou +200 no final do seu IP
- Para as interfaces privadas use a rede 192.168.0.0/24.
- O host deve ter um IP na rede privada e estar acessível aos guests

Faça um relatório no wiki do seu cenário, IPs, comandos e resultados obtidos.

Aula 07 - Dispositivos de discos

Sobre

- Objetivos:
  - Aprofundar o conhecimento sobre armazenamento virtualizado.
  - Questões de cache e otimização de acesso.
  - Administração de armazenamento.
  - Escalonadores de I/O.
  - Dando diferentes prioridades para máquinas virtuais.
  - Possíveis otimizações em sistema de arquivos.

Visão geral

- O host ou hypervisor:
  - Exporta discos virtuais para os guests
  - O guest usa-os como discos reais
- Os discos virtuais são mapeamentos para dispositivos reais
  - Discos inteiros, partições, volumes lógicos ou arquivos
  - Arquivos raw em um sistema de arquivos ou formatos de imagem

Virtual storage stack

Guest Filesystem
Guest Storage Driver
Storage hw emulation
Image format
Host filesystem
Host volume manager
Host storage driver

- Temos duas pilhas de armazenamento total no host e no guest
- Um sistema de arquivos potencialmente também nos dois
- Potencialmente também um formato de imagem (mini-sistema de arquivos)

Requisitos de Armazenamento

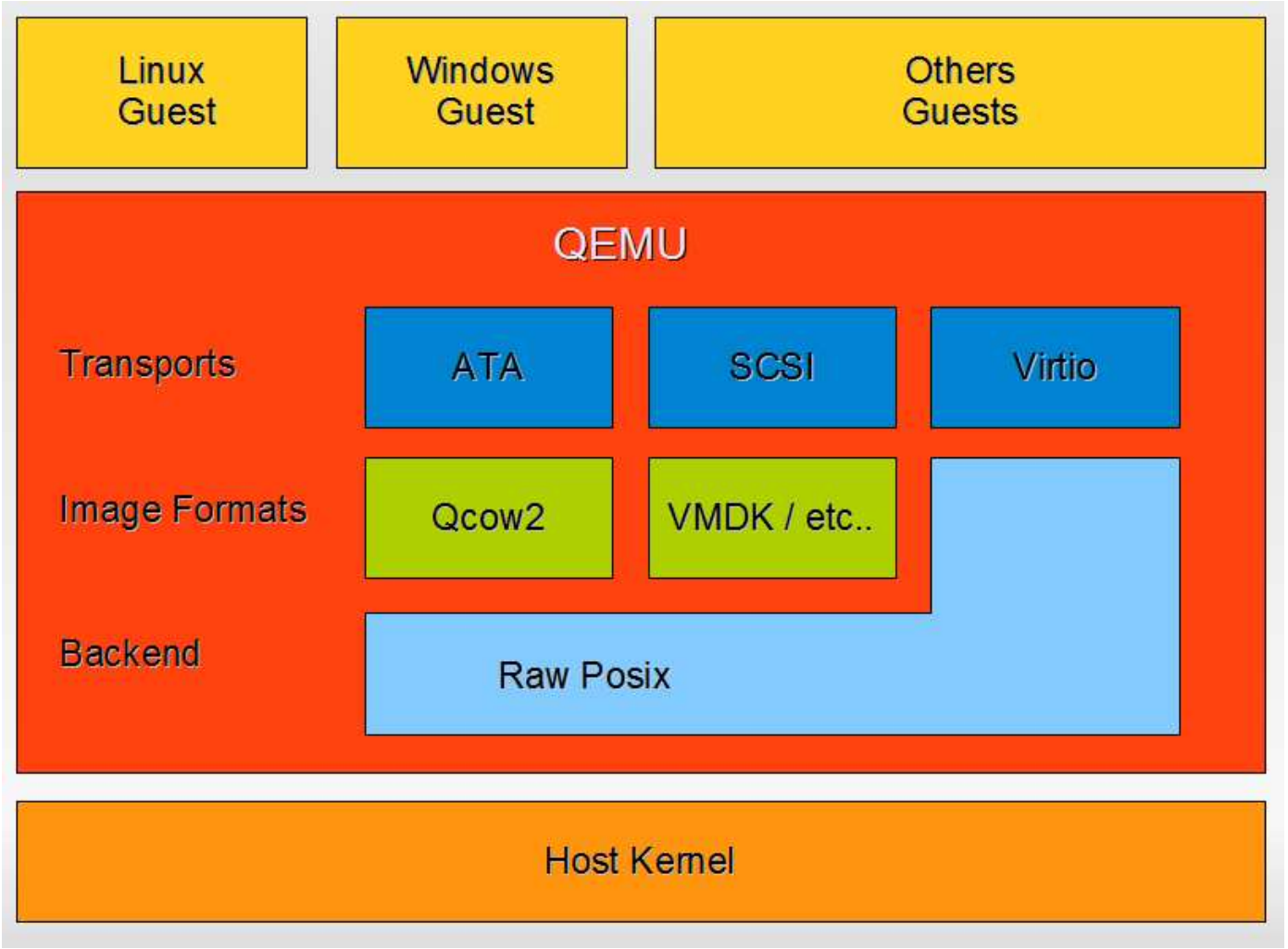
- Os requisitos de armazenamento tradicionais se aplicam:
  - A integridade dos dados: os dados devem realmente estar no disco quando o usuário/aplicação pedirem
  - Eficiência de espaço: queremos armazenar os dados do usuário/aplicação o mais eficientemente possível
  - Desempenho: fazer tudo o mais rápido possível
  - Capacidade de gestão: administrar diversos hosts

Requisitos - host

- É o lugar onde toda a inteligência fica
- Assegura a integridade dos dados

- Os dados estão realmente no disco quando o guest assume isso

Visão no QEMU/KVM



Storage Transports

- O QEMU fornece um simples controlador Intel ATA/IDE por padrão
- Funciona com quase todos os sistemas operacionais, porque é muito comum
- Alternativamente QEMU pode emular um controlador SCSI Symbios (ainda está melhorando)

Paravirtualização

- Fornecer interfaces melhores do que o hardware real
- Vantagem: deve ser mais rápido do que a virtualização/emulação completa
- Desvantagens: requer drivers especiais no guest

Paravirtualized storage transport

- QEMU fornece dispositivos virtualizados utilizando o framework VirtIO
- Fornece um simples driver de bloco

- Simples leitura/gravação de pedidos

Comando

```
# kvm -drive file=/dev/volumes/zacarias,media=disk,index=0,boot=on,if=virtio
# kvm -drive file=/dev/volumes/zacarias,media=disk,index=0,boot=on,if=ide
# kvm -drive file=/dev/volumes/zacarias,media=disk,index=0,boot=on,if=scsi
```

Formatos de imagem

- Usuários querem uma recursos de gerencia de volumes em arquivos de imagem
- Snapshots
- Criptografia
- Compressão
- Snapshots também precisam armazenar metadados adicionais (memória, estado, etc)

Formatos de imagem

O QEMU suporta diversos formatos de imagem:

- cow - User Mode Linux
- vpc - Microsoft Virtual PC
- vmdk - VMware
- vdi - VirtualBox
- Bochs
- Parallels
- dmg – MacOS filesystem
- Outros.

Formatos de imagem - qcow2

- O qcow2 foi o formato de imagem principal do QEMU
- Suporta snapshots
- Suporta criptografia e compressão
- É suportado pela Red Hat e tem recebido muitas melhorias nos últimos tempos

Backends não baseados em imagem

- curl: permite a utilização de imagens a partir da Internet através de conexões HTTP e FTP.
- nbd: acesso direto a servidores nbd.
- vvfat: permite exportar sistemas de arquivos FAT
- Baseados em cluster:
  - Sheepdog
  - CEPH

qemu-img

- O canivete suíço
- Permite criar, manipular, converter e outras operações em todos os formatos de imagem suportados.

```
# qemu-img create -f qcow2 -o preallocation=metadata vdisk.qcow2 50G
Formatting 'vdisk.qcow2', fmt=qcow2 size=53687091200 encryption=off cluster_size=0
preallocation='metadata'

# qemu-img info -f qcow2 vdisk.qcow2
image: vdisk.qcow2
file format: qcow2
virtual size: 50G (53687091200 bytes)
disk size: 7.9M
cluster_size: 65536
```

- Convertendo imagem raw para qcow2

```
# qemu-img convert -O qcow2 imagem.raw novaimagem.qcow2
```

Comando

```
# kvm -drive file=vdisk.qcow2,format=qcow2,media=disk,index=0,boot=on,if=virtio
# kvm -drive file=/dev/volumes/zacarias,format=raw,media=disk,index=0,boot=on,if=ide
```

Integridade de dados no QEMU - caching

- cache=none
  - usa O_DIRECT I/O que ignora o cache do host
- cache=writethrough
  - usa O_SYNC I/O que garante a confirmação da escrita no disco
- cache=writeback
  - usa o buffer de I/O do host

Integridade de dados - writethrough

- Este modo é o mais seguro
- Não há mais caches volátil no host
- É lento

Integridade de dados - writeback

- Quando o guest escreve dados, eles simplesmente são colocados no pagecache host
- Não há garantia de que ele realmente foi para o disco
  - O que é realmente muito semelhante à forma como os discos modernos funcionam
- O guest deverá emitir um comando flush para se certificar que os dados foram para o disco
  - Similar a discos reais modernos com cache de writeback
- E o host precisa implementar o comando flush e anunciá-lo
  - IDE e VirtIO só muito recentemente receberam suporte

A integridade dos dados - none

- Transferência direta para o disco deveria implicar que é seguro
- Só que ela não é:
  - Não se garante que caches de disco são liberados
  - Não se dá nenhuma garantia sobre metadados

- Também precisa de um flush cache explícito

Comando

```
# kvm -drive
file=vdisk.qcow2,format=qcow2,media=disk,index=0,boot=on,if=virtio,cache=writeback
```

Administração

Ferramentas utilizadas normalmente para administração de volumes, partições e sistemas de arquivos podem e devem ser utilizadas para a administração e manutenção de armazenamento virtual.

Modelos de armazenamento

- LVM
  - Cada LV possui um disco virtual de uma VM
- Partições
  - Pode-se exportar uma partição toda para uma VM
- Arquivos de imagem

Backend de armazenamento

- Centralizado em um storage
  - iSCSI
  - Fibre Channel
  - AoE
- Qualquer dispositivo de bloco que esteja acessível no host

Backend de armazenamento

- Filesystem compartilhado
  - NFS
  - GFS
  - OCFS

Ferramentas

- kpartx, fdisk, gparted, lvscan, fsck, etc.
- dstat, iotop, htop

Schedullers

- O Linux oferece quatro schedulers de I/O, também conhecido como elevators:
  - Noop
  - Anticipatory
  - Completely Fair Queuing (CFQ)
    - Divide a banda de I/O disponível, mantendo filas de requisições por processo.
  - Deadline
    - Submete requisições de I/O baseando-se no tempo em que estão esperando na fila, garantido um

início de serviço.
- Cada agendador é eficaz em um cenário

Trocando o scheduller

```
 # cat /sys/block/sda/queue/scheduler
noop deadline [cfq]

# echo deadline > /sys/block/sda/queue/scheduler

# cat /sys/block/sdb/queue/scheduler
noop [deadline] cfq
```

- Colocar como parâmetro de boot elevator=deadline
- Quando CFQ, usar ionice para dar prioridade diferenciada

Material de referência

- The KVM/gemu Storage Stack: http://events.linuxfoundation.org/images/stories/slides/jls09/jls09_hellwig.odp
- I/O Schedullers: http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/index.jsp?topic=/liaat/liaatbpscheduleroverview.htm
- http://duartes.org/gustavo/blog/post/page-cache-the-affair-between-memory-and-files
- http://www.ibm.com/developerworks/linux/library/l-virtio/

Base images e rebasing

- http://www.linux-kvm.com/content/how-you-can-use-qemukvm-base-images-be-more-productive-part-1
- http://www.linux-kvm.com/content/be-more-productive-base-images-part-2
- http://www.linux-kvm.com/content/be-more-productive-base-images-part-3
- http://outlyer.net/howtos-linux/kvm-qemu-notes/
- http://kacper.blog.redpill-linpro.com/archives/82

Benchmarks

- KVM I/O slowness on RHEL 6: http://www.ilsistemista.net/index.php/virtualization/11-kvm-io-slowness-on-rhel-6.html?showall=1
- Benchmark de HDs: http://techreport.com/articles.x/20562
- IO Containment: http://www.kernel.org/doc/ols/2008/ols2008v1-pages-151-162.pdf
- VM I/O benchmarks: http://learnitwithme.com/?p=198
- Rackspace Cloud Servers versus Amazon EC2: Performance Analysis: http://www.thebitsource.com/featured-posts/rackspace-cloud-servers-versus-amazon-ec2-performance-analysis/

Material para se aprofundar

- Which I/O controller is the fairest of them all?: http://lwn.net/Articles/332839/
- I/O Topology: http://www.kernel.org/doc/ols/2009/ols2009-pages-235-238.pdf
- The 2010 Linux Storage and Filesystem Summit, day 1: http://lwn.net/Articles/399148/

- The 2010 Linux Storage and Filesystem Summit, day 2: http://lwn.net/Articles/399313/
- I Will Keep Saying It: Align Your Partitions: http://lonesysadmin.net/2010/03/30/i-will-keep-saying-it-align-your-partitions/
- VMware I/O Problems: http://lonesysadmin.net/2006/05/20/vmware-io-problems/
- Controle de I/O com cgroups e libvirt: http://comments.gmane.org/gmane.comp.emulators.libvirt/32332
- High Performance VMM-Bypass I/O in Virtual Machines: http://www.cc.gatech.edu/classes/AY2007/cs8803hpc_fall/papers/dk-vmmio.pdf

Aula 08 - Dispositivos de rede

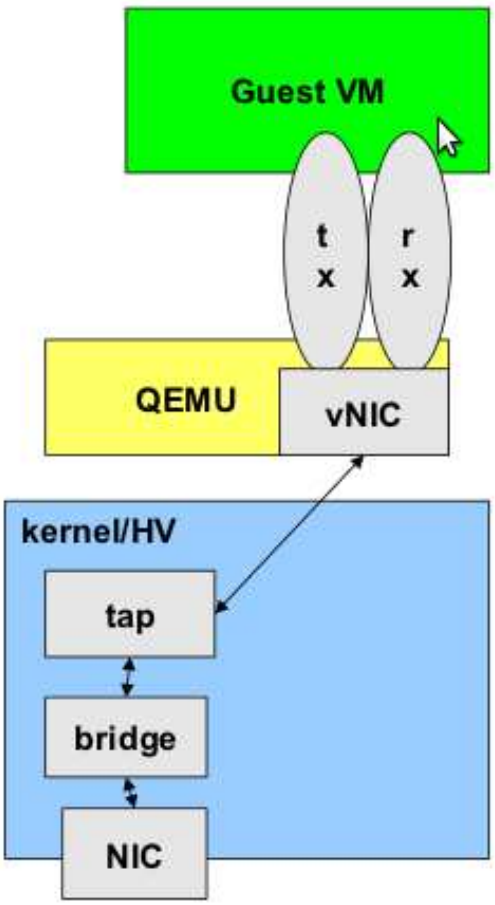
Sobre

- Objetivos:
  - Implementar mecanismos de segurança e isolamento de redes
  - Ajustes de performance para melhor desempenho em redes Gigabit
  - Introdução a redes virtuais

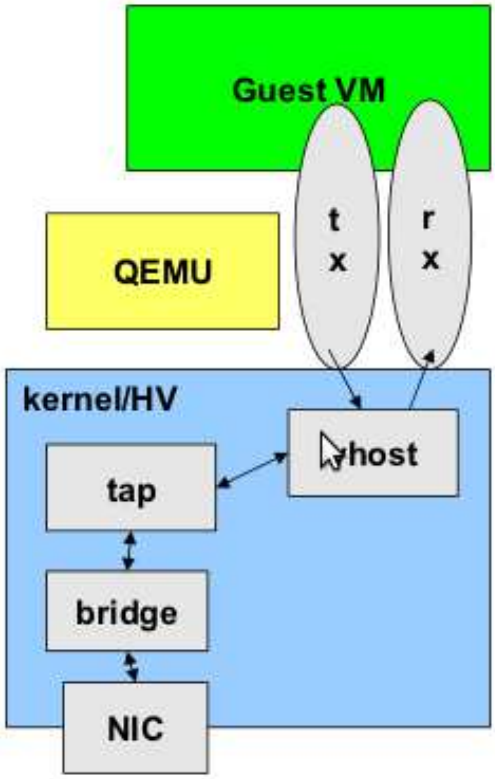
Ajustes de performance

- Comparativo de performance com iperf
- iperf host->guest (com e sem vhost)
- iperf guest->guest (com e sem vhost)

Sem vhost-net



Com vhost-net

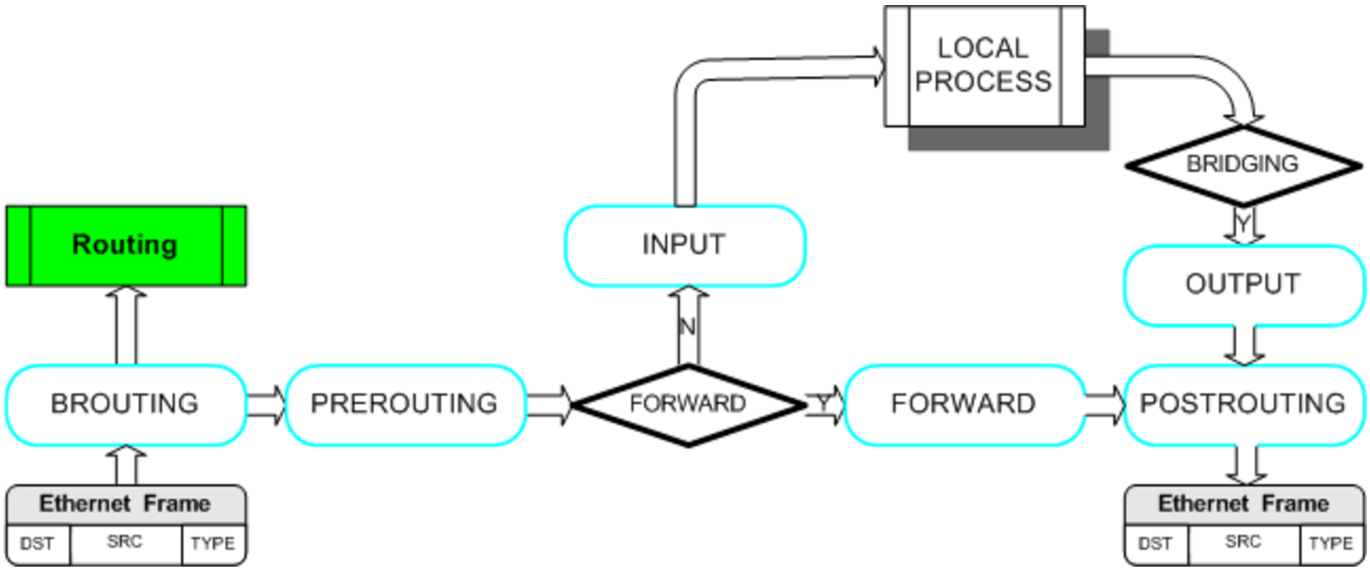


Segurança em bridges

- Camada de enlace
- Riscos conhecidos:
  - ARP spoofing
  - MAC flooding

Para proteger, usamos o ebtables:

```
# ebtables -P FORWARD DROP
# ebtables -A FORWARD -s 00:00:00:00:00:01 -i tap1 -j ACCEPT
# ebtables -A FORWARD -s 00:00:00:00:00:02 -i tap2 -j ACCEPT
# ebtables -A FORWARD --log
```



Deixando o host incomunicável com a bridge:

```
# ebtables -P OUTPUT DROP
# ebtables -A OUTPUT --log
```

- IDS: http://www.linuxsecure.de/index.php?action=90
- http://ebtables.sourceforge.net/br_fw_ia/br_fw_ia.html
- http://ebtables.sourceforge.net/misc/ebtables-faq.html

Cuidados com iptables

- Processamento de iptables dentro das bridges
- O módulo conntrack funciona globalmente

```
/etc/sysctl.conf:
 net.bridge.bridge-nf-call-ip6tables = 0
 net.bridge.bridge-nf-call-iptables = 0
 net.bridge.bridge-nf-call-arptables = 0
```

PCI passthrough

- Passar um dispositivo diretamente para o guest.
- Apenas um guest pode ter acesso.
- Pode ser útil em casos de muita sensibilidade à latência como VoIP, ou hardware especializado.

```
# Parâmetro intel_iommu=on no boot
# lspci (identificar endereço no barramento)
```

```
# lspci -n (pegar o ID do dispositivo)
# echo "10ec 8136" > /sys/bus/pci/drivers/pci-stub/new_id
# echo "0000:04:00.0" >
/sys/bus/pci/devices/0000\:04\:00.0/driver/unbind
# echo "0000:04:00.0" > /sys/bus/pci/drivers/pci-stub/bind
# hvm -device pci-assign,host=04:00.0
```

VMDq (Virtual Machine Device Queues

- Ordenador de camada 2 que envia o quadro para filas específicas associadas com uma interrupção específica para um guest.
- Intel VMDq Demonstration - Live

SR-IOV (Single Root I/O Virtualization)

- Permite que um dispositivo PCIe pareça ser múltiplos dispositivos físicos separados.
- Placas de rede com múltiplas portas
- Cada dispositivo pode ser passado para um guest
- Para saber mais:
  - http://blog.scottlowe.org/2009/12/02/what-is-sr-iov/
- Intel SR-IOV Explanation

VLAN 802.1q (VLAN taggeada)

- Configuração usando **vconfig** ou **ip** no guest.
- Importante: manter uma bridge por VLAN ID.
- Evitar misturar VLANs na mesma bridge.

Inter-conexão de bridges

- Tunelando bridges gretap: http://benoit.papillault.free.fr/blog/?p=54
- GRE: http://en.wikipedia.org/wiki/Generic_Routing_Encapsulation

Performance e QoS

- MTU (jumbo frames)
- Tuning 10Gb network cards on Linux: http://www.kernel.org/doc/ols/2009/ols2009-pages-169-184.pdf
- txqueuelen

- ethtool -K eth0
- Isolamento de banda utilizando Xen+tc (serve para KVM também) http://horms.net/projects/xen-bw-isolation/2010-08/bw.en.pdf
- cGroups

Switches e redes virtuais

- http://openvswitch.org/
- http://www.openflow.org/
- http://vde.sourceforge.net/

Referências

- Cenários: http://blog.klauskiwi.com/wp-content/uploads/2010/08/KVM-Security_en.pdf
- Artigo muito bom sobre detalhes internos de interfaces TAP: http://www.linuxfi.com.br/artigos/tun_tap.pdf
- Disable net.bridge.bridge-nf-call-*tables by default: https://bugzilla.redhat.com/show_bug.cgi?id=512206
- VLAN 802.1q, seguir thread: http://www.mail-archive.com/bridge@lists.linux-foundation.org/msg01269.html

Tópicos para estudo

- JLS2009: Generic receive offload: http://lwn.net/Articles/358910/
- Receive packet steering: http://lwn.net/Articles/362339/
- Rps: Receive packet steering: http://lwn.net/Articles/361440/
- Multiqueue networking: http://lwn.net/Articles/289137/
- http://virt.kernelnewbies.org/MacVTap
- Provide a zero-copy method on KVM virtio-net. http://lwn.net/Articles/407939/
- Linux Containes and Networking: http://blog.flameeyes.eu/2010/09/04/linux-containes-and-networking
- Comandos `ip` equivalentes ao ifconfig: http://jengelh.medozas.de/2008/0219-ifconfig-sucks.php

Aula 09 - Atividade prática

- Objetivos:
  - Utilizando os conhecimentos da aula anterior, montar um comparativo de performance sobre diferentes configurações de discos virtuais.
  - Implementar ajustes de prioridade de I/O.

Parte 1: benchmark de discos virtuais

- Comandos para o teste:

```
# dd if=/dev/zero of=dados02.dat bs=4k count=153600
# dd if=/dev/zero of=dados02.dat bs=4k count=153600 oflag=direct
```

Obter os dados de performance para as seguintes combinações de discos virtuais:

	LVM raw + VirtIO	imagem qcow2 + VirtIO
cache=writeback	-	-
cache=none	-	-
	LVM raw + IDE	imagem qcow2 + IDE
cache=writeback	-	-
cache=none	-	-

- Utilize o qemu-img para converter uma imagem de raw para qcow2

Parte 2: ajustando prioridades

- Inicie duas máquinas virtuais quaisquer, ambas utilizando o mesmo tipo de armazenamento.
- Utilizando o comando ionice, dê mais privilégio de I/O para uma VM e menos para a outra.
- Inicie as máquinas virtuais com cache=none e com 512MB de memória RAM.
- Dentro de cada máquina VM, rode o comando **dd if=/dev/zero of=dados.dat bs=4k count=1024**
- No host, utilizando iotop, relate os resultados de performance que você obteve e quanto tempo cada **dd** levou em cada guest.
- Apresente comandos e informações do cenário que você montou.

Aula 10 - Gerenciamento de memória

Sobre

- Objetivos:
  - Entender como o Linux gerencia memória e como o KVM tira vantagem disso.
  - Compreender e implementar ajustes finos de memória para maior performance.
  - Redução e aumento de memória durante a execução.

Aspectos sobre Linux e memória

- A maioria dos sistemas operacionais e aplicativos não usam 100% da memória RAM disponível o tempo todo.
- O kernel Linux aloca memória para cada processo quando o processo solicita mais memória.
- O kernel Linux faz swap de memória raramente usada da memória física para a área de swap.
- Quando a memória física é totalmente utilizada ou um processo fica inativo por algum tempo, o Linux move a memória de um processo para o swap.
- O swap é normalmente uma partição/volume que o Linux usa para aumentar a memória virtual.
- Swap é significativamente mais lento que RAM, devido à transferência e tempos de resposta dos discos rígidos e drives de estado sólido.

Gerenciamento de memória do Linux

- Comandos básicos
  - ps
  - top
  - vmstat
- Parâmetros do /proc/meminfo
-  The Linux Page Cache and pdflush: Theory of Operation and Tuning for Write-Heavy Loads

Aspectos sobre KVM e memória no Linux

- KVM pode alocar mais memória para os clientes virtuais do que fisicamente disponível no host.
- Os guests são apenas processos para o Linux.
- Os guests não tem blocos de RAM física dedicados a eles.

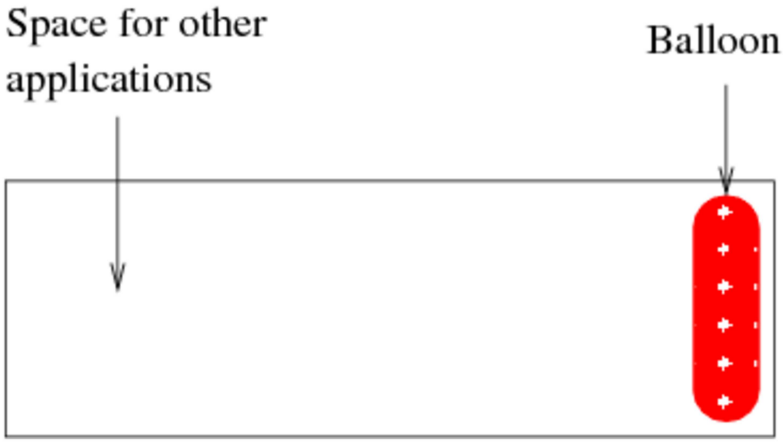
- KVM alocada memória quando solicitado pelo sistema operacional do guest.
- O cliente requer apenas um pouco mais de memória física do sistema operacional virtualizado relatórios como usados.
- Usando swap diminui a quantidade de memória real necessária pelos guests.
- Como as máquinas virtuais KVM são processos, memória subutilizadas ou ociosas de guests virtualizados é movida por padrão para swap. (swappiness)
- O total de memória usada pelos clientes pode ser maior que a memória física disponível (overcommit).
- O overcommitting requer espaço de swap suficiente para todos os guests e todos os processos.
- Sem espaço de swap suficiente para todos os processos na memória virtual do processo pdflush, o processo de limpeza, começa.

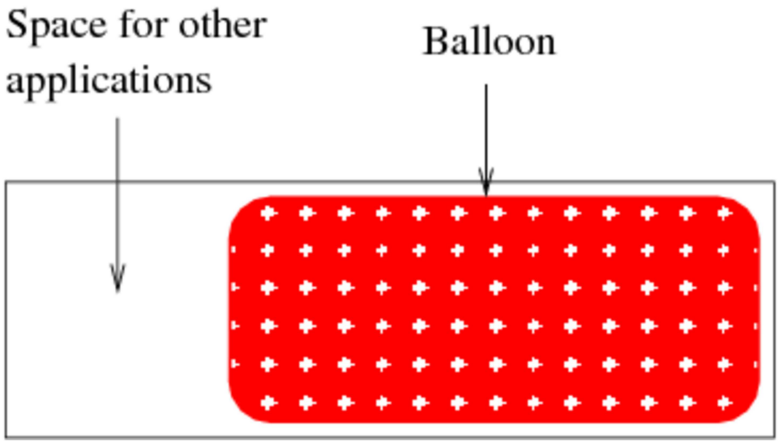
Tipos de overcommit

- Swap
  - Esta é a forma clássica de apoio ao overcommit, o host escolhe algumas páginas de memória de um dos guests e as envia para o disco. Se um hóspede exige memória que tenha ido para a swap, o host trás de volta do disco.
- Balões (ballooning)
  - Com o balão, o host e o guest cooperam em que páginas serão liberadas. É de responsabilidade do guest escolher a página e liberá-la, se necessário.
- Compartilhamento de páginas
  - O hypervisor olha para páginas de memória que possuem dados idênticos, estas páginas são fundidas em uma única página, que é marcada apenas para leitura. Se um cliente escreve em uma página compartilhada, ela é "descompartilhada" antes de conceder ao guest a gravação.
- Live-migration
  - O hypervisor move um ou mais guests para um host diferente, liberando a memória.

Balloning

- Balonismo é bastante eficiente, já que depende do guest para liberar a memória. Muitas vezes, os clientes podem simplesmente encolher seu cache para liberar memória, que pode ter um impacto muito baixo no guest. O problema com o balonismo é que ele depende da cooperação dos guests, o que reduz a sua confiabilidade.





- Como funciona:

```
# kvm -balloon virtio

ou

# kvm -device virtio-balloon-pci
```

- No guest:

```
# lspci | grep RAM
00:04.0 RAM memory: Red Hat, Inc Virtio memory balloon
modprobe virtio_balloon
```

- Acessar o monitor da VM:

```
(qemu) info balloon
balloon: actual 1024
(qemu) balloon 512
(qemu) info balloon
balloon: actual 512
```

- Muito bom: http://rwmj.wordpress.com/2010/07/17/virtio-balloon/

- Agente de gerenciamento de balão: http://aglitke.wordpress.com/2011/03/03/automatic-memory-ballooning-with-mom/

Swapping

Swap não depende do cliente, por isso é mais confiável do ponto de vista do host. No entanto, o host tem menos conhecimento do que o guest sobre a memória do guest, assim que swap é menos eficaz que o balão.

- Parâmetros de ajuste de swap:

```
# sysctl -w vm.swapiness=100 (quanto maior, mais agressivamente o
kernel fará swap)
# sysctl -w vm.overcommit_memory=2 (0 heurístico, 1 overcommit sem
limit, 2 não fazer overcommit)
# sysctl -w vm.overcommit_ratio=50 (quando overcommit_memory=2)
```

Quando overcommit_memory=2: **memória_total = swap + (memória_física * (overcommit_ratio / 100))**

Compartilhamento de páginas

Compartilhamento de páginas depende do comportamento dos hosts indiretamente. Quando os guests executam aplicativos semelhantes, o host vai atingir uma proporção elevada de compartilhamento.

- KSM - Kernel Samepage Merging
- Increasing memory density by using KSM
- Usando no host:

```
# echo 1 > /sys/kernel/mm/ksm/run (demora alguns segundos para os
primeiros resultados)
# cat /sys/kernel/mm/ksm/pages_sharing (multiplicar por 4KB para total
compartilhado)
5487
```

Estratégia do KVM

Então kvm usa uma estratégia mista: compartilhamento de página e balões são usados como métodos preferenciais para a overcommit de memória uma vez que são eficientes. Livre migration é usada para equilibrar a longo prazo dos requisitos de memória e recursos. Swap é usada como último recurso.

Hugepages

Hugepages permite uso de páginas de 4MB em sistemas x86 de 32 bit e 2MB de 64bit. Assim, há um uso mais eficiente da memória quando é necessário um grande volume de memória. Outra característica também é que esse tipo de páginas não pode ser movida para a área swap.

- Como configurar hugepages: http://www.freedominterface.org/2010/09/27/hugepages/
- KVM e hugepages: http://www.linux-kvm.com/content/get-performance-boost-backing-your-kvm-guest-hugetlbfs
- Detalhes técnicos de hugepages e TLB: http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/index.jsp?topic=/liaat/liaattunhp.htm
- Não dá pra misturar KSM e hugepages: http://us.generation-nt.com/answer/ksm-hugepages-help-196557231.html

Novidades

- KVM: Add host swap event notifications for PV guest: http://lwn.net/Articles/409961/

Para estudo aprofundado

- Kernel Virtualization Optimizations for KVM (muita coisa!): http://www.redhat.com/promo/summit/2010/presentations/summit/decoding-the-code/thurs/jshaksho-310-420-performance/larry_shak_perf_summit2010_v2.pdf
- Paginação de memória em hardware (NPT/EPT): http://avikivity.blogspot.com/2008/04/paravirtualization-is-dead.html
- Documentação definitiva de hugepages: http://lwn.net/Articles/374424/
- What every programmer should know about memory
  - Parte 1: http://lwn.net/Articles/250967/
  - Parte 2: http://lwn.net/Articles/252125/
  - Parte 3: http://lwn.net/Articles/253361/
  - Parte 4: http://lwn.net/Articles/254445/
  - Parte 5: http://lwn.net/Articles/255364/
  - Parte 6: http://lwn.net/Articles/256433/
  - Parte 7: http://lwn.net/Articles/257209/

Aula 11 - Gerenciamento de CPU e tempo

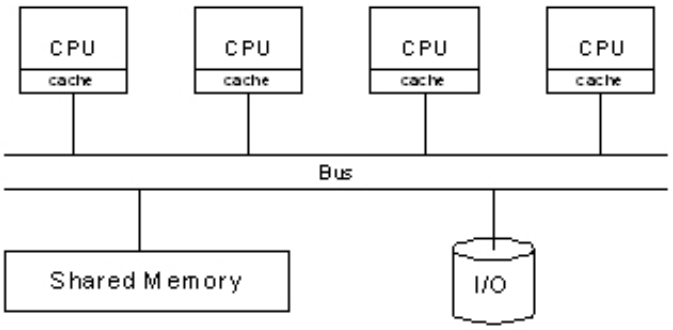
Sobre

- Objetivos:
  - Entender o funcionamento de processadores multi-core e as implicações para máquinas virtuais.
  - Ajustar prioridades de execução para diferentes máquinas virtuais.
  - Entender como manter e ajustar relógios virtuais.
  - Ajustar o host para balanceamento de IRQs e melhorar performance.

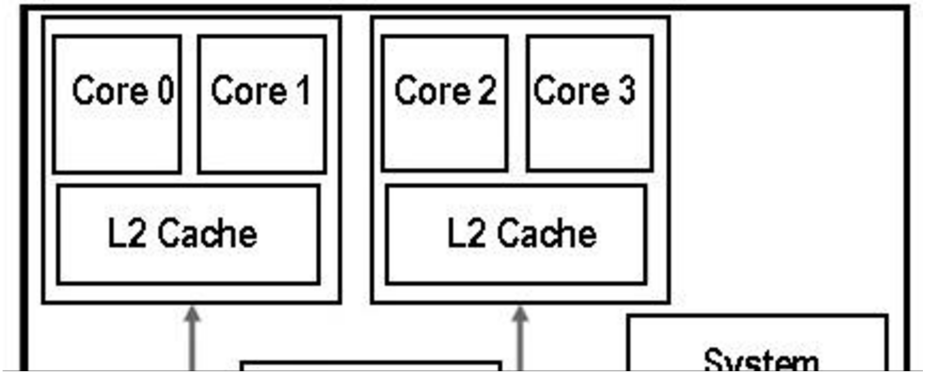
Overcommit de CPU

- O KVM suporta overcommitting de CPUs virtuais.
- CPUs virtualizadas podem ser sobrecarregadas na medida em que os guests tenham suas demandas atendidas.
- CPUs virtualizados estão utilizam melhor overcommit quando cada guest virtualizado tem apenas uma única CPU.
- O escalonador Linux é muito eficiente.
- Você não pode overcommit guests SMP em mais do que o número de núcleos físicos. Por exemplo, um guest com quatro vCPUs não deve ser executado em uma máquina com um processador de dois núcleos.
- Overcommitting do número de CPUs físicas em relação ao número de CPUs virtuais causa degradação significativa no desempenho.

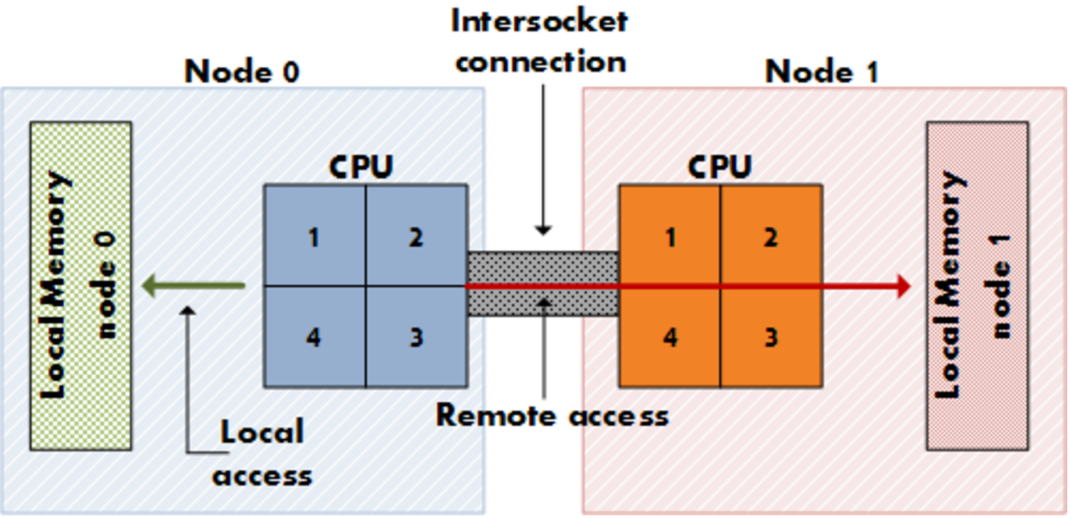
Arquitetura SMP



Arquitetura Multicore



Arquitetura NUMA



Topologia

```
# cat /proc/cpuinfo
processor       : 0 <logical cpu #>
physical id     : 0  <socket #>
siblings        : 16 <logical cpus per socket>
core id         : 0  <core # in socket>
cpu cores       : 8  <physical cores per socket>

# cat /sys/devices/system/node/node*/cpulist
node0: 0-3
node1: 4-7
```

Distribua carga de IRQ

- IRQ affinity: aliviar tratamento de interrupções e distribuir carga entre as CPUs.

```
# cat /proc/interrupts
# echo 2 > /proc/irq/217/smp_affinity
```

- http://www.alexonlinux.com/smp-affinity-and-proper-interrupt-handling-in-linux
- IRQ Balance: http://www.irqbalance.org

```
# irqbalance --debug
IRQ delta is 0
IRQ delta is 0, switching to power mode
Package 0:  cpu mask is 0000000f (workload 10901)
        Cache domain 0: cpu mask is 00000003  (workload 179)
                CPU number 0  (workload 120)
                CPU number 1  (workload 176)
                  Interrupt 21 (ethernet/55)
          Interrupt 22 (storage/0)
          Interrupt 18 (storage/0)
          Interrupt 30 (legacy/0)
        Cache domain 2: cpu mask is 0000000c  (workload 280)
                CPU number 2  (workload 120)
                CPU number 3  (workload 120)
          Interrupt 28 (storage/158)
          Interrupt 16 (legacy/0)
  Interrupt 24 (other/4097)
  Interrupt 25 (other/3155)
  Interrupt 27 (other/1654)
  Interrupt 26 (other/1651)
  Interrupt 4 (other/0)
```

Configurando uma CPU específica no KVM

```
# kvm -cpu ?
x86           [n270]
x86         [athlon]
x86       [pentium3]
x86       [pentium2]
x86        [pentium]
x86            [486]
x86        [coreduo]
x86          [kvm32]
x86         [qemu32]
x86          [kvm64]
```

```
x86       [core2duo]
x86         [phenom]
x86         [qemu64]

# kvm -cpu host

# kvm -smp 2
```

Ajustando o uso de CPU de VMs

- Prioridade:
  - nice
  - renice
  - chrt (específico do Linux)
- Posicionamento de vCPU em CPU real:
  - taskset (multicore e SMP): Running your VM on specific CPUs: http://www.linux-kvm.com/content/tip-running-your-vm-specific-cpus
  - numactl

Ligando e desligando CPUs

- Desligando uma CPU:

```
echo 0 > /sys/devices/system/cpu/cpu1/online
```

- Ligando:

```
echo 1 > /sys/devices/system/cpu/cpu1/online
```

Relógio virtual

- Fontes de tempo:

```
# cat /sys/devices/system/clocksource/clocksource0/current_clocksource
kvm-clock|tsc|hpet|acpi_pm

# dmesg | egrep -i "(time|clock|tsc)"
```

- NTP
  - Se usar kvm-clock, o relógio do guest acompanha. Instalar daemon NTP somente no host.
  - Se o guest não suporta kvm-clock, colocar NTP no guest pois o clock varia e

muito.

- Desabilite o hwclock (Don't run hwclock on guests running kvmclock)

```
# cd /sbin/
# ls -l hwclock*
# mv hwclock hwclock.dist
# touch hwclock
# chmod +x hwclock
```

TSC (timestamp counter)

- Flags na cpu: tsc, nonstop_tsc, constant_tsc
- Se CPU não tem nonstop_tsc:
  - Mensagem do kernel: Marking TSC unstable due to TSC halts in idle
- O kernel tentatá usar HPET ou ACPI se o TSC não for confiável

HPET (High Precision Event Timer)

- Guests que usam HPET consomem **MUITA** CPU inutilmente, devido ao alto número de interrupções.
- Alguns guests podem suportar TSC e HPET, mas não usam TSC preferencialmente. (CentOS 5, usar **kvm -no-hpet**)
  - http://home.coming.dk/index.php/don-t-emulate-hpet-with-kvm
- Mais referências:
  - Time and KVM - best practices: http://kerneltrap.org/mailarchive/linux-kvm/2010/3/21/6259882/thread
  - Timekeeping Virtualization for X86-Based Architectures

Referências

- Otimização de desktops com kernel Linux no /proc e /sys
- http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization/chap-Virtualization-KVM_guest_timing_management.html
- Conselhos preciosos sobre NUMA: http://permalink.gmane.org/gmane.comp.emulators.kvm.devel/56323
- Controlling guest CPU & NUMA affinity in libvirt with QEMU, KVM & Xen
- Why My Two vCPU VM is Slow: http://lonesysadmin.net/2008/04/22/why-my-two-vcpu-vm-is-slow/

- blog com vários benchmarks: http://vmstudy.blogspot.com/
- Best practices for KVM: http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaat/liaatbestpractices_pdf.pdf
- http://kerneltrap.org/mailarchive/linux-kvm/2010/3/14/6259558/thread

Extra

- virtio-serial: http://www.linux-kvm.org/page/VMchannel_Requirements
- Watchdog: http://rwmj.wordpress.com/2010/03/03/what-is-a-watchdog/
- kvm: the Linux Virtual Machine Monitor: http://www.kernel.org/doc/ols/2007/ols2007v1-pages-225-230.pdf

Aula 12 - Atividade prática

Parte 1: Testes de performance de CPU

- Inicie uma máquina virtual com 1GB de RAM, qualquer formato de imagem, VirtIO para discos e rede e execute o comando abaixo:

```
# wget http://www.ic.unicamp.br/~miguel/lib.tar
# time gzip -c lib.tar > /dev/null
```

- Agora, deslique a VM e execute-a acrescentando o parâmetro **-cpu host** na linha de comando do KVM. Execute novamente o comando **gzip**.
- Execute o mesmo comando, só que diretamente no host.

	CPU genérica	CPU do host	No host
Tempo			

Parte 2: Testes de performance de CPU + KSM + prioridades

Iniciando as máquinas virtuais

- Ative o KSM.

```
echo 1 > /sys/kernel/mm/ksm/run
```

- Dê boot em 6 máquinas virtuais, cada uma com 1GB de RAM, 2 processadores, qualquer formato de imagem, VirtIO para discos e rede.
- Dependendo dos resultados obtidos no experimento anterior, compensa utilizar **-cpu host** novamente?
- Ative o daemon ssh nas máquinas virtuais.

Configurando o ssh no host e VMs

- Instale o pacote pssh no host:

```
# aptitude install pssh
```

- Gere uma chave e copie-a para as VMs:

```
# ssh-keygen -t rsa
# ssh-copy-id -i ~/.ssh/id_rsa.pub root@VM1
# ssh-copy-id -i ~/.ssh/id_rsa.pub root@VM2
# ssh-copy-id -i ~/.ssh/id_rsa.pub root@VM3
# ssh-copy-id -i ~/.ssh/id_rsa.pub root@VM4
# ssh-copy-id -i ~/.ssh/id_rsa.pub root@VM5
# ssh-copy-id -i ~/.ssh/id_rsa.pub root@VM6
```

- Coloque em um arquivo texto os IPs das máquinas virtuais, um por linha. Use o seguinte comando para testar o acesso com o **parallel-ssh**:

```
# parallel-ssh -i -l root -h ips.txt -t -1 "uptime"
```

Ajustando as prioridades das VMs

- Escolha duas VMs e dê prioridade máxima a elas:

```
# renice -19 -p $PID
```

Executando o teste

- Baixe o arquivo lib.tar da parte um da aula em todas as VMs e a partir do host execute o comando:

```
# parallel-ssh -i -l root -h ips.txt -t -1 "time gzip -c /root/lib.tar
> /dev/null"
```

Anote seus resultados e quais foram as máquinas com maior prioridade.

Aula 13 - Tópicos

- Objetivos:
  - Apresentar dicas sobre assuntos variados ainda não abordados no curso.

Comandos úteis do monitor

- drive_add / drive_del
- netdev_add / netdev_del
- device_add / device_del
- netdev_add / netdev_del (Antigamente: host_net_add / host_net_remove)
- info block
- info blockstats
- info network / (No futuro pode ser info netdev)
- info qtree
- info snapshots
- savevm / loadvm
- stop / cont

Rede modernoza

- MacVLAN: http://jim.studt.net/depository/index.php/notes-on-linux-s-macvlan-module
- MacVTap: http://virt.kernelnewbies.org/MacVTap
- http://kerneltrap.org/mailarchive/linux-netdev/2009/3/7/5116004

Linux AIO

- O que é e como funciona Linux Asynchronus I/O: http://www.ibm.com/developerworks/linux/library/l-async/
- Usar AIO somente com writeback e imagens RAW: https://access.redhat.com/kb/docs/DOC-40644
- Evaluating Linux storage APIs for use in QEMU/KVM

```
# kvm -drive aio=threads|native
```

Usando saída no terminal

```
kvm -curses
```

Boot ninja

```
# kvm -kernel /boot/vmlinuz -initrd /boot/initrd.img -append
"root=/dev/opa"
```

- Pode ser prático na hora de automatizar instalações, por exemplo chamando o instalador do Debian:

```
# wget -O squeeze-kernel-amd64 \
   http://ftp.nl.debian.org/debian/dists/squeeze/main/installer-
amd64/current/images/netboot/debian-installer/amd64/linux

# wget -O squeeze-installer-amd64 \
   http://ftp.nl.debian.org/debian/dists/squeeze/main/installer-
amd64/current/images/netboot/debian-installer/amd64/initrd.gz

# kvm -kernel squeeze-kernel-amd64 -initrd squeeze-installer-amd64 -
drive ...
```

- CentOS:

```
# wget -O centos-installer-amd64 \

http://mirrors.kernel.org/centos/5.6/os/x86_64/images/pxeboot/initrd.im
g
# wget -O centos-kernel-amd64 \

http://mirrors.kernel.org/centos/5.6/os/x86_64/images/pxeboot/vmlinuz

# kvm -kernel squeeze-kernel-amd64 -initrd squeeze-installer-amd64 -
append "ks=http://servidor" -drive ...
```

Uma VM totalmente pelada

- Não cria dispositivos criados automaticamente, como rede e vídeo.

```
# kvm -nodefaults
```

Melhorando a segurança

```
# kvm -chroot /tmp -runas goiabinha
```

- Ajustar permissões nos arquivos de imagem ou dispositivos no /dev
- Ajustar também as permissões das interfaces tap

QMP (QEMU Monitor Protocol)

- Funcionalidade que visa habilitar o controle remoto de máquinas virtuais através de um protocolo de controle.
- Ainda em desenvolvimento, mas quase estável.
- Visa substituir o monitor.
- http://wiki.qemu.org/QMP

```
# kvm -qmp tcp:localhost:5000,server,nowait
# telnet localhost 5000
```

```
{"QMP": {"version": {"qemu": {"micro": 50, "minor": 14, "major": 0},
"package": ""}, "capabilities": []}}
{"execute": "qmp_capabilities"}
{"return": {}}
{"execute": "query-kvm"}
{"return": {"enabled": true, "present": true}}
{"execute": "system_powerdown"}
{"return": {}}
{"timestamp": {"seconds": 1306447724, "microseconds": 321633}, "event":
"POWERDOWN"}
{"timestamp": {"seconds": 1306447725, "microseconds": 446323}, "event":
"RTC_CHANGE", "data": {"offset": -1}}
{"timestamp": {"seconds": 1306447727, "microseconds": 587542}, "event":
"SHUTDOWN"}
```

Spice

- http://spice-space.org/

qed

- http://wiki.qemu.org/Features/QED

Live migration

- No host A, inicie o kvm normalmente:

```
# kvm -drive file=/caminho/imagem-vm.img, ...
```

- No host B, inicie o kvm usando o mesmo comando do host A, porém com o parâmetro **-incoming**:

```
# kvm -drive file=/caminho/imagem-vm.img, ... -incoming tcp:0:4444
```

- Agora no monitor da máquina virtual em execução na host A, executar o comando **migrate**:

```
(qemu) migrate -d tcp:host_b:4444
(qemu) info migrate
Migration status: active
transferred ram: 17343 kbytes
remaining ram: 1023952 kbytes
total ram: 1057216 kbytes
```

- Quando terminar, a máquina continua sua execução no host B
- Cuidados
  - As imagens/disposivos de discos virtuais devem estar acessíveis no host de destino.
  - Interfaces tap devem estar na mesma rede local, geralmente, dependendo da topologia da rede.
  - As versões do KVM devem ser exatamente a mesma.
- **migrate -b** envia os dispositivos de bloco juntos (block migration).
- Mais dicas de live migration: http://www.linux-kvm.org/page/Migration

Windows e VirtIO

```
# kvm -drive file=/vm/win-
teste/windoze.qcow2,format=qcow2,cache=writeback,if=virtio \
      -m 2048 \
      -usbdevice tablet \
      -no-hpet \
      -fda /root/virtio-win-1.1.16.vfd \
      -drive
media=cdrom,file=/root/pt_windows_7_professional_x64_dvd_x15-
65787.iso,boot=on \
      -boot d
```

- Código fonte dos drivers: http://git.kernel.org/?p=virt/kvm/kvm-guest-drivers-windows.git;a=summary
- Drivers assinados pela Red Hat: http://alt.fedoraproject.org/pub/alt/virtio-win/latest/images/bin/
- Registro da certificação dos drivers:

http://www.windowsservercatalog.com/item.aspx?idItem=97ecf94b-6a63-e08b-22f8-b45e442e7d0c&bCatID=1282

- Installing Windows paravirtualized drivers

Eventos e atualidades

- Vídeos e slides do KVM Forum 2010
- KVM Forum 2011 já está confirmado para 15 e 16 de agosto.
- Red Hat Summit 2011.

Linux Containers (lxc)

- Linux Containers oferecem uma forma leve de virtualização, sem a necessidade de emulação de hardware.
- Assemelhando-se ao Jail BSD e Solaris Zones.
- Linux Containers permite isolar aplicações com recursos específicos, permitindo que estas executem sem interferir com outras aplicações, garantindo isolamento e performance.
- Virtualização em nível de sistema operacional.
- Vantagens:
  - Compartilham a mesma interface de chamadas de sistema e não há tem qualquer emulação.
  - Não necessita de imagens de disco, apenas do um sistema de arquivos.
  - Processos dentro do container são executados como processos normais.
  - Acesso à rede é baseado no isolamento.
  - Apenas um kernel para todos os containers.
  - Uso dos ricos recursos do Linux de escalonamento.
- Desvantagens:
  - Não tem suporte a livre migration.
  - Requer que o kernel de acolhimento deve ser corrigido.
  - Uma falha no kernel poderá ser explorada por todos os containers.
  - Todos os containers devem ser Linux.
  - Filtragem de pacotes pode ser tornar complicada.
  - Algumas chamadas de sistema, principalmente relacionados a hardware e partes do /proc / e /sys podem estarão disponíveis para os containers.
- Referências:
  - A Five Minute Guide to Linux Containers for Debian
  - LXC: Linux container tools

Aula 14 - Gerenciamento de ambientes virtualizados

Sobre

- Objetivos:
  - Apresentar os cuidados e necessidades de se ter uma ferramenta de gerenciamento.
  - Mostrar uma comparativo sobre as diferentes soluções.
  - Apresentar o Ganeti e especificar o projeto de cluster de máquinas virtuais para os alunos desenvolverem.

Ambientes pequenos

- Apenas um servidor (barato e rápido)
- Gerenciamento manual
- Backup limitado
- Sem redundância
- Não tem hardware para reposição rápida
- Alocação de VMs é manual (cp + rsync + lvcreate = magia negra)
- Sem acompanhamento da carga do hardware e se o que foi alocado é realmente necessário

Mais máquinas chegando, e agora?

- Com a chegada de mais servidores e o acúmulo de máquinas virtuais, se se continua trabalhando manualmente, o ambiente resultante simplesmente não tem como ser gerenciado.
- Toda a alocação de máquinas virtuais seria manual, não existirá redundância de nada e se caminhará cada vez mais para o colapso, pois concentremos mais serviços em menos hardware real.
- O volume de dados para se voltar de backup seria cada vez mais colossal, incluindo aí as próprias máquinas virtuais.

Quesitos para um ambiente campeão

- Escalabilidade
- Disponibilidade
- Monitoramento

- Balanceamento
- Acessibilidade
- Suporte

Escalabilidade

- Esse crescimento tem que ser sustentável e estruturado (aplica-se a qualquer ambiente, não?)
- A expansão do número de hosts tem que agregar recursos a todo o ambiente.
- Se forem compradas máquinas novas, essas máquinas tem que entrar no conjunto de hosts e estarem facilmente disponíveis para receber mais máquinas virtuais.
- A visão que precisamos ter sobre o total de disco e memória é do conjunto dos hosts, não individualmente de cada um.
- Se comprar mais máquinas, simplesmente terei mais recursos disponíveis (escalabilidade horizontal).

Disponibilidade

- Precisamos de redundância. Se pelo menos um for desligado, é possível manter todas as máquinas virtuais desse host em funcionamento, até que volte.
- Isso implica que os dados contidos no host A precisam ser replicados em tempo real para outros hosts.

Monitoramento

- Garantir boa performance e monitoramento dos recursos.
- O ambiente precisa ter um termômetro e um velocímetro para sabermos o que está acontecendo, checagem de integridade e estado geral.

Balanceamento

- O balanceamento dos hosts tem que ser fácil. Se no host A uma máquina virtual está precisando de mais CPU ou memória e no host B existem recursos disponíveis, essa transição tem que ser simples.
- A instalação, criação e alocação de novas máquinas virtuais tem que ser simples, rápida e o mais automatizada possível. Isso garante maior consistência e segurança no geral.
- A localização física da máquina virtual deve ser configurada automaticamente.

Acessibilidade

- Acesso a diferentes redes. Muitas vezes algumas máquinas virtuais precisam ter placas de rede em redes diferentes. Isso tem que ser possível.

Suporte

- Use distribuições sólidas e com suporte abundante, estável e confiável: CentOS , Red Hat, Debian , Ubuntu LTS
- Use ferramentas que também tenham suporte abundante

E quanto custa?

- Não menos importante:
  - Podemos ter tudo isso com baixo custo de implementação?
  - Não precisar de hardware sofisticado como um storage ou uma SAN?
  - Não depender de licenciamento de software proprietário?

Ferramentas livres de gerenciamento e cloud computing

- Existe uma selva de opções.
- Estão em constante e rápida evolução.
- Cada uma com suas vantagens e desvantagens.

O que procurar nessas ferramentas?

- Suporta diferentes Hypervisors? De maneira transparente?
- Tem comunidade ativa? É mantido apenas por uma empresa ou grupo fechado?
- Qual é a licença?
- Quantas pessoas e quem está desenvolvendo o código?
- É de fácil instalação? Já existe empacotado nas distribuições mais populares?
- Seria possível entender o código fonte?
- Quais são os requisitos de funcionamento? Storage? Servidores de armazenamento?
- Esquecemos de algo?

Projetos

- libvirt + Virt-Manager
- Ganeti + Ganeti Web Manager

- OpenStack
- RHEV-M 3.0
- oVirt
- ConVirt 2.0 OpenSource
- Eucalyptus
- OpenNebula
- Karesansui
- abiCloud
- OpenQRM
- Outros projetos, menções honrosas:
  - Xen Cloud Platform
  - SolusVM
  - Enomaly
  - Domain Technologie Control (DTC)
  - XenServer
  - OpenNode
  - Nimbus
  - LxCenter (Xen e OpenVZ)

Libvirt

- É o projeto mais antigo.
- Oferece uma API agnóstica de hypervisor para gerenciar máquinas virtuais.
  - Suporta: Xen, QEMU/KVM, LXC, UML, OpenVZ, VirtualBox, VMware ESX e GSX.
- Anatomia da biblioteca de virtualização libvirt: Uma API para facilitar a virtualização Linux
- Usa arquivos XML com definições razoavelmente abstratas, mas é possível especificar detalhes de cada Hypervisor quando necessário.
- É uma muito madura e robusta.
- Desenvolvida primeiramente pela Red Hat, mas com diversas outras contribuições externas.
- É utilizada por quase todos os gerenciadores como base.
- Para saber mais:
  - Libvirt: Hypervisor Independent Virtual Machine Management
  - Non-intrusive Virtualization Management using libvirt
  - Secure remote management with virtualization

Ganeti

- Ganeti foi pensado justamente para se fazer clusters de virtualização com baixo orçamento e funcionar em hardware humilde, sem depender de storage e SAN.
- Ele provê um mecanismo de replicação, baseado no DRBD
- Ele não reinventa a roda, usa apenas comandos e recursos disponíveis em qualquer distro Linux.
- O projeto é mantido principalmente por desenvolvedores do Google e outros.
- Puro Python
- Ganeti: An Open Source (GPLv2+) cluster manager for KVM
- Documentação do Ganeti
- http://www.lancealbertson.com/2010/05/creating-a-scalable-virtualization-cluster-with-ganeti/
- Creating a low-cost clustered virtualization environment
- The evolution of Ganeti, an Open Source manager for clusters of virtual machines
- DRBD: a distributed block device

Projeto do curso

- Implantar um ambiente de virtualização utilizando o Ganeti 2.4
- Documentar
  - Dependências
  - Instalação
  - Configuração e ajustes de cada nó
  - Comandos do Ganeti utilizados para criar o cluster, e o que é feito a cada etapa.
  - Como criar máquinas virtuais com e sem replicação.
  - Live Migration
- Grupos de até 4 pessoas
- Postar no wiki, será avaliado no final de semana dos dias 2 e 3 de julho.

Aula 15 - Atividade prática

- Fazer em dupla
- Utilizar 3 computadores, sendo
  - 1 computador para ser servidor de arquivos, usando NFS, onde ficarão as imagens.
  - 2 computadores para serem os hosts de execução.

Parte 1 - Montar um servidor NFS

- Conforme visto na aula anterior, precisamos de um armazenamento compartilhado entre os hosts para termos live migration.
- Existem várias maneiras de se montar um ambiente assim, para esse experimento devido às restrições de tempo, vamos usar NFS mesmo 😬
- Os caminhos abaixo são somente sugestões, configure de acordo com sua preferência.

```
apt-get install nfs-kernel-server
mkdir -p /export/images
mount --bind /var/lib/images /export/images
```

- /etc/exports

```
/export
10.1.1.0/24(rw,fsid=0,insecure,no_subtree_check,sync,no_root_squash)
/export/images
10.1.1.0/24(rw,nohide,insecure,no_subtree_check,sync,no_root_squash)
```

```
exportfs -r
```

- Nos clientes:

```
apt-get install nfs-common
mount -t nfs4 IP-SERVIDOR:/images /srv/
```

- Teste o acesso e gravação no ponto de montagem.

Parte 2 - Fazendo o live migration

- Utilize os comandos e instruções da Aula13 e execute o live migration.
- Execute a máquina virtual utilizando **cache=none** para os discos. Teria a mesma confiabilidade de se usar **cache=writeback**?

- Faça testes com ping e outros comandos, enfim, deixe algo em execução na máquina virtual enquanto ela é migrada. Por exemplo uma conexão SSH.
- Manipule os comandos do monitor **migrate_set_downtime** e **migrate_set_speed** e documente seus resultados e testes realizados.